

XIV CONFERENCIA LATINOAMERICANA DE INFORMATICA / CLEI

17as JORNADAS ARGENTINAS DE INFORMATICA E INVESTIGACION OPERATIVA / SADIO

BUENOS AIRES, 26 AL 30 DE SEPTIEMBRE DE 1988

TRABAJOS SELECCIONADOS

SOCIEDAD ARGENTINA DE INFORMATICA
E INVESTIGACION OPERATIVA

005.74
CONge
v.2.

MFN 824

PEDECIBA
INFORMATICA

COMITE EJECUTIVO

Rodolfo A. Boldt (Presidente)
Héctor Monteverde (Coordinador)
Clara Fuks (Tesorera)
Domingo Trassens (Prensa)
Juan Ignacio Hernández (Editor)

COMITE DE PROGRAMA

Jorge Vidart (Presidente) ESLAI, Argentina
Marcos Borges/ Núcleo de Computación, UFRJ, Brasil
Armando de Giusti / U.N. de La Plata, Argentina
Riccardo Ferrari / U.C.N. Sra. de Asunción, Paraguay
Alberto Flores Balanza / U.N. San Andrés, Bolivia
Pedro Hepp / U.C. de Chile, Chile
Ida Holz / U. de la República, Uruguay
Rodrigo López / U. de los Andes, Colombia
Diómedes Sarmiento / APCI, Perú
Amelia Soriano / U.C. de Venezuela, Venezuela
Alberto Teszkiewicz / UBA, Argentina

17/10/88 RETIRADO Nº 272

T O M O 2

I N D I C E

	Página
CAPITULO 6: SOFTWARE PARA TIEMPO REAL	
A Simple Implicit Dictionary with Polylog Average Cost Ricardo A. Baeza - Yates - Canadá	544
Software para control de tiempo real: La concepción estructurada de un Bios H. Fabián Torres, Marcelo Cardós, Jose Lizzoli, Armando De Giusti - Argentina	558
Procesamiento de eventos para sistemas de tiempo real Wolfgan A. Halang - Alemania	576
JOYCE+ , Modelo, lenguaje, metodología de programación de sistemas distribuidos sobre redes de comunicación María Consuelo Franky - Colombia	590
CAPITULO 7: REDES	
Análisis de un sincronizador para datos codificados convolucionalmente Oscar Bria, Hugo Llorente, Carlos Muravchik- Argentina	616
Efecto del uso de pre-énfasis y de-énfasis en enlaces digitales de AFSK FM Hugo Lorente - Argentina	631
Evaluación de comportamiento de protocolos de redes locales normalizadas en ambientes críticos J.Santos, J.Orozco, O.Alimenti - Argentina	647
Uma Analise do servico quarentena de dados local com modo de diálogo Semi-Dúplex María Cavalcanti Cabral - Brasil	667
Protocolo de comunicación entre Host y Front-End Proccesor J. Mechoulan - Marcela A. Pérez - Argentina	685
Servicos de transferencia de arquivos em redes locais de automacao industrial Teresa Melo de Brito Carvalho - Brasil	705
Diseño de una Red de Transmisión de Mensajes J.A. Aguirre, N. Ambrogetti, M. Ramos Martínez, Paulina Frenkel - Argentina	730

CAPITULO 8: EVALUACION DE SISTEMAS

MACOS: Modelo algoritmico de costos de Software 744
L. Guerra, S. Concha - Chile

Modelo de desempeño de un sistema computacional universitario 755
Francisco Ovalle, F. Aurtenechea, I. Casas -Chile

CAPITULO 9: INTELIGENCIA ARTIFICIAL

APREND - Um sistema para aquisicao automatica de cohecimento a partir de exemplos 778
F. Carvalho Gomes, Giuseppe Mongiovi, H. de Menezes Silva - Brasil

Uma abordagem geral sobre crenca e conhecimento no contexto social 794
Juan C. Mangione - Brasil

Computing certainty factors in Expert Systems Using Dihypergraphs Flows 816
J. Aráoz, E.Broner, A. Torres - Venezuela

Verificacao de base de conhecimento 828
José C. Barbosa Damski -Brasil

Una interfase de lenguaje natural para programar 838
Adolfo Kvitca - Argentina

Gramáticas de determinacao: uma proposta metodológica 858
Clarisse Sieckenius de Souza

Correcao de erros de ortografia através da fonética 873
J. Courtin, I. Kowarski, D. Dujardin, D. Genthial, De Lima Vera - Francia

Tratamiento de textos de química en castellano 892
Osvaldo Cairó Battistutti - Méjico

PROSEDEM: Prototipo de Sistema Experto de Diagnóstico de enfermedades metabólicas en ganado vacuno 912
M.S. Frassoni, Néstor Vigil - Argentina

Um sistema especialista para previsao de tempo 929
Vera de Avila Duarte, Fernando de Castro Giorno Brasil

Knowledge engineering: Prototyping a Small Knowledge Based System 939
Walter Rolandi - Estados Unidos

CAPITULO 10: INFORMATICA EN LA UNIVERSIDAD

Un laboratorio de estructuras de datos 948
Jorge Villalobos, Mario Otero, Alejandro Quintero, Olga Mariño - Colombia

LUBO- 1: Un modelo de máquina abstracta para un primer curso universitario de programación 962
Patricia Pesado, María Madoz, Lía Molinari, Armando De Giusti - Argentina

CAPITULO 11: INVESTIGACION OPERATIVA

Estrategias para el control de señales de tráfico urbano 976
Sergio Di Gerónimo, M. Fernández Romero- Argentina

Nuevas técnicas en el Análisis de series de tiempo discretas: aplicación en modelos económicos 998
Oscar Liberman, Fernando Tohme - Argenatina

Un algoritmo para el cálculo de la prueba espectral en el caso de generadors congruenciales lineales GCL (A.C.M.) con período máximo 1020
Rafael Pastoriza - Venezuela

Técnicas eficientes para generar valores normales y lognormales por el método del cociente de uniformes 1028
R. Lavieri, R. Pastoriza - Venezuela

Implementación de un módulo optimizador en un simulador de procesos químicos 1038
Jorge Montagna, Oscar Iribarrea - Argentina

Optimización de sistemas dinámicos vía perturbación singular de la ecuación de Hamilton-Jacobi-Bellman 1053
Laura Aragone - Argentina

Problemas de optimización de sistemas económicos con criterios cuadráticos y controles monótonos. Un algoritmo para su solución numérica 1067
Laura Aragone - Argentina

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

A Simple Implicit Dictionary with Polylog Average Cost

Ricardo A. Baeza-Yates
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1 *

Abstract

We study different deletion algorithms in a simple implicit data structure that had already known $O(\log^2 n)$ worst case time for searches and insertions. Experimental results shows that the dominant term in the deletion cost is the number of moves, giving a time complexity of $O(\log^3 n)$.

1 Introduction

Bentley *et al.* [1] introduced a simple implicit data structure based in sorted lists. They store a set of n elements in n consecutive storage locations which are viewed as $\lceil \log(n+1) \rceil$ or fewer sorted lists. [1] There is one list of length 2^i for each i such that the i^{th} bit of the binary representation of n is 1.

In this work we summarize known results of this structure and we discuss the application of this structure to the dictionary problem, in particular to support deletions. We use the RAM model, counting comparisons and movements of data values.

2 Search and Insertion

Using binary searches we have a $\log^2 n/2$ worst case search algorithm. If the lists are searched in decreasing order of size, we have a $2 \log n$ comparison algorithm on the average [1].

Inserting an element requires only the addition of a list of size 1. If there is already a list of size 1, the two are merged into a list of size 2. This process continues as long as possible. In the worst case, an insertion produces at most $\log n$ merges, and then $O(n)$ comparisons and moves in the worst case. However, any sequence of n insertions requires at most $n \log n/2$ comparisons and the same number of moves. That is, the amortized worst case of n insertions is $O(\log n)$ time per insertion [1].

*This work was supported by the Institute of Computer Research of the University of Waterloo and by the University of Chile.

[1] All logarithms are in base 2

If we spread out the merging cost between n insertions, and we maintain at the same time $O(\log n)$ searching in each list, we have a $O(\log n)$ worst case bound per insertion with a more complicated algorithm [4].

Hence, we have a structure that supports insertions in $O(\log n)$ worst case time and searches in $O(\log^2 n)$ worst case time and $O(\log n)$ average case time. Given the simplicity of the structure, this result is very interesting compared with the current best implicit structure for the dictionary problem, that supports the same operations and deletions in $O(\log^2 n)$ worst case time [3].

3 Deletion

Proposed Algorithm

In [1] a few approaches to deletions were suggested. For example, flagging elements. However, this implies 1 extra bit per element (that is, the structure is not implicit any more) and a small increase in run-time. In [4] a different approach was suggested. If we use a redundant binary system (with the digits 0, 1, and 2, where 2 means 2 lists of that size) we can guarantee that there is always at least one list of each possible size. We can see this as having the elements represented in 2 structures, an adding (bitwise) the number of elements of each structure we get the redundant binary code. In general, there is more than one representation in this encoding for each number. We use the one that does not have any 0 digit. The algorithm for a deletion is

```
Delete( a, x )
Sorted_list a;
Element x;
{
   Find and remove x, this leaves a hole in a list with 2**r elements
   for( i = r-1; i > 0; i = i-1 )
   {
       Find a[k] < x < a[k+1] in a list of size 2**i
       Replace x in list r by a[k] or a[k+1] and reorder the list
       x = a[k] or a[k+1]
       r = i
   }
   If there is no lists of length 1; initiate a sequence of "unmerges"
      analogous to the merges required by an insertion
}
```

Clearly, we have three different costs. First, the searching cost to find x. This will be $O(\log^2 n)$ comparisons. Second, the moves and comparisons to "unmerge" if at the end of the deletion no lists of size 1 exists. Using the same idea when inserting, we spread out the cost of the unmerges, having an $O(\log n)$ worst case cost for "unmerges". Here, we will have lists being unmerged and lists being merged. In some cases, an insertion will transform a list in an "unmerging" state to the normal state. The same with deletions, and a list being merged. To keep these states we will need 2 other digits, however, we still need only $O(\log n)$ bits to represent the structure.

Note that the "unmerge" cost is independent of the sequence of replacements used and the cost of searching x is independent of the algorithm used to replace x. Hence, these costs will be always $O(\log^2 n)$ in the worst case and we will not discuss them any more.

Finally, we have the moves and comparisons to find the replacement elements and to replace x and reorder the lists. This is the interesting cost.

We will concentrate our analysis into the number of moves. Clearly, in the worst case, the number of moves is $O(n)$. What about the average? We can divide the number of moves in two parts. The first one, or *basic moves* ($\overline{M}_b$) will account for all the replacements. In the worst case and in the average, we have $O(\log n)$ replacements in the proposed algorithm. The second part, will account for all the moves necessary to reorder the lists (*internal moves*, $\overline{M}_i$). The analysis for this quantity is not trivial. If we assume that the distribution in all the lists remain uniform, we would expect 1 or 2 extra moves per list, giving a $O(\log n)$ time on the average. However, experimental results does not support this assumption [2,4].

The number of comparisons needed to find the replacements $\overline{C}_r$ depends on the algorithm used, and will be analyzed later. The number of comparisons needed for the reordering ($\overline{C}_o$) will be proportional to the number of moves. More specifically

$$\overline{M}_i \leq \overline{C}_o \leq \overline{M}_i + 2\overline{M}_b$$

because we have one comparison for each internal move performed, and in the worst case, we have 2 test comparisons for each basic move.

Replacement Heuristics

The first issue in the given algorithm, is how to replace x. If we have more than one list of the same size, we have 2 obvious choices:

1. Random choice of one of the lists
2. Choose the list that provides the closest element to x

Obviously, the second choice will be better, and only increases the search time by at most a factor of 2.

Suppose that we have now only one list. If we have two choices for replacing x, which one we use? We look at 4 possible alternatives:

1. Random: We flip a coin to choose which element we use.
2. Alternation: We alternate between the right and the left element.
3. Closest: We choose the one closer to x. That is, the one that minimizes the absolute difference between x and the element.
4. Nearest to one side: We choose the one that is nearest to one of the ends of the list. The idea is that (hopefully) if we have to replace the minimum or the maximum of a list, the probability of reordering will be smaller.

Figure 1 shows the experimental results for update cycles (see next section) for $n = 2^{11} - 1 = 2047$ elements. Clearly, the closest heuristic outperforms all the others, and the last heuristic (surprisingly) is the worst one. In the following, we use the *closest* element heuristic unless we explicitly state something different.

Update Cycles

In the following, we assume that n is of the form $2^k - 1$ (the basic problems are captured without loss of generality) and to measure the number of moves we will consider a sequence of updates (deletion/insertion pairs). We delete a randomly chosen element and we insert a random value from a uniform distribution in the range [0..1]. For this type of n, the insertion is very simple, because after the deletion there is no list of size 1.

In this case, we have exactly $k - 1$ basic moves in the worst case, and

$$\overline{M}_b = \sum_{i=1}^{k-1} \frac{i\ 2^i}{n} = k - 2 + \frac{k}{n} = \log(n+1) + O(1)$$

on the average. We will use this measure to check our empirical results. In the worst case, the number of internal moves is bounded by

$$0 \leq M_i \leq n - k = n - \log(n+1)$$

The analysis of the expected number of internal moves is not simple, even for the first update. In [4] a simple analysis gives $4/5\ (\log(n+1) - 2)$ internal moves for the first update, and the experimental results supported the hypothesis that the constant was $1/2$ instead of $4/5$.

Experimental results in [2,4] showed that the average number of internal moves was not logarithmic and a bound of $\log^2 n/2$ was conjectured. The structure seemed to converge after $O(n)$ updates to a non-uniform distribution, where clusters of elements appeared. Figure 2 shows the change in the distribution variance (initially uniform) for the lists of size 16 to 1024 for $n = 2047$ with several updates. Clearly, the distribution is less uniform for small lists. The number of updates necessary to reach the stable distribution is approximately $3n$ and after that small oscillations continue to exist.

Exact Analysis for Small n

For the analysis we distinguish each possible configuration of the lists using only the ranks (not the values) of the elements, and we associate each configuration to a state in a Markov chain. For example if $n = 3$, the possible configurations are $[(1,2)(3)]$, $[(1,3)(2)]$, and $[(2,3)(1)]$. For example $[(1,2)(3)]$ means that the elements of rank 1 (minimum) and 2 are in the list of size 2, and the maximum element (rank 3) is in the list of size 1. If we have more than one list of each size, then we use a lexicographical order on those lists. Because each list is sorted and each element has a unique rank, each list must be an ascendant sequence of ranks. For $n = 2^k - 1$ the total number of configurations is

$$\binom{n}{2^{k-1}} \prod_{i=1}^{k-2} \binom{n - \sum_{j=i+1}^{k-1} 2^j}{2^i}$$

When we delete an element, we delete its rank from the configuration and we rank the remaining elements again. If we have that 2 choices are possible for a replacement, and both ranks are at the same distance to x, we assign the same probability to each element $(1/2)$ of being the closest one. In the case of an insertion, we append a singleton, and we rank all the elements again. Each possible rank of the new singleton is equally likely, because the overall distribution of all the elements is always uniform.

For example, if we delete the element of rank 3 from $[(2,3)(1)]$ we obtain $[(1,2)()]$. Now, if we insert a big element, we obtain $[(1,2)(3)]$. Figure 3 shows the Markov chain for $n = 3$. The solution in this case is very easy. In the steady state of the Markov chain (that is, asymptotically in the number of updates), the probability of being in each state is the same $(1/3)$. In all the states with probability $2/3$ we have a basic move and in states 1 and 3 with probability $1/3$ we have an internal move. This gives an average of $2/3$ basic moves and $2/9$ internal moves per update.

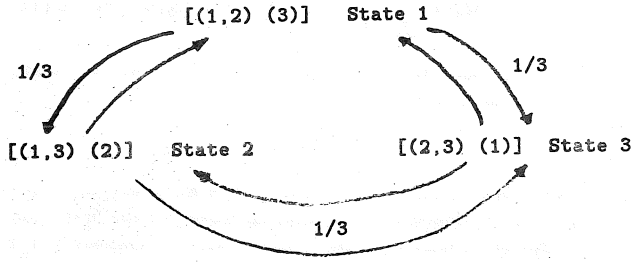


Figure 3: Markov chain for $n = 3$.

Table 1 shows the expected number of moves for $n < 8$ with the experimental results between parentheses. In some cases we give real numbers because the fractional values are too long to be printed.

n	Number of States	Binary Redundant Code	Basic Moves $(\overline{M_b})$		Internal Moves $(\overline{M_i})$	
3	3	11	2/3	(.667± .001)	$2/9 = .2\overline{2}$	(.2223± .0001)
4	6	12	1/2		5/52 = .096154	
5	15	21	4/5		$4/15 = .2\overline{6}$	
6	45	22	2/3		.140665	
7	105	111	10/7=1.42857	(1.429± .002)	.58253	(.5984 ± .0003)
8	420	112				

Note that the number of moves is not monotonic between $n = 3$ and $n = 7$ (but it is monotonic in n of the form $2^k - 1$). Also, the experimental results (95% confidence interval, 100 runs, 50000 updates) are in good agreement with the analysis.

Figures 4a and 4b gives the experimental results for different k and different number of updates. Figure 5 shows the asymptotic expected number of basic moves and the theoretical results; the agreement is perfect. Figure 6 shows the asymptotic expected number of internal moves. A least squares approximations in a model of the form $a(\log^2(n+1)-1)$ gives $x = 2.8617$ and $a = 0.0184$ using 14 values. The error is approximately 0.717. Including lower order terms, we found that a model of the form

$$(a\log^2(n+1)+b)(\log(n+1)-1)$$

was better (note that again we have only 2 parameters). In fact the error was only 0.103, giving $a = 0.0127$ and $b = 0.209$ (this curve is also given in Figure 6). Then, we have

$$\overline{M}_i = 0.013\log^3(n+1) - O(\log^2(n+1))$$

This suggests that the number of internal moves per list is $O(\log^2 n)$. Therefore, the number of internal moves is the dominant cost (asymptotically). The constant of the dominant term is so small that the number of comparisons to find the replacements,

$$\overline{C}_r = \log^2(n+1)/2 + O(\log n)$$

is bigger than $\overline{M}_i$ for $n < 4 \times 10^{11}$. The overall cost for this deletion algorithm is $O(\log^3 n)$ moves and comparisons.

Improving the number of internal moves

If the internal moves are the dominant cost, what we really want is to minimize the number of internal moves to remove the singleton. In general, the optimal sequence of replacements depends on the current state of the data structure. For that reason, we are using a heuristic that will try to work well for almost all the cases.

In the *closest* heuristic is implicitly assumed that a good sequence of replacements must look each time to the next smaller list. However, in general this is not true. May be, the optimal sequence (in the sense of minimal number of internal moves) is to go to some list, and then go back to a bigger one and so on (see Figure 7, case a). Also, it is possible that the optimal sequence visits a list more than once (see Figure 7, case b) or does not visit all the lists (see Figure 7, case c).

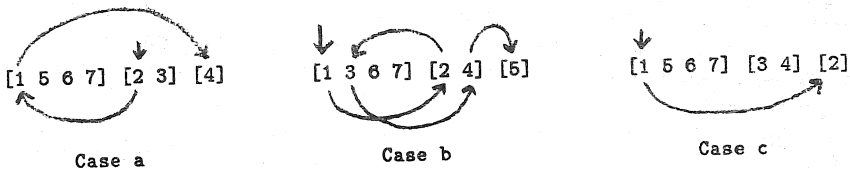


Figure 7: Examples of optimal sequences of replacements.

Intuitively, on the average, the *closest* heuristic seems to be a good approach. However, we have only a 50% chance that the closest element is in the next list rather than in all the others smaller

lists (assuming a uniform distribution). One way to increase this chance, it is to look to the closest in all the smaller lists and then use that element. The drawback, is that now the complexity of the searches is $O(\log^3 n)$ and we are looking for something better. One compromise between both schemas, it is to look at the 2 next smaller lists for the replacement element (that is a 75% chance to find the closest one). This introduces at most a factor of 2 in the searching time and hopefully will give a more stable behavior to the structure. This seems to happen based in the experimental results. Now, approximately only $2n$ updates are needed to reach the stable distribution and the oscillations that persist seems to be smaller. The period of the oscillations also seems to be shorter.

Suppose now, that the replacement element (the closest of 2 lists) is in the next list with probability p, and with probability $1-p$ in the next one. If the distribution were uniform, we would expect $p = 2/3$. However, in general, p depends on n, and on the size of the current list. Hence, the expected number of replacements, if we delete an element in a list of size 2^j, is given by the recurrence

$$T(j) = 1 + pT(j-1) + (1-p)T(j-2)$$

with $T(1) = 1$ and $T(0) = 0$. The solution is

$$T(j) = \frac{j}{p-2} + \frac{1-p}{(2-p)^2}(1-(p-1)^j)$$

Then, the expected number of basic moves is given by

$$\overline{M}_b = \sum_{i=1}^{k-1} T(i)\frac{2^i}{n} = \frac{2p^2-7p+6}{(3-2p)(p-2)^2}\log(n+1) + \frac{9-9p+2p^2}{(3-2p)(p-2)^2} + O\left(\frac{\log n}{n}\right)$$

For $p = 2/3$ we have $3\log(n+1)/4 - 21/16 + o(1)$.

The same Markovian analysis for $n = 7$ (in the other cases there are no differences) gives $\overline{M}_b = 1.21101$ and $\overline{M}_i = .50284$. This approach minimizes the number of internal moves for this n, however does not minimizes the total number of moves. In fact, if we always use the singleton as the replacement element, for $n = 7$, we have $\overline{M}_b = 6/7 = .85714$ and $\overline{M}_i = 2/9$. For small n this heuristic is better and we will use this fact ahead.

The analysis for $n = 7$ suggests a value of $p \approx .62$ for this case instead of $2/3$. However, the experimental values $\overline{M}_b = 1.262 \pm .006$ and $\overline{M}_i = .5332 \pm .0009$ indicates a value of $p \approx .71$. For bigger n, p seems to approach $2/3$ (see Figure 5 for the experimental and theoretical curve with $p = 2/3$). Figures 8a and 8b shows the experimental results. The asymptotic values for the internal moves are shown in Figure 6.

For a model of the form $a\log^x(n+1)$ we obtained $x = 2.575$ and $a = 0.0247$ with an error of 0.458. Using a model of the form $(a\log^{3/2}(n+1)+b)(\log(n+1)-1)$ we obtain a much smaller error (0.021) with $a = 0.0308$ and $b = 0.0714$ (this curve is shown in Figure 6). Hence, the dominant cost is still the number of moves, but the complexity is now

$$\overline{M}_i = 0.031\log^{5/2}(n+1) - O(\log^{3/2} n)$$

This suggests that the number of moves in each list is $O(\log^{3/2} n)$. The number of comparisons $\overline{C}_r = \log^2(n+1) + O(\log(n+1))$ is bigger than the number of internal moves for $n < 3 \times 10^{334}$ (!!).

Improving the overall cost

An optimal algorithm in the number of comparisons will be any method that does not search for a replacement element. For example, to use always the singleton to replace the hole. In general, this is not a good idea, because the number of internal moves is then linear on average. Considering the singleton as a random element to be inserted in a uniform distribution, we have that the number of internal moves in a list of size s is

$$M(s) = \frac{1}{s}\sum_{i}\frac{1}{s+1}\sum_{j}|i-j| = \frac{s-1}{3}$$

where i are all possible positions for x, and j are all possible positions for the singleton. The expected number of internal moves is then

$$\overline{M}_i = \sum_{i=2}^{k-1}\frac{2^i\ M(2^i)}{n} = \frac{n-1}{9}$$

Experimental results are in good agreement with this formula. Both curves are shown in Figure 6.

If we use a cost function of the form

$$Cost = \alpha(\overline{M}_i + \overline{M}_b) + \beta(\overline{C}_o + \overline{C}_r)$$

we may find a hybrid algorithm that optimizes this cost. Suppose $\alpha = \beta = 1$. In that case, the overall cost of the dumb algorithm is better for small n. Based on empirical results the dumb algorithm is better than the original deletion algorithm for $n < 200$ (approximately). Based in the empirical models and our analysis the overall cost for the original algorithm is better than the new algorithm proposed for any $n < 1 \times 10^{17}$. Hence, for practical values of n, the original algorithm runs faster (see Figure 9).

4 Final Remarks

For large n, we can combine all the different ideas presented in this paper. For example, intuitively the biggest number of internal moves, on average, will be in the biggest list. Hence, why not trying to find the closest one on the next 2 lists in that case? Also, if we are deleting from a small list, why not using the singleton element for the replacement. All these decisions will depend on the application.

For implementation details we refer the reader to [1]. A simple non-implicit implementation (but still using less space than a search tree or other classic dictionary structure) is to use a table of pointers of size $O(\log n)$. The total space is then $n + O(\log n)$.

References

[1] Bentley, J., Detig, D., Guibas, L. and Saxe, J. "An Optimal Data Structure for Minimal Storage Dynamic Member Searching", Dept. of Computer Science, Carnegie-Mellon University, 1978.

[2] Chan, V. "Simulation of Insertion and Deletion in Binomial List", Master Essay, Dept. of Computer Science, University of Waterloo, 1982.

[3] Munro, J.I., "An Implicit Data Structure for the Dictionary Problem that Runs in Polylog Supporting Insertion, Deletion and Search in $O(\log^2 n)$ Time", *JCSS* 33 (1986), 66-74.

[4] Munro, J.I. and Poblete, P., "Searchability in Merging and Implicit Data Structures", *Proc. ICALP*, Lecture Notes in Computer Science 154, Springer-Verlag, New York/Berlin, 1983, 527-535.

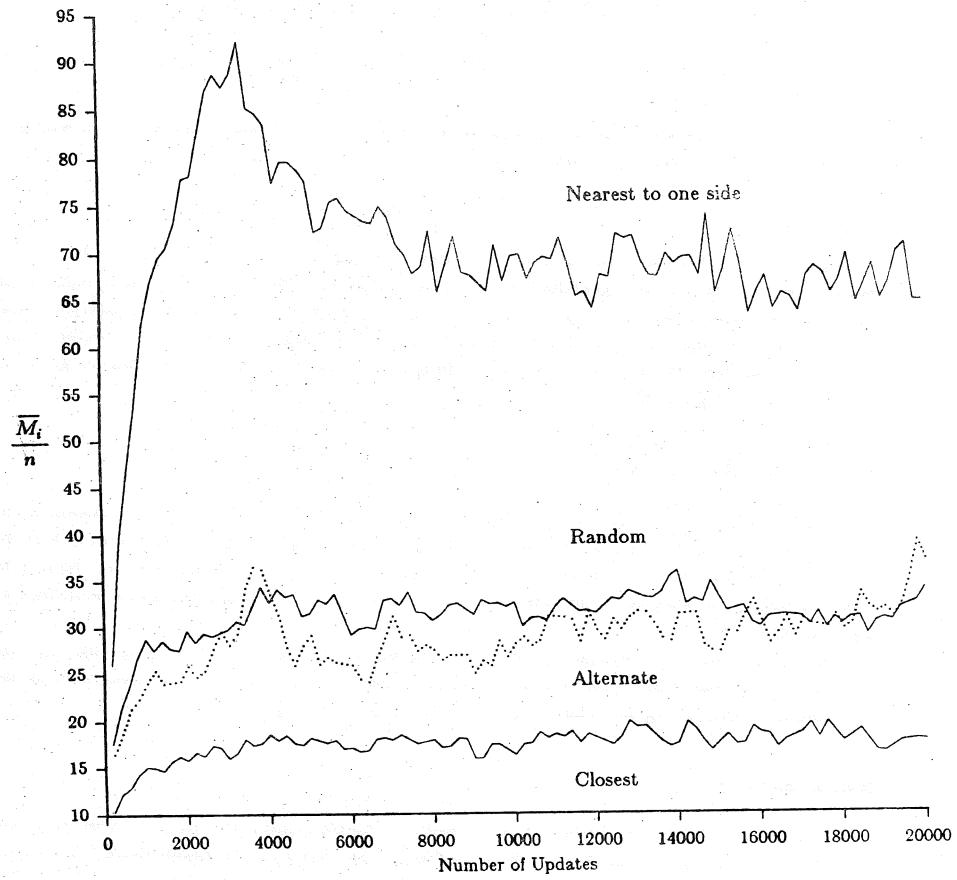


Figure 1. Simulation results for different heuristics (n=2047).

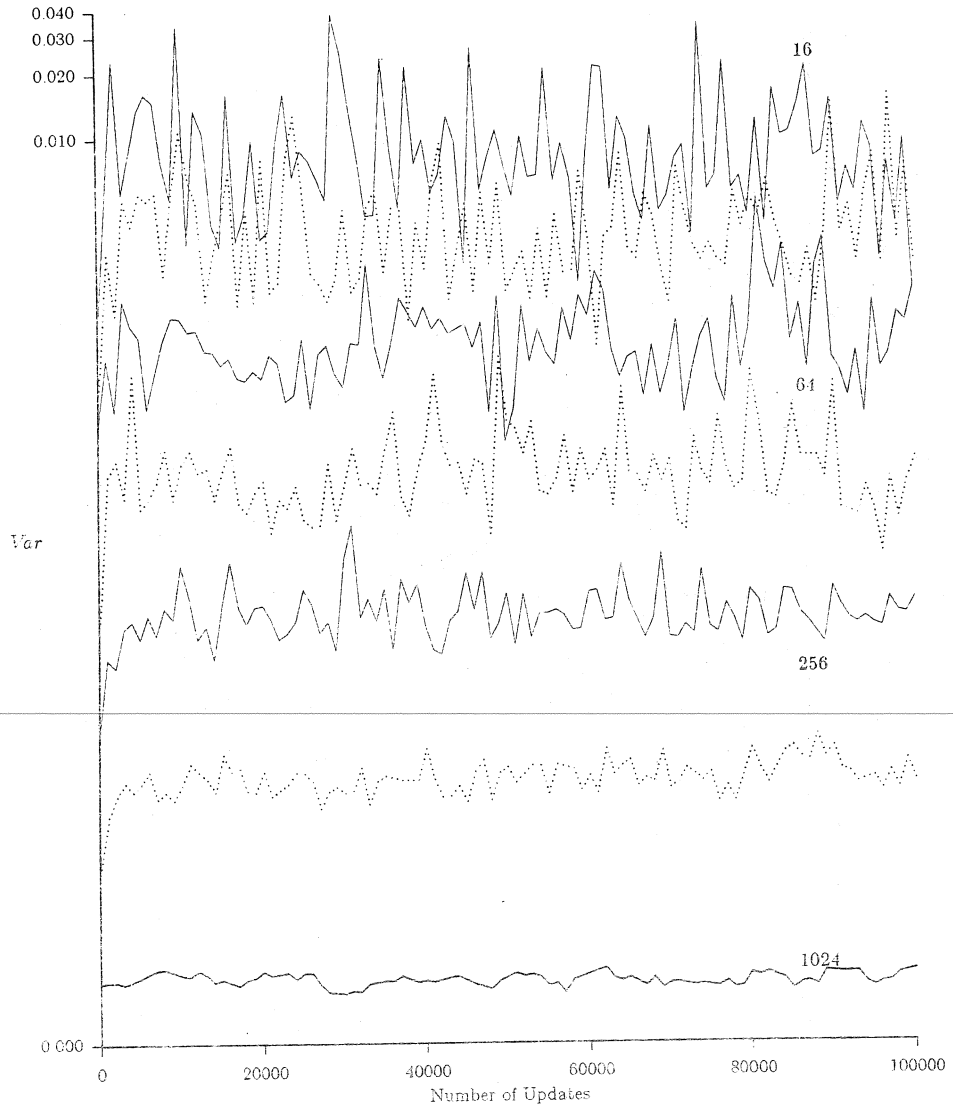


Figure 2. Simulation results for the variance on each list for $n=2047$.

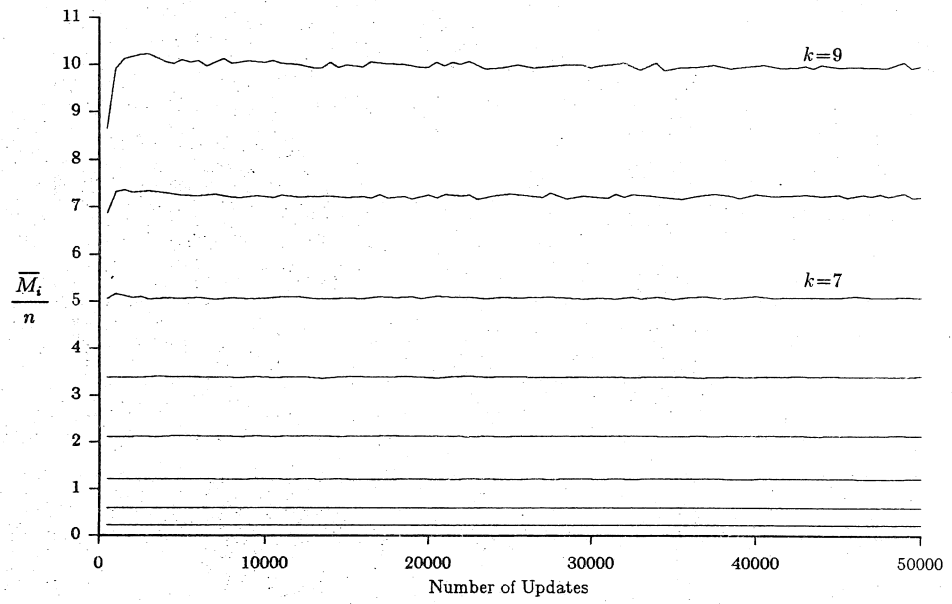


Figure 4a. Simulation results for $n=2^k-1$, $k=2..9$

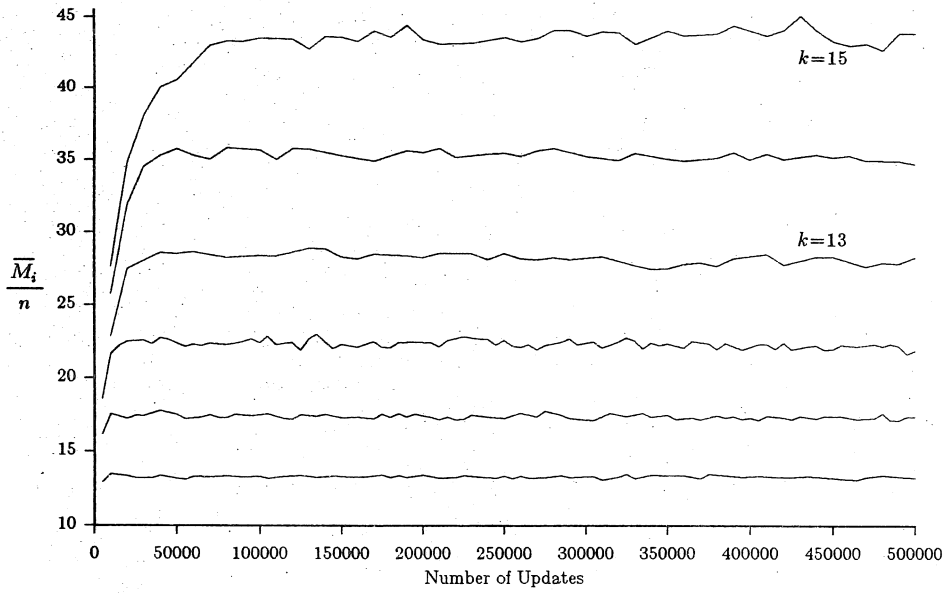


Figure 4b. Simulation results for $n=2^k-1$, $k=10..15$

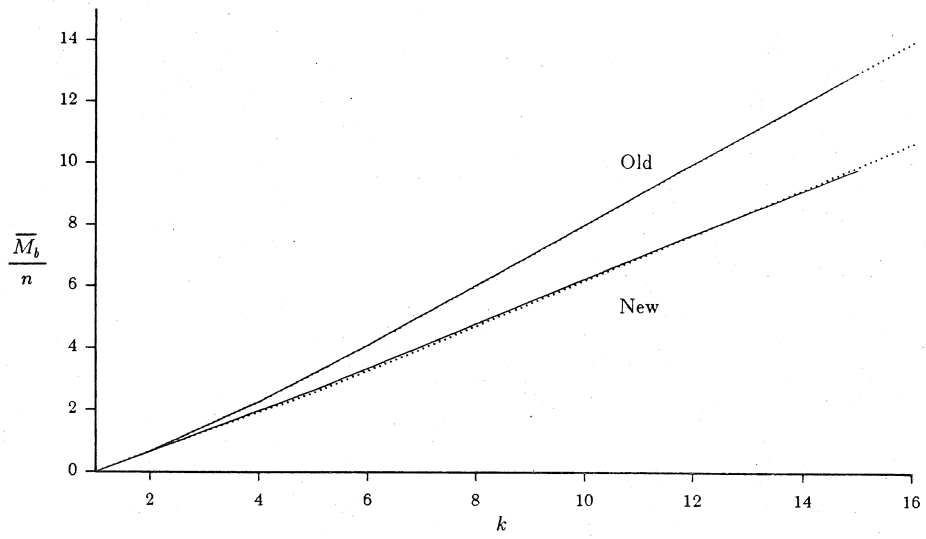


Figure 5. Asymptotic simulation results for $n=2^k-1$, $k=2..15$

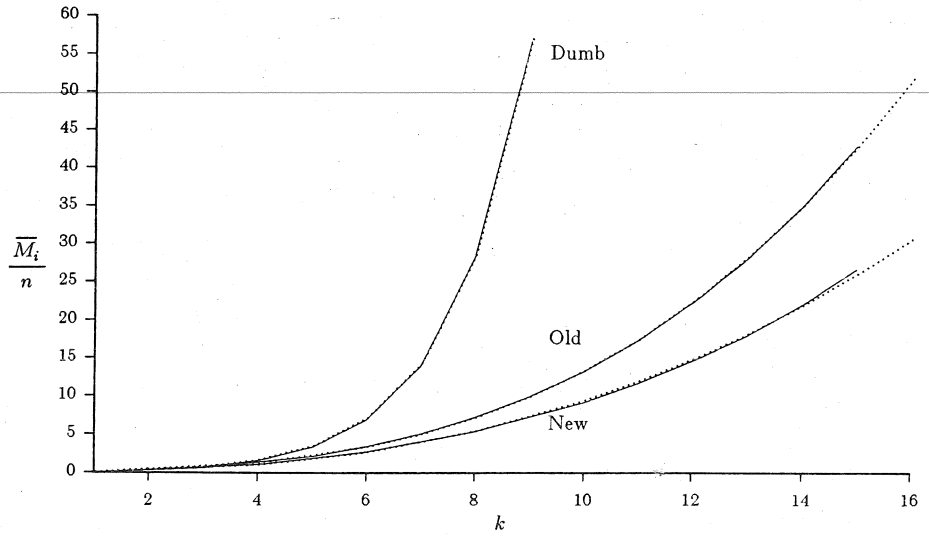


Figure 6. Asymptotic simulation results for $n=2^k-1$, $k=2..15$

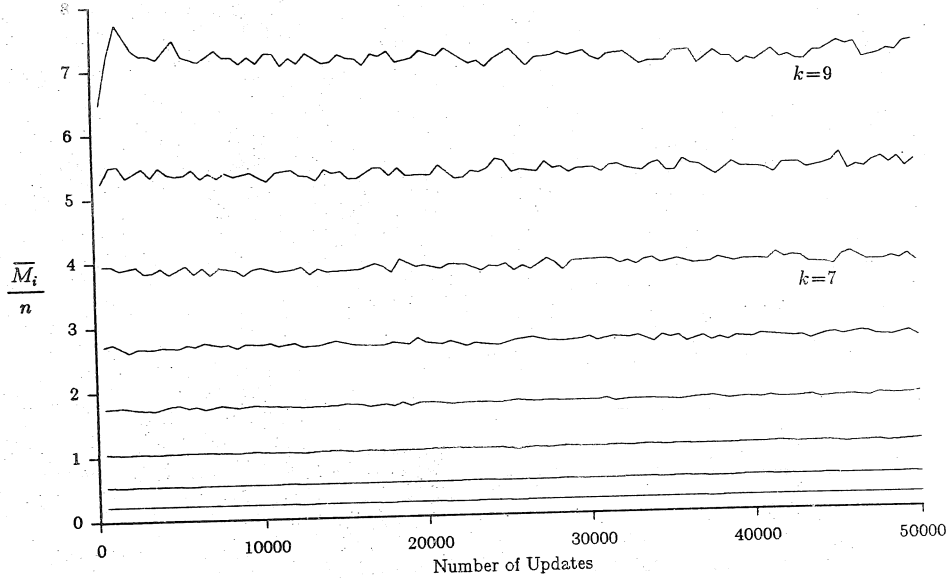


Figure 8a. Simulation results (new algorithm) for $n=2^k-1$, $k=2..9$

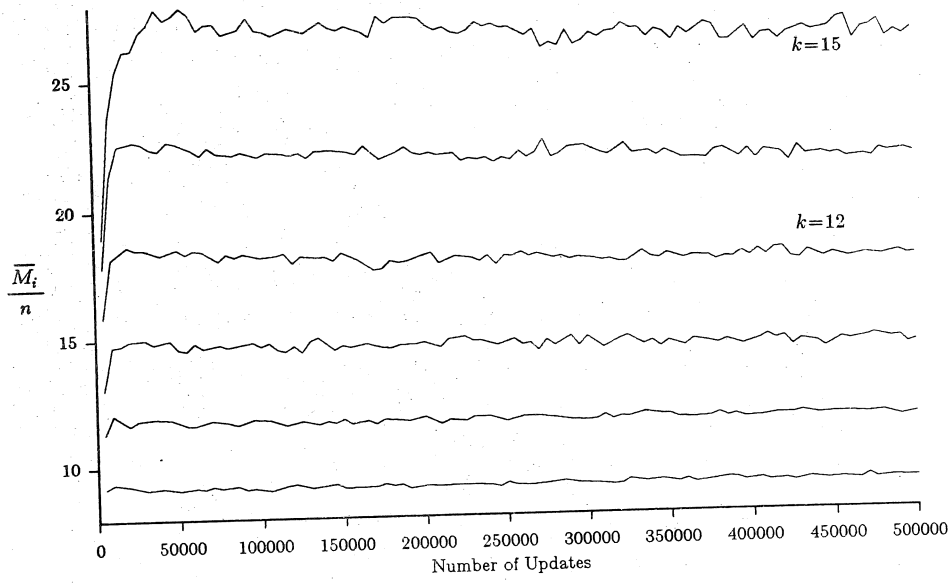


Figure 8b. Simulation results (new algorithm) for $n=2^k-1$, $k=10..15$

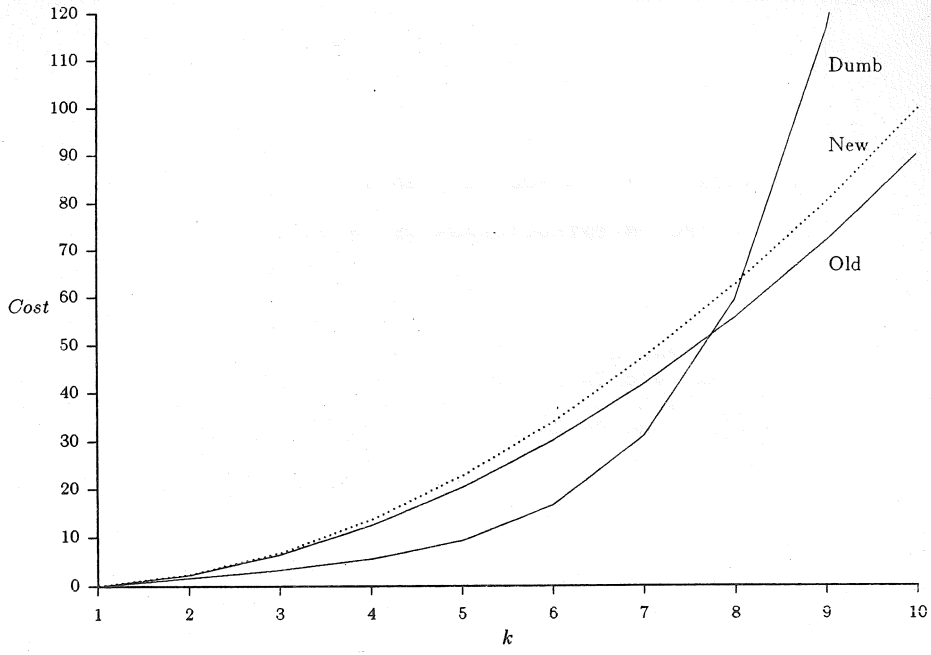


Figure 9. Total experimental cost for $n=2^k-1$, $k=2..9$

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

SOFTWARE PARA CONTROL DE TIEMPO REAL:

LA CONCEPCION ESTRUCTURADA DE UN BIOS

H. Fabián Torres. (*)
Marcelo Cardós. (*)
José Lizzoli. (*)
Armando De Giusti. (**)

(*) Alumno avanzado de la Licenciatura en Informática.

(**) Profesor Titular de la Fac. de Cs. Exactas de la UNLP.
Investigador Independiente del CONICET.

RESUMEN:

Se presenta el desarrollo de un BIOS para microcomputadora, orientado a ser utilizado en aplicaciones de tiempo real.-

A partir del estudio detallado de un BIOS convencional, se analiza su implementación en lenguaje C y la posibilidad de extender y adecuar sus posibilidades para sistemas de control dedicados.-

Se presentan medidas de rendimiento del software desarrollado y las extensiones que permiten controlar hardware "no convencional" desde el mismo BIOS.-

INTRODUCCION:

El desarrollo de sistemas de tiempo real, y especialmente de control dedicado, basados en microprocesadores ha caracterizado el área electrónica desde mediados de la década del 70, en nuestro país y en el mundo (Ref. 1,2,3,4).-

La primer etapa tecnológica relacionada con estos desarrollos se caracterizó por los equipos "a medida" desde el punto de vista del hardware y la utilización de lenguaje assembler desde el punto de vista del software (Ref. 5,6).-

Simultáneamente el desarrollo explosivo del mercado de microcomputadoras personales (PC's) y el establecimiento de arquitecturas estándar produjo un notorio abaratamiento de los costos de este tipo de hardware y el desarrollo de una serie de productos de software de carácter general (Sistemas Operativos, Lenguajes, utilitarios) permitieron extender sus aplicaciones originales de cálculo y procesamiento de información al desarrollo de sistemas dedicados basados en arquitecturas tipo PC (Ref. 7,8).-

Una rápida caracterización de un sistema de tiempo real nos permite apreciar tres "capas" o niveles tanto de hardware como de software:

1- La capa más "externa" está constituida por la interfaz entre el fenómeno de tiempo real tratado y el sistema digital que lo procesa. En ella habitualmente existe algún tipo de conversión de tipo a nivel de señal y las soluciones son dedicadas según las características del fenómeno que se está tratando.-

2- La capa de "comunicación" es la que permite resolver la recolección y trasmisión de los datos al procesador y de éste al subsistema externo. Habitualmente es una capa "crítica" desde el punto de vista de los tiempos involucrados y dadas las velocidades muy diferentes de los distintos procesos que pueden estar

en curso, es característico que dicha comunicación se base en interrupciones asincrónicas.-

3- Por último la capa más "interna" está constituída por el procesamiento puro de la información ya convertida. Habitualmente el software desarrollado para ella no se diferencia de cualquier software de aplicación, salvo en características de contexto como pueden ser la memoria disponible o el hecho de que la versión definitiva debe residir en ROM y por lo tanto debemos generar un módulo ejecutable que sea reubicable a partir de una dirección específica.-

Si se analiza desde el punto de vista de software, normalmente el trabajo de mayor envergadura está en la capa de comunicación:

** Es necesario un conocimiento detallado del hardware involucrado y del manejo de interrupciones del procesador.-

** Las soluciones son difíciles de generalizar para diferente tipo de sistemas de tiempo real.-

** La especificación formal y mucho más la validación de esta capa de software constituye un tema aún no resuelto en forma general (Ref. 9,10,11,12).-

Asimismo la utilización de hardware "PC like", que simplifica notablemente el costo de desarrollo de una parte importante del equipamiento, pero sobre todo el costo de desarrollo del software "de procesamiento", se encuentra limitada por la capa de más bajo nivel (BIOS), que fue desarrollada "a medida" para un subconjunto especial y restringido de periféricos, y que no es muy fácil de extender ni mucho menos de portar a otro procesador central.-

Por otra parte el BIOS en la mayoría de los sistemas de microcómputo de propósito general ha sido desarrollado en assembler del microprocesador involucrado, con criterios de optimización de memoria y tiempo de ejecución, que no hacen muy fácil su modificación o re-programación (Ref.13,14).-

Nuestro trabajo consistió en el análisis sistemático de un BIOS estándar y en el desarrollo en lenguaje C de un modelo de BIOS fácilmente extendible para el control de periféricos "no convencionales" (por ejemplo un conversor analógico-digital) o para disponer de nuevas funciones (por ejemplo criptografía de información) con alta eficiencia y mayor portabilidad.-

ANALISIS DEL BIOS DE UNA PC CONVENCIONAL:

El BIOS es un conjunto de rutinas destinadas a controlar el hardware de una microcomputadora tipo PC.-

Estas rutinas residen en ROM, en la parte alta del mapa de memoria y constituyen la capa de software más relacionada con la implementación física de la arquitectura de la microcomputadora involucrada.-

De este modo los programas de mayor nivel (incluyendo al mismo sistema operativo) "ven" una máquina "abstracta" relativamente independiente de su implementación física, brindando una mayor flexibilidad e independencia y asegurando la portabilidad de estos programas a modelos de arquitectura "compatible" que no han sido implementados con idéntico hardware y/o BIOS.-

Un rápido análisis de las rutinas del BIOS nos permite apreciar:

** Atención de RESET.-
** Atención de NMI (interrupción no-enmascarable).-
** Operación de carga del Sistema Operativo mínimo.-
** Atención de Servicios (video, teclado, diskettes, impresora, puerta serie).-
** Manejo del reloj de tiempo real.-
** Atención de otras interrupciones.-

La figura 1.A nos muestra una máquina de estados del funcionamiento del BIOS en un equipo convencional y la figura 1.B expresa simbólicamente el "programa" que ejecuta este conjunto de rutinas. En la figura 2 podemos ver el diagrama de ejecución del servicio de manejo de diskettes.

Figura 1.A: Máquina de estados del BIOS.

Estados:

A: Inicialización y testeo.
B: Espera de interrupciones de la máquina BIOS.
C: Atención de una interrupción de hardware.
D: Atención de una interrupción de software.

Condiciones:

A: Conexión de la alimentación.
B: Invocación de interrupción.
C: Invocación de una interrupción de software.
D: Desconexión de la alimentación.

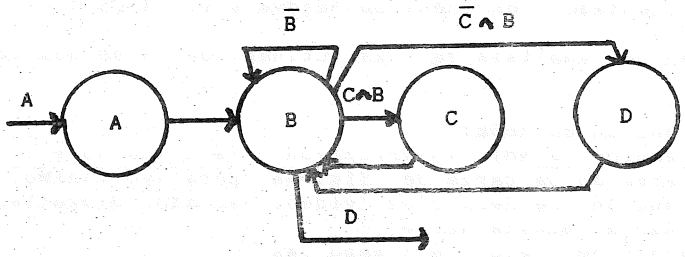


Figura 1.B: Pseudocódigo.

```
repeat
     repeat
     until  Invocación_de_Interrupción_de_la_Máquina_BIOS;
     case Interrupción of
          2           : NMI;
          5           : Prt_Scrn;
          8           : Timer_Int;
          9           : Kb_Int;
          A,B,C,D,F   : D_EOI;
          E           : Disk_Int
          10h         : Video_IO;
          11h         : Equipment;
          12h         : Memory_Size_Det;
          13h         : Diskette_IO;
          14h         : RS232_IO;
          15h         : Cassette_IO;
          16h         : Keyboard_IO;
          17h         : Printer_IO
          18h         : ROM_Basic;
          19h         : Boot_Strap;
          1Ah         : Time_of_Day;
          1Bh,1Ch     : Dummy_Return
     end;
until FALSE;
```

Figura 2: Máquina de estados del controlador de diskettes.

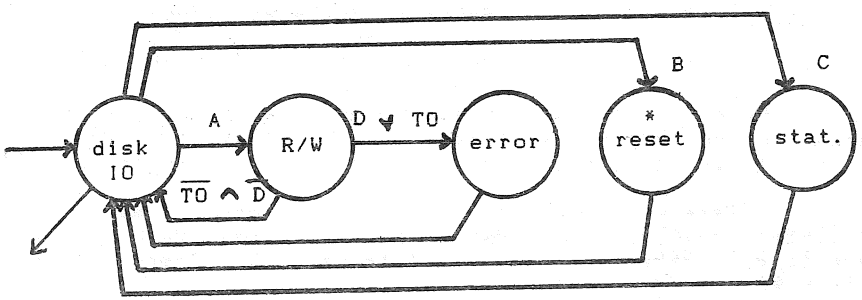
Estados:

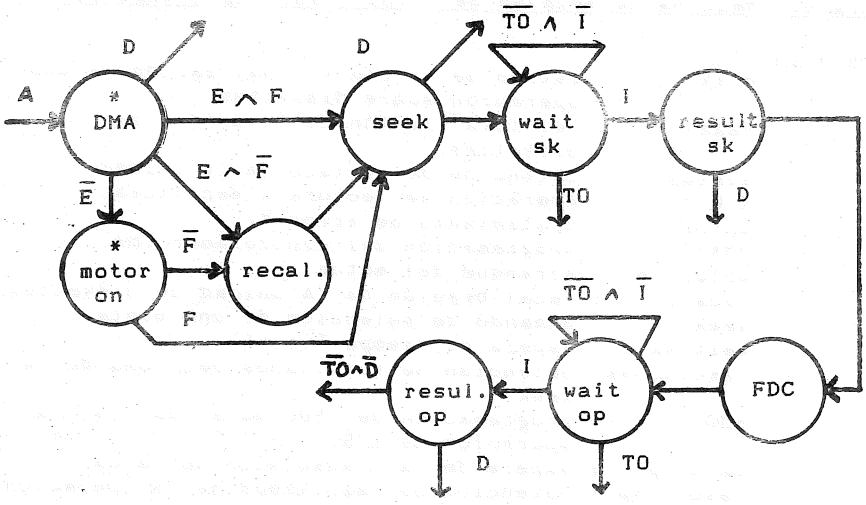
disk_IO	: estado de inicio y finalización de una operación sobre diskettes.
reset	: reinicialización de la unidad de diskettes.
status	: obtención del estado de la unidad.
R/W	: operación de lectura o escritura.
error	: tratamiento de errores.
DMA	: programación del controlador DMA.
motor on	: arranque del motor.
recal.	: recalibración de la unidad de diskettes.
seek	: comando de selección de una pista.
wait sk	: espera de tiempo 'seek'.
result. sk	: obtención de resultados del comando de seek.
FDC	: programación del FDC para realizar la operación de E/S.
wait op	: espera de la transmisión de datos.
resul. op	: obtención de resultados de la operación.

* Indica estado no interrumpible.

Condiciones:

A : solicitud de operación de lectura o escritura.
B : " " " de reinicialización.
C : " " " de informe del estado.
D : error en alguna de las operaciones que conforman la lectura o escritura.
E : motor en marcha.
F : recalibración no requerida.
TO: error producido por exceso de espera.
I : interrupción producida por el controlador de diskettes.





Las rutinas de servicio que forman parte del BIOS "normal" son:

Int 5h	-----	Servicio de vuelco de la pantalla a la impresora.
Int 10h	-----	Servicio de video.
Int 11h	-----	Servicio de listado de configuración.
Int 12h	-----	Servicio de tamaño de la memoria.
Int 13h	-----	Servicio de diskette.
Int 14h	-----	Servicio de comunicaciones.
Int 15h	-----	Servicio de cassette.
Int 16h	-----	Servicio de teclado
Int 17h	-----	Servicio de impresora.
Int 18h	-----	Servicio de activación del ROM-BASIC.
Int 19h	-----	Servicio de carga y puesta en marcha del sistema.
Int 1Ah	-----	Servicio de hora y fecha.

Para utilizar cualquiera de estos servicios, es necesario cargar los parámetros requeridos en los registros del microprocesador y luego ejecutar la interrupción de software deseada.-

Existe además un grupo de rutinas destinadas a controlar las interrupciones de hardware y previsiones para rutinas del usuario que deban ejecutarse a intervalos de tiempo determinados y la atención de 'break'.-

La implementación de estas rutinas, realizada en Assembler 8Ø88 o similar, es altamente eficiente tanto en código como en velocidad de procesamiento. Sin embargo la realización de cualquier modificación o extensión funcional de este conjunto de rutinas exige un profundo conocimiento de la arquitectura de la microcomputadora y del lenguaje Assembler del microprocesador correspondiente.-

CONCEPCION DEL LIDI-BIOS:

Del análisis del BIOS se pueden abstraer sus funciones de modo de obtener una estructura jerárquica como la que muestra la figura 3.-

Como se ve, tenemos tres grupos bien diferenciados que podemos describir como sigue:

** Las rutinas de inicialización del sistema son invocadas cuando se enciende la máquina y consisten en testeos de verificación del procesador y de memoria, inicialización de los circuitos integrados que lo requieran, inicialización de la tabla de vectores de interrupción, verificación de conexión de opcionales a la configuración básica y, si hay controlador de disco, carga del sistema operativo desde éste.

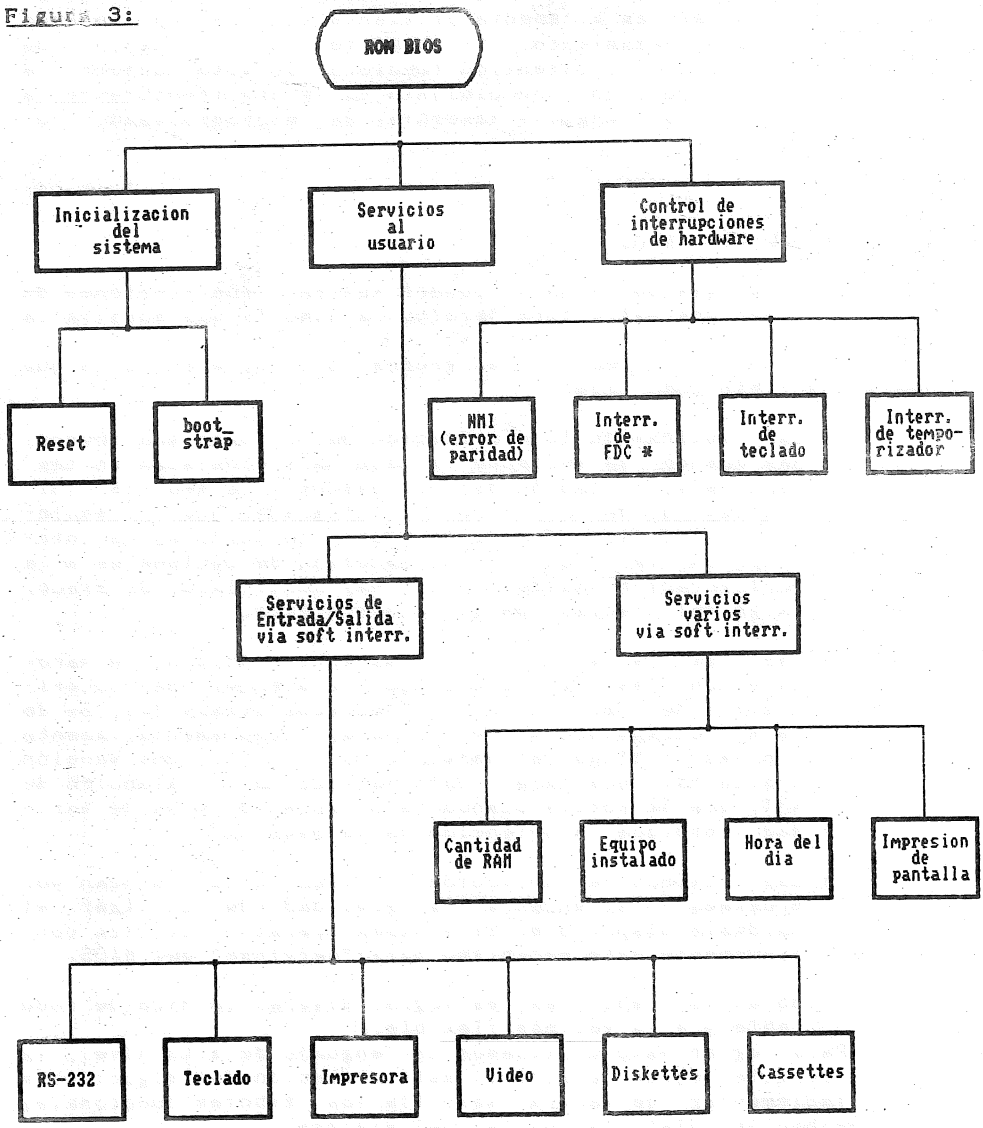
** Las rutinas de control de las interrupciones de hardware son las que se encargan de atender los requerimientos de comunicación con los subsistemas físicos de la microcomputadora y trabajan complementariamente con las rutinas de servicios al usuario. Una mención aparte hay que hacer para la rutina de atención de NMI, que se activa cuando se producen errores de paridad y provoca la detención de la máquina.

** Las rutinas de servicios al usuario se invocan por programa y atienden a la necesidad de utilizar el hardware disponible. El sistema operativo utiliza continuamente este mecanismo para "servirse" del BIOS.-

El objetivo, entonces, es lograr diseñar un BIOS de modo que sea portable y a la vez más flexible.

Para lograr esto utilizamos un lenguaje de alto nivel, ya que nos permite lograr una fácil estructuración del algoritmo e incluso implementar de manera sencilla los árboles funcionales que describen las distintas partes del sistema.

Figura 3:



* **FDC** : **Floppy Disk Controller**

El lenguaje de programación C nos permite establecer una relación casi directa con el hardware, ya que nos provee las siguientes herramientas:

** Aritmética de direcciones (punteros)

** Manejo total de la memoria
En este sentido existe la posibilidad de solapar información en una misma posición de memoria, definición de estructuras a nivel de bit y otros mecanismos que son parte del lenguaje y que otorgan gran flexibilidad en las operaciones sobre memoria.-

** Control de periféricos
Para esto contamos con operaciones que nos permiten programarlos por intermedio de sus puertas de datos y control de entrada/salida.-

** Control de interrupciones
Se pueden enmascarar y desenmascarar interrupciones, de modo de asegurar la exclusión mutua de los procesos en ejecución.-

** Control de llamadas a procedimientos de biblioteca
Podemos controlar la inclusión de procedimientos de bibliotecas, ya que éstas pueden involucrar invocaciones al sistema operativo, perdiendo además el control sobre el tamaño del código generado.

Contando con un lenguaje que posee estas características, vemos que es posible escribir un BIOS en un lenguaje de alto nivel de manera sencilla, legible y que al mismo tiempo sea razonablemente eficiente.-

Acerca de las rutinas interrupt:

Cuando fue descripta la estructura del BIOS, se vio que éste está compuesto por un conjunto de módulos de manejo de interrupción; algunos nuevos compiladores de C proveen el modificador de función 'interrupt' que le da al programa las características necesarias para el manejo de interrupciones.

En efecto estas rutinas realizan el salvado del estado de la máquina (apilado de los registros del micro) y el retorno se realiza por medio de la instrucción IRET, la que, a diferencia del retorno normal, tiene en cuenta el desapilado de las banderas de estado, que son automáticamente apiladas al producirse una interrupción (sea de software o de hardware). Además se permite el acceso a los registros salvados sobre la pila, para obtener su contenido o para modificarlos; esto es muy importante para el planteo de nuestro problema, dado que en las rutinas de servicios del BIOS el intercambio de datos entre éstas y los programas lla-

mayores se realiza por intermedio de los registros del microprocesador.

Finalmente, podemos decir que el LIDI-BIOS, así concebido, cumple con las condiciones que inicialmente habíamos planteado acerca de la portabilidad y flexibilidad, en el sentido de que con cambios mínimos es posible utilizar diferentes implementaciones del lenguaje y adaptarlo a distintos requerimientos de hardware en arquitecturas similares.

LA INSTALACION DEL LIDI-BIOS EN ROM:

Anteriormente fue dicho que el BIOS, al residir en ROM, forma una capa de software transparente al usuario que aísla a los programas de aplicación del hardware. El LIDI-BIOS fue desarrollado teniendo en cuenta la posibilidad de implementarlo en alguna memoria de lectura solamente.-

El problema que surge en forma inmediata al plantear el tema de la generación de código "romable", es la ubicación en memoria de las variables. El BIOS estándar utiliza una pequeña área de RAM de menos de 256 bytes reservada para su uso exclusivo, donde son almacenados ciertos datos que deben mantenerse entre llamadas sucesivas al BIOS, tales como direcciones de puertas de periféricos, parámetros corrientes de video, etc; las "variables" del BIOS son los propios registros del microprocesador.-

Obviamente, este esquema no es aplicable a los lenguajes de alto nivel, dado que en este caso no se posee control directo sobre los registros y las variables siempre ocupan algún área en la memoria, mayor que los 256 bytes antes mencionados.-

En la implementación del LIDI-BIOS también se planteó la existencia de alguna RAM para uso exclusivo, la cual podría estar fuera de los 640 K estándar o "quitándole" memoria al sistema haciendo que el BIOS reporte menos memoria que la real; en el trabajo citado en Ref. 15 puede encontrarse una discusión general del tema y también el detalle de la técnica de implementación utilizada en este caso.-

Las variables utilizadas por LIDI-BIOS fueron definidas como estáticas; la imposibilidad de utilizar variables automáticas está dada por ser éstas definidas sobre el segmento de pila y es muy común que los programas de aplicación y comandos del sistema no provean de suficiente espacio de pila como para, además de salvar el estado de la máquina y realizar el apilado de los parámetros en el llamado a procedimientos, definir las variables locales. De haberse pretendido utilizar variables automáticas se debería haber provisto de un código de inicialización para cada módulo que armase una pila local, éste código debería haberse re-

alizado en assembler lo cual hubiera desvirtuado un objetivo central del trabajo.

El LIDI-BIOS aún no ha sido implementado en ROM. El mecanismo utilizado consiste en reservar un espacio de memoria dividido en dos partes de direcciones conocidas; en una de éstas es cargado, mediante un comando hecho al efecto, el módulo romable "simulando" ser la ROM, la otra parte constituye la RAM dedicada. El sistema se arranca mediante un comando que le pasa el control a la rutina de reset, cuyas principales tareas son mover las constantes a la RAM, dado que serán direccionadas junto con las variables, y reapuntar los vectores de interrupción. Este último proceso es realizado con las interrupciones deshabilitadas y, al ser rehabilitadas, es el LIDI-BIOS el que realiza el control de los periféricos.

EXPERIENCIAS REALIZADAS:

Todas las experiencias se realizaron teniendo en cuenta comparaciones de rendimiento entre las rutinas del BIOS original y el LIDI-BIOS. Se corrieron programas de prueba que ejecutaban llamadas al BIOS y utilizaban el temporizador de la máquina para determinar la cantidad de ticks utilizados.

El testeo de la cuenta se realizó mediante programas escritos en assembler que involucraban únicamente las intrucciones de máquina indispensables para la generación de los lazos y la invocación a las rutinas.

1- Manejo de video

Se realizaron dos experiencias sobre la función de simulación de impresora (función 14, utilizada por el MS-DOS), que consistieron en:

A- Impresión de 25 líneas (una pantalla completa) comenzando en la esquina superior izquierda. No se consume tiempo en scrolling.
B- Impresión de 200 líneas a partir de la última fila (nótese el tiempo demorado en el scrolling).

2- Manejo de disco

Se realizaron dos experiencias:

A- Lectura secuencial de las dos caras de un disco.
B- Lectura de 720 sectores tomados al azar.

3- Manejo de impresora

Se realizaron dos experiencias:

A- Impresión de 7000 caracteres, menos que la capacidad del buffer de la impresora utilizada.
B- Impresión de 32000 caracteres, aproximadamente 4 veces la capacidad del buffer.

	1.A	1.B	2.A	2.B	3.A	3.B
Estándar	35	340	2833	3066	169	4025
LIDI-BIOS	46	563	2694	3049	169	4025
Rendimiento	0.76	0.60	1.05	1.01	1.00	1.00

Los tres dispositivos seleccionados para estas experiencias tienen características dispares que pueden justificar los diferentes resultados obtenidos.

El control de video demanda un uso intensivo del procesador ya que la principal tarea consiste en el manejo de la RAM del adaptador. Este manejo se realiza por medio de punteros 'far' lo cual explica el menor rendimiento del programa en C.

La unidad de discos flexibles tiene piezas móviles con baja velocidad respecto del procesador, además la transferencia de información se realiza mediante DMA, lo cual no involucra uso del procesador. Debido a esto es que el programa en C puede competir con el assembler. Las diferencias a favor son debidas a que los algoritmos involucran lazos de espera optimizados por el compilador. En este punto debe hacerse notar la dificultad de los lenguajes de alto nivel para manejar retardos ya que dependen de cómo el compilador los implemente.

En el caso del control de la impresora no se notan diferencias por tratarse de un dispositivo lento. La sincronización del envío de información es realizada por medio de lazos de espera prolongados donde no se encuentra ninguna ventaja en los programas escritos en assembler.

EXTENSION DE LAS PRIMITIVAS DEL BIOS

Tal como lo hemos expresado a lo largo de este trabajo, uno de los objetivos centrales ha sido la extensión de las prestaciones estándar del BIOS a fin de poder manejar primitivas especialmente adecuadas para sistemas de tiempo real.-

Nuestras primeras experiencias, que a continuación presentamos, se refieren a la incorporación del control de una tarjeta de conversión analógica/digital (A/D) directamente a nivel de BIOS, y a la posibilidad de disponer entre los servicios para el usuario una forma de registro y recuperación de datos en diskettes con codificación y decodificación bajo clave de seguridad.-

1- El manejo de un conversor A/D desde el LIDI-BIOS:

En lo que sigue se ejemplificará con la plaqueta conversora DT28Ø1 (Ref. 16); sin embargo los conceptos son similares para una gama de módulos de E/S.-

La plaqueta DT28Ø1 es un conversor analógico/digital que admite 8 ó 16 entradas (cada una de ellas con tensiones unipolares o bipolares de 1.25, 2.5, 5.Ø ó 1Ø.Ø Voltios) y entrega una salida de 12 bits de resolución.-
La velocidad de respuesta es de 13.7ØØ muestras por segundo (15 microsegundos por muestra para la adquisición y 25 microsegundos por muestra para la conversión propiamente dicha).-

Esta plaqueta se instala directamente en el bus de una PC compatible, ocupando dos bytes en el mapa de direcciones de E/S y pudiendo ser direccionada mediante una combinación de 1Ø bits. Esta dirección es reprogramable por jumpers.-

Los comandos de conversión A/D son 4:

** READ A/D inmediato: realiza una única conversión sobre una entrada y devuelve el valor binario equivalente.-

** SET A/D : permite fijar la ganancia, el o los canales sobre los que se realizarán las conversiones y el número de muestras a tomar. Es un comando previo a todo READ A/D.-

** READ A/D: este comando inicia una secuencia sincrónica de conversiones en los canales especificados por un previo SET A/D.-

** READ A/D con DMA: se trata del mismo comando anterior, pero los datos convertidos son transferidos a memoria vía DMA, es decir sin intervención del procesador.-

En una configuración de PC estándar, el manejo de una plaqueta como la DT28Ø1 debe hacerse mediante un programa en BASIC, FORTRAN o ASSEMBLER. Esto significa un importante overhead respecto del tiempo de muestreo propio de la plaqueta, y además restringe los lenguajes "de procesamiento" seleccionables para las aplicaciones de la plaqueta a aquéllos que permiten el direccionamiento absoluto del mapa de I/O. Más aún, si las muestras deben grabarse en diskette, muchas veces el tiempo de proceso excede lo aceptable para la aplicación.-

En nuestro caso se optó por incorporar a los servicios del LIDI-BIOS la atención de una placa "tipo DT28Ø1"; esto significó un reducido bloque en lenguaje C que implementa la activación de los 4 comandos mencionados anteriormente (Ref. 2Ø). La ventaja que inmediatamente se obtiene es que desde cualquier lenguaje de programación que pueda invocar al BIOS se puede manejar este servicio con altísima eficiencia.-

Lógicamente una extensión como la mencionada anteriormente es muy especial y "dedicada"; sin embargo éste era precisamente el objetivo buscado: poder personalizar mediante las funciones del BIOS un hardware estándar a fin de utilizarlo en aplicaciones dedicadas.-

2- Codificación/Decodificación directa:

Es indudable la utilidad e importancia de poder guardar información "protegida" en un ámbito compartido (por ejemplo un soporte magnético accesible por diferentes usuarios).-

Una primer aproximación, muy común a nivel de sistema operativo, es el control del acceso a la información a través de "palabras clave". Sin embargo esta solución es relativamente segura frente a las diferentes técnicas de "lectura física" o de "revisión exhaustiva" del contenido de un soporte magnético.-

Un nivel mayor de seguridad podemos obtenerlo si la información a preservar es "codificada" antes de grabarse en el soporte, por ejemplo agregando redundancia mediante un código lineal o no lineal (Ref. 17) y utilizando como generador del código un polinomio "exclusivo" del usuario.-

La recuperación de la información sólo podrá realizarse conociendo el polinomio "clave" y toda copia de la información no decodificada sería inútil.-

Desde el punto de vista del LIDI-BIOS, se trata simplemente de agregar una rutina de servicio en el árbol de la figura 3. Esta rutina de servicio es muy semejante a de lectura/grabación de diskettes, sólo que recibe como parámetro

adicional los coeficientes del polinomio (en nuestro caso 4 números hexadecimales a fin de manejar un polinomio de grado 15 o menor, que significa 16 bits de redundancia).-
Con este polinomio como dato, en la grabación se simula por software el funcionamiento del registro de desplazamiento que realiza la codificación en este tipo de códigos (Ref. 18, 19) y luego se graba el bloque de información más la redundancia.-
Recíprocamente, en la lectura se simula la función del registro de desplazamiento "decodificador" (divisor) y se recupera el dato original.-

Lógicamente, a nivel de sistema operativo o desde un lenguaje de aplicación puede explotarse esta prestación del LIDI-BIOS en forma mucho más abstracta, por ejemplo manteniendo el modo de trabajo de "palabra clave" y realizando internamente la transformación en el polinomio asociado, o teniendo un archivo con los polinomios útiles de grado 15 y permitiendo al usuario el manejo de una clave numérica directamente asociada con los elementos de este archivo.-

Por otra parte es de hacer notar que la elección de un código polinomial lineal obedece simplemente al objetivo de indicar posibilidades dentro del LIDI-BIOS; tal vez para una aplicación de seguridad en textos convendría un código orientado a carácter y no a bit (Reed-Solomon, por ejemplo).-

CONCLUSIONES:

Se ha presentado el desarrollo de un BIOS en lenguaje C y algunas extensiones para la operación de periféricos dedicados o el tratamiento especial de información en sistemas de tiempo real.-

En el LIDI (Cs. Exactas - UNLP) está disponible el software desarrollado para aquéllos que estén interesados en el mismo.-

La línea de trabajo actual es el desarrollo de un ambiente para la construcción de Sistemas Operativos dedicados a partir de un kernel de primitivas basadas en el LIDI-BIOS.-

BIBLIOGRAFIA

1- IEEE TUTORIAL on MICROPROCESSOR APPLICATIONS.
IEEE Press 1981.-

2- A. O. WILLIMAN, H. J. JELINEK.
"Microprocessor Developments".
Computer IEEE. Jun 1976.-

3- T. IWASAKA, T. KANEKO.
"Control system for elevators".
Hitachi Review. Vol 28. 1979.-

4- A. KOCHHAR, N. BURNS.
"Microprocessors and their manufacturing applications".
E. Arnold Publishers. 1983.-

5- D. LIBES.
"An introduction to microcomputer languages".
Microsystems. Spt/Oct. 1982.-

6- IEEE TUTORIAL ON PROGRAMMING LANGUAGES.
IEEE Press 1982.-

7- M. BUCHNER, I. LEFKOWITZ.
"Distributed computer control for industrial process systems: characteristics, attributes and an experimental facility".
IEEE Control systems magazine, 1982.-

8- D. BERRY.
"Distributed intelligence in process control"
Control Engineering, may 1987.-

9- S. FAULK, D. PARNAS.
"On syncronization in hard real-time systems".
Communications of the ACM Vol 31, Num.5, Mar 1988.-

10- W. QUIRK.
"Verification and validation of real time software"
Springer Verlag 1985.-

11- C. HOARE.
"Communicating sequential processes"
Prentice Hall 1985.-

12- M. WATKINS.
"A technique for testing command and control soft ware"
Communications of the ACM Vol. 25, Num. 4, 1982.-

13- IBM PC XT Technical Reference.
1986.-

14- R. JOURDAIN.
"Programmer's problem solver for the IBM PC XT & AT"
Prentice Hall 1986.-

15- H. FABIAN TORRES, A. DE GIUSTI.
"Código romable en lenguaje C".
Informe Técnico LIDI 88-0601Ø, 1988.-

16- User Manual for DT2801 Series: Single board analog and digital I/O system.-
1984.-

17- SHU LIN, D. COSTELLO.
"Error Control Coding". Prentice Hall 1983.-

18- W. PETERSON, E.J. WELDON.
"Error Correcting codes". MIT Press 1972.-

19- E.R. BERLEKAMP.
"Algebraic coding theory". McGraw Hill 1968.-

2Ø- A. DE GIUSTI, H. FABIAN TORRES.
"Comando de un conversor A/D tipo DT28Ø1 desde el LIDI-BIOS".
Informe Técnico LIDI-88-0601l, 1988.-

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa.
Buenos Aires, Setiembre 1988.

Procesamiento de eventos para sistemas de tiempo real

Wolfgang A. Halang

Bayer AG
Ingenieurbereich Prozessleittechnik
D-5090 Leverkusen, Bayerwerk, Alemania Federal

y

University of Illinois at Urbana-Champaign
Computer Systems Group, Coordinated Science Laboratory
1101 West Springfield Avenue, Urbana, IL 61801, EE.UU.

Resumen

En este articulo se describe una arquitectura extendida de computadores adecuada para mejorar el desempeño de sistemas concatenados que se encuentran en medios de tiempo real. Esta aumenta la confiabilidad y eficiencia explotando las posibilidades para procesamiento en paralelo inherentes a los sistemas de tiempo real. Se muestra que las transmisiónes internas de datos debidas a las operaciónes de conmutación contextual pueden ser eliminadas y la unidad central de procesos liberada de una cantidad considerable de trabajo rutinario proveyendo un modulo separado para funciónes mayores de sistemas operativos de tiempo real; esto es para reconocimiento de señales e interrupciónes, para administración del tiempo y las tareas, la transferencia de tareas entre diferentes estados y la inicialización del sistema. Considerando el modelo "en capas" de los sistemas operativos modernos de tiempo real estas funciónes constituyen el nucleo y la primera capa de un sistema operativo. Se desarolla una teoria matematica de la distribución de las tareas y sus condiciónes, bajo las cuales estas deben ser realizadas. Se describen las funciónes del procesador de eventos detallando un numero de algoritmos ejecutandose como reacciónes a los eventos ocurridos. Sus complejidades son esencialmente proporcionales al numero de tareas en observación. Esta implementación "hardware" de soporte tipico de tiempo real provee una separación fisica clara de las funciónes intrinsecamente independientes de reconocimiento de eventos y procesamiento de las tareas del procesamiento general. A la vez de reducir la carga y minimizar los tiempos de respuesta y reacción en general, la arquitectura hace posible gratizar una cota superior, predefinida, para el tiempo de reacción a eventos externos e internos.

1. Introduction

The purpose of this paper is to describe an extended computer architecture suitable of improving the performance of embedded systems encountered in hard real-time environments. With the latter term industrial, scientific, and military areas of application are meant, which are characterised by strict time conditions, that must not be violated under any circumstances. In contrast to this, commercial systems, e.g. for automatic banking and airline reservations, only need to fulfill soft real-time requirements, i.e. although they are designed with the objective of fast response time, the user may expect the completion of his transactions.

Real-time data processing systems are expected to recognise and to react to occurring events as soon as possible, that means instantaneously in the ideal case. Therefore, the response time is the most important measure for the performance of real-time systems. With presently available hardware a reaction can only be accomplished by interrupting the running task, determining the source of the event, and switching to a response program. Note that the running task is pre-empted although it is most likely independent on the just arriving interrupt. Furthermore, another task will not necessarily be executed before the current one, after the interrupt has been identified and acknowledged. Owing to this inherent independence, the possibility to apply parallel processing is given here. In order to preserve data integrity, in the conventional architecture the operating system and user tasks may prohibit to be interrupted during the execution of critical regions. Hence, there is a considerable delay between the occurrence of an event and its recognition, and an upper bound for it cannot be guaranteed [7]. This situation is further impaired when several events occur at (almost) the same time resulting in the mutual interruption of their service tasks and postponement of the low priority reactions.

In a conventional real-time computer, every interrupt causes a considerable overhead. This is very unproductive, since the context contained in the register sets needs to be saved and later reloaded. The majority of the interrupts is generated by the interval timer, typically 1000 times per second in order to realise system clocks with a one millisecond resolution. The clock interrupt handling routine updates its time and date variables and checks - mostly unsuccessfully - whether any time-scheduled activities have become due. It is clear that thus a considerable amount of a computer's available processing time is wasted. Furthermore, this kind of timing is not accurate, because the hardware timers have a low resolution, and due to the unpredictable operating system overhead that may supersede the timer routines.

In the course of the progressive increase of semiconductor devices' integration density, and the development of microprocessors, the cost relations have overturned; now, data transmission components within a computer system are more expensive than the processing and storage elements.

The advances in semiconductor technology, however, left the classical von Neumann architecture essentially unaffected. Hence, it appears to be necessary to consider structural changes, in order to minimise, or at least to reduce, the internal transmission expenses that have become the determinative factor of the hardware prices by exploiting possibilities for parallel processing inherent to real-time systems. Screening the pertinent literature, it can be concluded, that no attempt has been made yet to utilise hardware and firmware facilities for the implementation of typical hard real-time support features that cannot otherwise be realised. In this paper, therefore, it is shown that internal data transmissions due to context-switching operations can be eliminated and the CPU relieved from a considerable amount of routine work by providing a separate module for the task of event recognition and processing. By the advances in VLSI technology, this concept has now become realisable. Moreover, such a migration of functions is feasible, as was shown in [8-11], since it leaves invariant the execution sequence of a task set imposed by a given synchronisation scheme, and has further advantageous effects with regard to deadline driven task scheduling and virtual memory management [4].

2. Outline of the Concept for an Event Processor

A computer working in batch or time-sharing mode receives jobs for processing which it does not know in advance. They are to be executed as soon as possible. In contrast to this, the tasks of a process control computer are available in and known to the system. It has to continuously check the conditions for the activation of the various tasks, which are dependent on the actual time, external and internal events, and internal states. These task scheduling functions are assigned to a separate, independently working module, because they impose a heavy load on the CPU and because they are logically independent from general task execution. The resulting asymmetrical double processor configuration is depicted in Figure 1.

The execution of tasking operations is generally linked to conditional expressions, called schedules, which are dependent on the time and on binary quantities. The latter we call events. The module allows the usage of very general scheduling conditions, the most complex ones of which are temporal schedules whose initial times depend upon interrupts. In the next section, we shall present a mathematical theory of these schedules and of the conditions, under which they are fulfilled. The schedules are realised as Boolean procedures. It is the purpose of the event processor to observe the events and to evaluate the schedules. Time schedules are implemented as function procedures yielding, for every invocation, the next critical point in time, for which a tasking operation is planned. The time values obtained from all schedules in the system can be ascendingly ordered and the respective minimum is loaded into an "alarm clock".

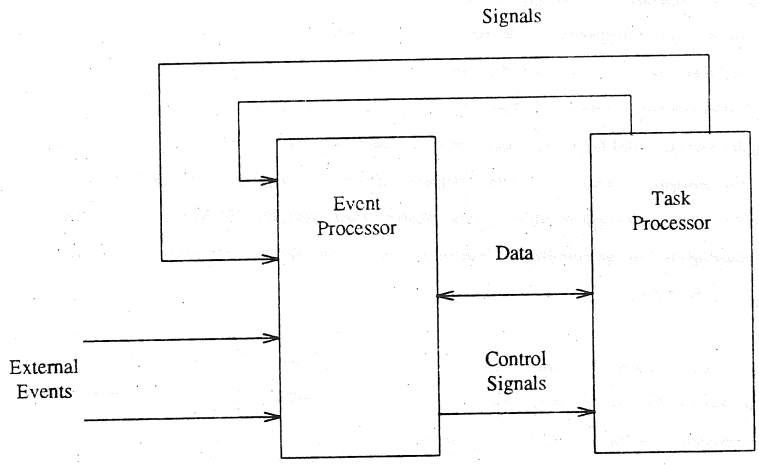


Figure 1

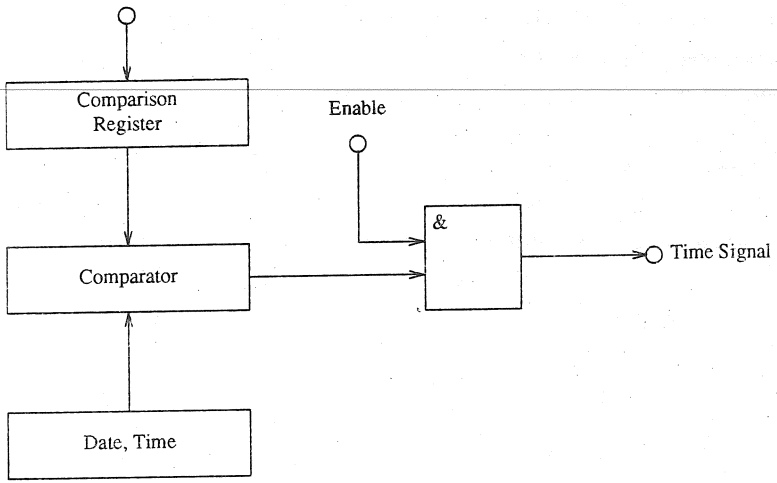


Figure 2

The event processor's time management is supported by an elaborate hardware module, which provides accurate time readings and eliminates all superfluous servicings of the clock. Such a timer was first described in [3] and later again in [12]; for full details the reader is referred to [5]. A schematic diagram with its main components is given in Figure 2. The clock is realised in form of of a divided, individually addressable counter yielding day, hour, minute, second, and an appropriate subdivision thereof. Opposite to the counter stands a correspondingly structured register. The outputs of both units are connected to a comparator generating a time signal upon equality. To keep the number of time events to be processed as low as possible, a signal is only raised when a moment is reached for which a certain action is scheduled.

The servicing of the various events is performed in the following way. In order to guarantee a predefined, short upper bound for the reaction time, the recognition of the events is carried through by continuous cyclic interrogation of the corresponding signal lines. This method is similar to the one employed in Programmable Logic Control (PLC) devices and was already used in the method of synchronous programming for the timely recognition of external events. The cyclic interrogation process has to be carried through with a high frequency, if the recognition time is to be kept short. In the course of the polling process, the event arrival times are saved for future reference. The occurrence of a time signal requires more service than that of other events. So, time schedules need to be handled by checking if they have been exhausted or by calculating the next critical moments. After determining the minimum of all these critical points in time, it is loaded into the clock's comparison register. Upon completion of each interrogation cycle, information is passed over to the operating system residing in the general processor. These data specify the set of schedules actually being fulfilled and the activities associated with them, which are now to be carried through.

Only fast event recognition and closely connected operations are carried out. Thus, the complexity requirements for the event processor are quite moderate. Since it also does not need to be freely programmable, it is realised as a rather simple, fully microprogrammed device, guaranteeing a high speed of operation. In the next but one section, the functions of the event processor will be described by detailing a number of algorithms running as reactions to occurred events. Their complexities are mainly linearly proportional to the number of observed schedules.

There are many possibilities to realise the data exchange between the system components. In order to prevent excessive overhead connected with writer/reader synchronisations, it appears most feasible to send operation parameters via first-in-first-out memories from the general processor to the event processor. Generally, the leading idea for selecting a certain means of data exchange with the event processor must be to achieve high speed combined with simplicity in order not to produce new bottlenecks.

3. Formal Description of the Tasks of the Event Processor

It has been shown in [2] that four states are sufficient to characterise the actual situation of the processing of a task. At any time every task is in exactly one of the states, which are defined as follows:

known: the task is known to the system, its scheduling condition, however, is presently not fulfilled;

ready: the task waits for the allocation of the resources required for its execution;

running: the task is being executed; and

suspended: the execution of the task was interrupted for synchronisation or user specified purposes.

The event processor manages the known tasks according to this model and carries out the transition to the state *ready*, when the corresponding condition are fulfilled. The other states and state transfers are organised by a different unit, viz. the dispatcher, whose description is outside of the scope of this paper. Furthermore, the event processor supervises the schedules of all other tasking operations.

In order to describe the event processor's duties, we consider the time instant t_0, at which the $m(t_0)\geq 0$ tasking operations T_i, i=1,...,m, may be known. The latter are combined into the set T. The elements E_j of the set of events E:={ E_j| j=1,...,n}, whose cardinality $n\geq 0$ may not depend on the time, are conjunctions of the actual events E'_j and possible maskings $M_k^{(j)}$:

$$E_j=E'_j\cdot\prod_{k=1}^{l_j}M_k^{(j)},\ l_j\geq 0,\ j=1,...,n.$$

With F:T→S the set T of tasking operations is surjectively mapped onto the set S:={S_k(t,E)| k=1,...,l(t_0)}, $l(t_0)\geq 0$, of scheduling conditions. Between the events and these run conditions for the tasks there exists the relation

R_1:={(E_j,S_k)| j=1,...,n, k=1,...,l, S_k depends explicitly on E_j}$\subset E\times S$.

Each of the explicitly time dependent schedules in S implies a sequence of instants, at which the corresponding tasking operations may have to be executed. We join all these instants into a countable set Z:={$t_1<t_2<\cdots$}$\subset[t_0,\infty)$ having a strictly monotonous increasing order. There exists a further relation between this set Z and the set of scheduling conditions

R_2:={(t_i,S_k)| i=1,...,|Z|, k=1,...,l, S_k depends explicitly on t and implies t_i}$\subset Z\times S$.

In order to reduce the event processor's processing expense, we assume without loss of generality,

that if it holds $(E_j,S_k)\in R_1$, j∈{1,...,n} and k∈{1,...,l}, than only E_j, however not $\bar{E}_j$, is present in the expression for S_k. This does not represent a restriction, because E_j can be replaced by the event $\bar{E}_j$ or both E_j and $\bar{E}_j$ can be elements of E, respectively. Hence, the Boolean function S_k, k∈{1,...,l}, can change its value from 0 to L at the instant $t\in[t_0,\infty)$ if and only if

1. $(t,S_k)\in R_2$ holds, or
2. at least one E_j, j∈{1,...,n}, with $(E_j,S_k)\in R_1$ has changed its state from 0 to L.

Therefore, it is the responsibility of the event processor to evaluate all scheduling conditions $S_k\in S$, $1\le k\le l$, for which holds

1. $(t,S_k)\in R_2$, if there is a $t_i\in Z$ with $t=t_i$, i∈{1,...,|Z|}, or

2. $(E_j,S_k)\in R_1$, if the event E_j, $1\le j\le n$, has assumed the state L at instant t.

Finally, let

$S^{(t)}:=\{S_k\in S \mid S_k(t-0)=0 \text{ and } S_k(t+0)=L,\ k=1,...,l\}$

and $t\in[t_0,\infty)$, then the tasking operations contained in $\dot{F}^{-1}(S^{(t)})$ must be executed. In order to perform the mentioned processing steps, the task scheduler needs to have the scheduling conditions available in the form of Boolean procedures. If the schedule S_k is explicitly dependent upon time, then additionally a function procedure s_k is required with the property

$$s_k(t)=\begin{cases}\min\{Z_k\cap(t,\infty)\}, & \text{for } Z_k\cap(t,\infty)\neq\emptyset\\ \text{"exhausted indicator"}, & \text{otherwise}\end{cases}$$

where Z_k designates the subset of Z which is implied by S_k. Based on this construction, it is not necessary to have all elements of Z and the entire relation R_2 available at any time. Instead, it is sufficient to evaluate at the instant t_0 all procedures s_k, to join all results into the set Z', to arrange them in increasing order, and to store the following subset of R_2:

$R'_2:=\{(t_i,S_k)\in R_2 \mid t_i\in Z',\ i=1,...,|Z|,\ k=1,...,l\}$

The smallest element of Z' is loaded into the comparison register of the "alarm clock" (cp. Figure 2), which generates a signal when this instant is reached. In contrast to the usage of a relative timer, no program run-times need to be taken into account when this method is applied. Besides the activities

already mentioned above, the event processor has to carry out the following operations, when the time signal arrives at the instant $t_i \in Z$, $i \in \{1,...,|Z|\}$:

1. Remove the smallest element from the set Z': Z':=Z'-{t_i};

2. Evaluate all procedures s_k with $(t_i, S_k) \in R_2$, $1 \leq k \leq l$;

3. Join the instants obtained in step 2 with Z' and sort the resulting set Z';

4. Check if the schedules S_k associated with the exhausted time schedules s_k cannot assume the value L any more; if this is the case, remove all corresponding elements from R_1;

5. Form the up-to-date subset R'_2 of R_2;

6. Load the comparison register of the alarm clock with the minimum of Z'.

4. Task Scheduling Algorithms

After having compiled in the preceding section the tasks of the event processor and having described its essential data structures, we want to detail now the module's function by specifying a number of algorithms. All these routines are tasks themselves, which are executed by the unit and activated by external signals.

Besides the already defined sets, in the following the set of time procedures s:={$s_k(t)$| k=1,...,l} will appear as operand. The mapping zp:S→s is given by the association of a procedure s_k with each schedule $S_k \in S$. The alarm clock's comparison register is designated by *vr*. A set is being sorted in ascending order by submitting it as actual argument to the procedure *sort*. The minimum of a set is provided by the function min. The signal *eof* indicates, whether there is presently no more control information to the read by the event processor. The parameters of tasking operations to be carried through are transmitted to the dispatcher, which has the address *dp*. All other items appearing in the following programs are auxiliary variables; in particular, t stands for the actual time.

A reset signal initialises the event processor and its data structures:

on reset **do**
T:=F:=S:=s:=zp:=Z':=R_1:=R_2:=∅ , enable:=**false** , vr:=∞
od

The event processor commences its activities with loading the alarm clock after occurrence of a start signal:

```
on start do
 if Z' ≠∅ then vr:=min(Z') ; enable:=true
od
```

At any time after a reset signal, the parameters of tasking operations and scheduling conditions can be transmitted to the event processor, which is requested to accept the corresponding data by a load signal. The following program module inserts the parameters read in into the unit's data structures. If new scheduling conditions for already known tasking operations are specified, then the old schedules will be overwritten. The procedure *delete* which erases these conditions will be defined further below.

```
on load do
 while not eof do
  get(TN,SN,sn,EN) ;
  if TN∈T then Σ:=F(TN) ; F:=F-{(TN,Σ)} ;
   if F⁻¹(Σ)=∅ then delete(Σ) fi fi ;
  T:=T∪{TN} , S:=S∪{SN} , s:=s∪{sn} , F:=F∪{(TN,SN)} ,
  zp:=zp∪{(SN,sn)} , R1:=R1∪{(e,SN)| e∈EN} ;
  if sn≠ nil then
   tn:=sn(t) ;
   if tn="exhausted" then
    if not [tz>t=>SN(tz,E)] then delete(SN) fi
   else Z':=Z'∪{tn} , R'2:=R'2∪{(tn,SN)} ; sort(Z') ;
    if tn=min(Z') then vr:=tn fi
   fi
  fi
 od
od
```

After a prevent signal the event processor removes scheduled tasking operations from his lists; hence, it performs the operation inverse to load. If there is no further tasking operation associated with the schedule of the one to be prevented, then also the schedule will be erased by calling the procedure *delete* .

```
on prevent do
 while not eof do
  get(TN) ;
  T:=T-{TN} , SN:=F(TN) ; F:=F-{(TN,SN)} ;
  if F⁻¹(SN)=∅ then delete(SN) fi ;
 od
od
```

The following procedure *delete* removes a scheduling condition and all information related to it from the event processor's data sets and, if need be, loads a new value into the comparison register of the clock.

```
proc(proc bool) delete=(proc bool Σ):
[σ:=zp(Σ) ; S:=S-{Σ} , s:=s-{σ} , zp:=zp-{(Σ,σ)} ,
 for all e∈E with (e,Σ)∈R1 do R1:=R1-{(e,Σ)} od ;
 M:=F⁻¹(Σ) ; T:=T-M ,
 for all m∈M do F:=F-{(m,Σ)} od ;
 if σ≠nil then
  for all tz∈Z' with (tz,Σ)∈R'2 do
   R'2:=R'2-{(tz,Σ)} ;
   if R'2∩{(tz,SN)| SN∈S}=∅ then Z':=Z'-{tz} fi
  od
  vr:=if Z'≠∅ then min(Z') else ∞ fi
 fi]
```

Now, we turn to the actual tasks of the event processor, viz. the evaluation of scheduling conditions when events occur. Upon every 0-to-L-transition of an $e \in E$ the following program is executed:

```
on e=0→L-transition do
 for all SN∈S with (e,SN)∈R1 do check (SN) od
od
```

whereby all schedules are checked which are in relation to the event e. If a scheduling condition assumes the logical value L, the associated tasking operations are transmitted to the dispatcher for execution:

```
proc(proc(real,[1:n]bool)bool) check=
      (proc(real t,[1:n]bool E)bool Σ):
⌈if Σ(t,E) then
 for all TN∈T with (TN,Σ)∈F do put(dp,TN) od
 fi⌋
```

When the time signal of the alarm clock occurs, all schedules are checked as outlined above, which are related by R'_2 to the actual minimum of Z'. The smallest element from Z' and the corresponding tupels of R'_2 are removed. The time schedules associated with the checked task scheduling conditions are evaluated and the set Z' and R'_2 are updated with the resulting instants. Schedules corresponding to exhausted time schedules are deleted from the event processor's lists, if they cannot assume the value L any more. Finally, the clock's comparison register is loaded with the new minimum of Z'.

```
on time_signal do
 tmin:=min(Z') ; Z':=Z'-{tmin} ;
 for all SN∈S with (tmin,SN)∈R'2 do
  check(SN) , tn:=zp(SN)(t) , R'2:=R'2-{(tmin,SN)} ;
  if tn="exhausted" then
   if not [tz>t=>SN(tz,E)] then delete(SN) fi
  else Z':=Z'∪{tn} , R'2:=R'2∪{(tn,SN)}
  fi
 od ;
 sort(Z') ; vr:=if Z'≠∅ then min(Z') else ∞ fi
od
```

On the premises that the number of tasking operations associated with a single schedule - which is normally 1 - is small, the complexity of the program handling the occurrence of an event in E is proportional to the cardinality of S. Let N(|Z'|) operations be necessary to sort Z'. Then the program responding to a time signal requires the execution of O(|s|+N(|Z'|)) instructions. The prevent routine's complexity is linearly dependent on the number of input data, the one of the load routine is proportional to the product of N(|Z'|) and the number of the received data sets.

The signals can be realised as leading edge triggered flip-flops. As already mentioned, the sets S and s consist of procedures, that have to be made available to the event processor. The elements of T contain an identification, parameters for the dispatcher and for the later program execution. The set Z' and the mappings F and zp can be represented by one-dimensional arrays, and the three relations R_1, R'_2 and F^{-1} as inverted files or as two-dimensional bit arrays. In the latter, case a number of the above mentioned statements can be realised by bit chain operations.

5. Evaluation

For the evaluation of the described computer architecture some qualitative considerations are sufficient, since a quantitative evaluation based on analytic modeling was already carried out by Tempelmeier [8,10,11].

First of all, it has to be stressed that the evaluation criteria for real-time embedded systems are quite different from those ones used with respect to batch or time-sharing computer systems. For real-time systems the throughput and the CPU utilisation are less important. What the user expects, is the reliable and predictable fulfillment of his requirements. Also, the cost of a process control computer has to be seen in a larger context. Naturally, the cost of a two-processor system is higher than that of a conventional von Neumann computer. Since the latter cannot guarantee reaction times, it may be unable to cope appropriately with exceptional and emergency situations of the external technical process. In comparison to the costs of a damage of such a process, which is caused by a computer's malfunction, the price for an event processor will be almost negligible.

The problem of prolonged reaction times caused by long phases, during which the operating system disables the recognition of interrupts, was pointed out by Schrott [7]. This measure is usually applied to avoid interrupt cascades while executing elementary operating system functions and to synchronise the access to basic operating system lists. Our architecture solves this problem by distributing the intrinsically independent functions of event recognition and task administration and execution to separate units.

In the early days of real-time data processing the fundamental requirements of timeliness and simultaneousness [6] were realised by the user himself. He employed the method of synchronous programming to schedule, within his application software, the execution of the various tasks. To this end, he usually wrote his own organisation program, a "cyclic executive". Thus, a predictable software behaviour could be realised and the observation of the time conditions could be guaranteed. Later, this method was replaced by the more flexible approach of asynchronous programming, which is based on the concept of the task. Tasks can be activated and run at any time, asynchronously to a basic cycle. The flexibility and conceptual elegance of the method was gained by renouncing predictability and guaranteed time conditions. The event processor has been designed to solve these problems of asynchronous programming.

The most important measure for the performance of real-time systems is the response time. It depends not only on the overall computing speed of a system, but also on the software organisation and here especially of the operating system, since its routines are executed together with the user tasks in an interleaved manner. Thus, the overhead becomes part of the task response times. Also the

interrupt reaction times depend on a system's hardware and software characteristics. As mentioned above, owing to the internal organisation of an operating system and the necessary functions to be performed, e.g. context-switching, there may be a considerable delay of unpredictable length before a conventional system can acknowledge a received interrupt. Both stated problems have been attacked with the here described approach. They could be solved by the observation of the inherent independence between running user tasks or operating system functions on one hand, and operating system routines or external requests on the other. Based on their independence, the mentioned activities were assigned to different devices and can be carried out in parallel, which reduces both task response and interrupt reaction times.

Conclusion

The clear physical separation of the intrinsically independent functions event recognition and processing from general task processing yields a number of improvements as against conventional structures, especially with respect to enhancement of performance. Based on their independence, the mentioned activities can be carried out in parallel, which reduces both task response and interrupt reaction times. By providing a special device for the handling of all events, unnecessary context-switchings are avoided and the normal program flow is only interrupted when required by the scheduling algorithm. Nevertheless, the event servicing tasks will be processed under observation of their due dates, but in a way disturbing the currently active tasks as little as possible. Besides being reduced, the operating system overhead becomes predictable and an upper bound independent of the actual workload for the time required to react upon events can be guaranteed. In general, the transfer of the event processing functions to specialised hardware contributes to enhancing reliability and efficiency essential in real-time applications. Finally, the event processor provides and relies on a more accurate timing facility than conventional process control computers.

To summarise, we have presented here a special module for major functions of real-time operating systems, viz. for interrupt and signal recognition, for the time management, the administration of task schedules, the transfer of tasks between different states, and the initial start-up of the whole system. Considering the layer model of contemporary real-time operating systems [1], these functions constitute the kernel and the first layer of an operating system.

References

[1] Baumann, R. et al., 'Funktionelle Beschreibung von Prozessrechner-Betriebssystemen'. Richtlinie VDI/VDE 3554. Berlin-Cologne: Beuth-Verlag 1982

[2] Eichenauer, B.F., 'Dynamische Prioritaetsvergabe an Tasks in Prozessrechensystemen", PhD Thesis, Universitaet Stuttgart, 1975

[3] Halang, W.A., 'Ansaetze zu funktionsorientierten Prozessrechnerstrukturen', PhD Thesis, Universitaet Dortmund, 1980

[4] Halang, W.A., 'Implications on Suitable Multiprocessor Structures and Virtual Storage Management when Applying a Feasible Scheduling Algorithm in Hard Real-Time Environments', SOFTWARE - Practice and Experience 16 (8), 761 - 769, 1986

[5] Halang, W.A., 'Architectural Support for High-Level Real-Time Languages', Proceedings International Conference on Software Engineering for Real-Time Systems, Cirencester, 28 - 30 September 1987

[6] Lauber, R.J., 'Prozessautomatisierung I'. Berlin-Heidelberg-New York: Springer-Verlag 1976

[7] Schrott, G., 'Ein Zuteilungsmodell fuer Multiprozessor-Echtzeitsysteme', PhD Thesis, Technical University Munich, 1986

[8] Tempelmeier, T., 'Antwortzeitverhalten eines Echtzeit-Rechensystems bei Auslagerung des Betriebssystemkerns auf einen eigenen Prozessor', PhD Thesis, Technical University Munich, 1980

[9] Tempelmeier, T., 'Auslagerung eines Echtzeitbetriebssystems auf einen eigenen Prozessor', Proceedings Fachtagung 'Prozessrechner 1981', pp. 196-205, 1981

[10] Tempelmeier, T., 'Antwortzeitverhalten eines Echtzeit-Rechensystems bei Auslagerung des Betriebssystemkerns auf einen eigenen Prozessor', Teil 2: Messergebnisse, Report TUM-I8201, Technical University Munich, 1982

[11] Tempelmeier, T., 'Operating System Processors in Real-Time Systems - Performance Analysis and Measurement', Computer Performance, Vol. 5, no. 2, pp. 121-127, 1984

[12] Volz, R.A., and Mudge, T.N., 'Instruction Level Mechanisms for Accurate Real-Time Task Scheduling', Proceedings of the 1986 IEEE Real-Time Systems Symposium, New Orleans, 2 - 4 December 1986, pp. 209-215, 1986

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

JOYCE+
MODELO, LENGUAJE Y METODOLOGIA DE PROGRAMACION DE SISTEMAS DISTRIBUIDOS SOBRE REDES DE COMUNICACION

MARIA CONSUELO FRANKY

Grupo de investigación SINBAD
Depto. Ingeniería de Sistemas y Computación
Universidad de los Andes
Apartado aéreo 4976
Bogotá - Colombia

RESUMEN:

Se propone un lenguaje JOYCE+ como una modificación del lenguaje JOYCE definido por Brinch Hansen *[Hansen 87]* . A diferencia de JOYCE que utiliza un esquema de comunicación entre procesos de tipo sincrónico "rendez vous", JOYCE+ propone un esquema de comunicación asincrónico para programar Sistemas Distribuidos que van operar sobre una red de comunicación de máquinas monoproceso y/o multiproceso. El lenguaje permite programar los procesos que componen un Sistema Distribuido de manera concisa, estructurada y con interfaces bien definidas. De hecho la comunicación entre procesos se hace de manera indirecta a través de objetos "puertos" y "canales", lo cual permite programar cada proceso sin tener que conocer nombres de otros procesos y adicionalemente, permite verificar desde compilación el uso correcto de las primitivas de comunicación. En este artículo, además del lenguaje JOYCE+ se propone una metodología para programar Sistemas Distribuidos en la cual se utiliza un esquema único para expresar cualquier clase de proceso en JOYCE+ .

1. INTRODUCCION

JOYCE es un lenguaje propuesto por Brinch Hansen *[Hansen 87]* para describir y programar Sistemas Distribuidos que operan en una sola máquina con capacidad real o simulada de multiproceso. El lenguaje presenta ventajas importantes en relación a otros lenguajes utilizados para el mismo fin, como son CSP *[Hoare 78]* , OCCAM *[Jones 86]* y ADA *[Alford 85]* . Entre las ventajas se pueden citar las siguientes:

- La interfaz de cada proceso está bien definida y el esquema de identificación entre procesos que se comunican es indirecto a través de objetos intermediarios llamados "canales". Esto permite independizar la programación de cada proceso de cualquier nombre utilizado por los demás procesos. Aún los procesos de una misma clase no necesitan distinguirse entre si con nombres o subíndices.

- Los mensajes que puede enviar o recibir un proceso están delimitados por su interfaz, lo cual permite verificar desde compilación el uso correcto de las primitivas de comunicación.

- La activación de procesos es dinámica y eventualmente recursiva, lo cual permite expresar problemas de procesamiento paralelo de manera concisa y sin necesidad de prever de manera estática el número de procesos que se van a activar.

En este artículo proponemos un lenguaje **JOYCE+** como una modificación del lenguaje JOYCE. En el nuevo lenguaje se conservan las ventajas mencionadas pero a diferencia de JOYCE, que utiliza un esquema de comunicación entre procesos de tipo sincrónico "rendez vous", JOYCE+ propone un esquema de comunicación asincrónico para programar Sistemas Distribuidos que van operar sobre una red de comunicación de máquinas monoproceso y/o multiproceso.

En este artículo presentaremos inicialmente (parte 2) el Modelo de Sistema Distribuido JOYCE+ en el cual se basa el lenguaje JOYCE+ . Es necesario tener en cuenta ese modelo cuando se quiere expresar un sistema distribuido en el lenguaje JOYCE+ .

En la parte 3 presentaremos los Elementos del lenguaje JOYCE+, definidos como una extensión de PASCAL con primitivas de comunicación inspiradas en CSP.

En la parte 4 propondremos una Metodología JOYCE+ para programar Sistemas Distribuidos la cual implica ciertas etapas y un esquema único para expresar cualquier clase de proceso componente de un Sistema Distribuido en el lenguaje JOYCE+ .

Finalmente en la parte 5 haremos un recuento de los Proyectos de Software que hemos emprendido con relación al lenguaje JOYCE+.

2. EL MODELO JOYCE+ DE UN SISTEMA DISTRIBUIDO

El lenguaje **JOYCE+** implica un Modelo específico de Sistema Distribuido. A continuación definiremos las clases de objetos que hacen parte del Modelo y sus interrelaciones. La definición se hará sin hacer referencia al lenguaje JOYCE *[Hansen 87]* que fue el que definió originalmente los componentes del Modelo : **JOYCE+** tomó los componentes de JOYCE y los modificó adaptándolos a un ambiente de sistemas distribuidos sobre red de comunicación.

2.1 ESTRUCTURA LOGICA DE UN S.D. EN JOYCE+

Un Sistema Distribuido en el Modelo **JOYCE+** es un conjunto de procesos paralelos llamados **agentes**, que trabajan con algún objetivo común, para lo cual cooperan entre si intercambiando mensajes. Cada agente pertenece a una clase, la cual especifica una secuencia fija de instrucciones que debe ejecutar. Cada agente se ejecuta en un sitio lógico específico de la red sobre la cual opera el Sistema Distribuido. Los agentes no comparten entre si ningún tipo de memoria, aún en el caso de agentes de un mismo sitio lógico. Cada **mensaje** enviado de un agente a otro pertenece a una clase y contiene un dato : para cada clase de mensajes existe un tipo de dato asociado.

La comunicación entre agentes se hace a través de **canales** que son máquinas virtuales que permiten la transmisión unidireccional de un mensaje a la vez. Cada canal es compartido por dos o más agentes pero solo uno de ellos puede asumir el rol de destinatario de los mensajes que se transmitan por el canal : los demás agentes asumen el rol de fuentes de mensajes. Cada canal es de cierto tipo que especifica un conjunto fijo de clases de mensajes que puede transmitir.

Para poder utilizar un canal, un agente debe poseer un **puerto** asociado al canal lo cual constituye una conexión virtual a dicho canal. Como los canales son unidireccionales, los puertos también lo son : un puerto de un agente le permite enviar o recibir mensajes por el canal asociado pero no las dos funciones. Cada puerto es de cierto tipo que especifica el tipo de canal al que se puede conectar. Cuando varios agentes comparten un mismo canal, todos sus puertos conectados al canal deben ser del mismo tipo, que corresponde al tipo del canal.

La estructura lógica de un sistema distribuido en el modelo **JOYCE+** está constituida por el conjunto de agentes, canales y puertos que componen el sistema y sus conexiones. Dicha estructura se puede expresar a través de un grafo en donde cada nodo es un agente junto con sus puertos, y los arcos son las conexiones entre agentes a través de canales. En la figura 1 se ilustra, a manera de ejemplo, el grafo de la estructura lógica de un sistema distribuido.

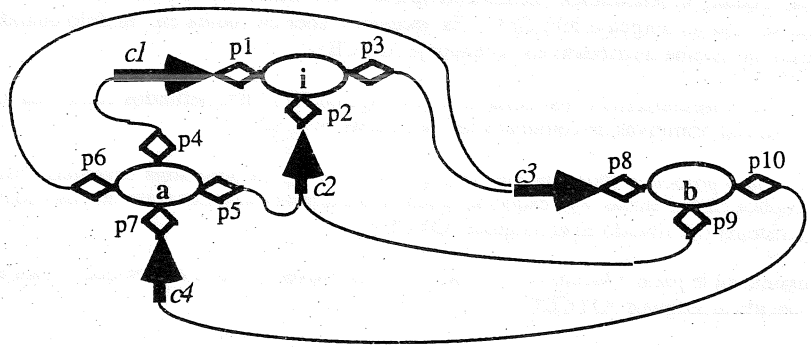


Figura 1 : Grafo de la estructura lógica de un sistema distribuido

Notación : los óvalos representan agentes (para el ejemplo se tienen los agentes **i** , **a** y **b**)
las flechas representan canales ; los rombos representan puertos
las lineas finas representan extensiones de los puertos para conectarse a los canales

En el modelo **JOYCE+** los agentes, los puertos y los canales son creados dinámicamente :

Para iniciar actividad en un sistema distribuido, solo uno de sus agentes (designado el agente inicial) es activado y asignado al sitio lógico 1 (suponiendo una numeración secuencial de los sitios lógicos a partir de 1). Cualquier agente activado puede crear un canal y conectarse a él a través de

uno de sus puertos. También puede activar otro agente asignándolo a cualquier sitio lógico e informándole (en el momento de activación) la referencia a un canal ya creado. Esta información permite al nuevo agente activado conectarse a su vez al canal referenciado a través de uno de sus puertos. Para cada canal compartido entre varios agentes, solo uno de esos agentes es responsable de su creación y los otros reciben en su activación la referencia al canal.

La activación de un agente da lugar entonces a un proceso que debe ejecutar una secuencia de instrucciones (de acuerdo a la clase del agente) en un sitio lógico específico de la red, y que podrá comunicarse con otros agentes ya existentes vía los canales cuya referencias reciba en el momento de activación.

Para el ejemplo del sistema distribuido mostrado en la figura 1, se puede asumir que el agente **i** es el agente inicial, el cual crea los canales *c1, c2* y *c3* y luego activa al agente **a** pasándole referencias a esos canales. El agente **a** a su vez crea el canal *c4* y luego activa al agente **b** pasándole referencias a los canales *c2* , *c3* y *c4* .

En general, la forma como se activan todos los agentes de un sistema distribuido corresponde a un árbol jerárquico. El grafo que representa la estructura lógica de un sistema distribuido es entonces dinámico pues, para cada momento de la vida del sistema, el grafo representa únicamente los agentes activos en ese momento.

La activación recursiva de agentes también es contemplada en el modelo **JOYCE+** : un agente puede activar a otro de su misma clase creándose dos procesos paralelos que ejecutan las mismas instrucciones pero que pueden usar diferentes canales de comunicación .

2.2 TIPO DE COMUNICACION ENTRE AGENTES

Para que un agente **a** pueda enviar un mensaje de clase M a otro agente **b** se deben cumplir los siguientes requisitos :

- El agente **a** debe tener un puerto **pa** conectado a un canal **C** con destino otro puerto **pb** del agente **b** .
- Los puertos **pa** y **pb** deben ser del mismo tipo T correspondiente al tipo del canal.
- La clase M de mensajes debe estar considerada en el tipo T del canal.

Una vez cumplidos los anteriores requisitos, la transmisión de un mensaje del agente **a** hacia el agente **b** se realiza de manera asincrónica :

- El agente **a** ejecuta una primitiva enviar en la cual indica el puerto de salida **pa** a través del cual quiere transmitir y el mensaje que quiere enviar. Esta primitiva no implica el bloqueo del agente **a** hasta obtenerse una confirmación de recepción del mensaje por parte del agente **b** destinatario: únicamente hay bloqueo del agente **a** hasta lograr que el canal **C** esté libre para transmitir el mensaje. Sin embargo, el hecho de que el agente **a** obtenga servicio de transmisión por parte del canal **C** no le asegura que el mensaje llegará realmente a su destino (agente **b**) debido a la posibilidad de fallas de comunicación en la red.

- El canal **C** atiende en orden de llegada las solicitudes de envío de los diferentes agentes que tienen puertos conectados a él. Cuando atiende el envío del agente **a** se encarga de depositar el mensaje en una cola de recepción asociada al puerto de entrada **pb** de **b** . Se asume una red de comunicación en la que la única falla de comunicación visible para los canales es la caída o aislamiento de uno de los sitios lógicos de la red : en ese caso, el canal abandona la solicitud de envío que tiene por destino un sitio lógico aislado o caído (por ejemplo el sitio lógico donde se ejecuta el agente **b**).

- Para que el agente **b** reciba efectivamente el mensaje enviado por el agente **a** , debe ejecutar una primitiva recibir en la cual indica el puerto de entrada **pb** por el cual quiere recibir y la clase M de mensaje que quiere recibir. Esta primitiva bloquea al agente **b** hasta encontrar un mensaje de clase M en la cola de recepción asociada a **pb** .

- En el modelo JOYCE+ no se diferencia la comunicación local entre agentes de un mismo sitio lógico de la comunicación externa entre agentes de sitios lógicos distintos.

3. ELEMENTOS DEL LENGUAJE JOYCE+

El lenguaje **JOYCE+** se define como una extensión del lenguaje *PASCAL* para incluir conceptos de programación concurrente. Aquí presentaremos los elementos nuevos de JOYCE+ respecto a *PASCAL* sin hacer referencia al lenguaje JOYCE de Brinch Hansen *[Hansen 87]* que fue el que introdujo originalmente esos elementos nuevos : **JOYCE+** tomó los elementos de JOYCE y los modificó adaptándolos a un ambiente de sistemas distribuidos sobre red de comunicación.

Los elementos nuevos de JOYCE+ respecto a *PASCAL* son básicamente las declaraciones e instrucciones que permiten construir dinámicamente la estructura lógica de un sistema distribuido de acuerdo al modelo JOYCE+ : declaración de una clase de agentes, declaración de un tipo de puerto, declaración de una variable puerto, activación de un agente, creación de un canal, emisión de un mensaje, recepción de un mensaje.

3.1 ESTRUCTURA DE UN PROGRAMA EN JOYCE+

Un programa en **JOYCE+** tiene la siguiente estructura :

```
program  < nombre del programa >;
  < declaración de constantes globales >
  < declaración de tipos globales >
  < declaración del agente inicial >
end .
```

El programa declara solo un agente inicial que contiene a su vez las declaraciones de las clases de agentes que se quieren activar. Todos los agentes que hacen parte de un sistema distribuido deben quedar declarados en un solo programa. Cuando se inicia la ejecución del sistema distribuido, el agente inicial es activado en el sitio lógico 1.

Debe notarse que no hay declaración de variables globales, reflejando el hecho de que los agentes deben sincronizarse únicamente a través de mensajes y no a través de memoria compartida.

La < declaración de constantes globales > y < declaración de tipos globales > usa la misma sintaxis de *PASCAL* , extendiéndo esta última para considerar un tipo de datos adicional : el tipo puerto.

3.2 DECLARACION DE UN TIPO DE PUERTO

La declaración de tipos de puertos permite enumerar las clases de mensajes que pueden transmitirse a través de los diferentes canales que conectan los agentes de un sistema distribuido. El tipo de un

canal es el alfabeto de clases de mensajes que puede transmitir. Cada puerto de un agente debe pertenecer a un tipo de puerto, el cual indica el tipo de canal al cual puede conectarse.

Cada puerto de un agente corresponde en el programa a una variable de un tipo de puerto. El tipo enumera un alfabeto de clases de mensajes, donde cada clase tiene un nombre y un tipo de dato asociado. A continuación ilustramos la declaración de un tipo de puerto denominado "tpto" el cual contempla 3 clases de mensajes llamadas "número" , "código", y "ack":

```
TYPE   tpto = [número (integer), código (char), ack ( ) ]
```

Como se observa en el ejemplo, algunas clases de mensajes podrían no tener tipo de dato asociado (como "ack") : a estas clases se les denomina *señales* . El tipo de dato asociado a una clase de mensajes es cualquier tipo de *PASCAL*., simple o compuesto (en este último caso, debe declararse previamente).

3.3 DECLARACION DE UNA CLASE DE AGENTES

Cada agente del sistema distribuido descrito por un programa JOYCE+ pertenece a una clase, la cual está asociada a una secuencia específica de instrucciones. La < declaración del agente inicial > también declara una clase de agentes de la cual solo va a activarse una instancia : el agente inicial del sistema distribuido (esta activación se hará de forma automática cuando el sistema distribuido empiece su ejecución).

En general la declaración de una clase de agentes tiene la siguiente estructura :

```
agent <nombre de la clase> (<declaración de parámetros formales>);
  <declaración de constantes>
  <declaración de tipos>
  <declaración de clases de agentes>
  <declaración de variables locales>
  <declaración de procedimientos locales>
begin
  <cuerpo de instrucciones>
end
```

De forma semejante a un procedimiento de *PASCAL* , un agente en JOYCE+ puede declarar constantes, tipos (incluyendo tipos de puertos), variables locales (incluyendo variables de tipo puerto) y procedimientos locales.

Además una clase de agentes *A* puede declarar a otra clase de agentes ***B*** : ésto significa que una vez activado un agente de clase *A* , él puede activar (a través de instrucciones de su < cuerpo de instrucciones >) a uno o varios agentes de clase ***B***.

Cada agente puede recibir en el momento de su activación parámetros por valor (exceptuando al agente inicial): estos parámetros corresponderán generalmente a "apuntadores" a canales ya existentes pero también pueden incluir cualquier parámetro por valor válido en *PASCAL*. La ausencia de parámetros por referencia (en los cuales se pudieran devolver resultados) y de variables globales impide a los agentes compartir memoria y los obliga a sincronizarse únicamente por mensajes que se envían a través de los canales.

En el < cuerpo de instrucciones > asociado a una clase de agentes puede ir cualquier instrucción de *PASCAL* e instrucciones nuevas para crear canales, activar agentes, enviar mensajes y recibir mensajes. Estas instrucciones nuevas se explicarán en las secciones que siguen.

3.4 CREACION DE UN CANAL CON CONEXION A UN PUERTO

Para que un agente pueda crear un nuevo canal del sistema distribuido, debe poseer una variable del tipo puerto asociado al canal. En el momento de la creación del canal, dicha variable "puerto" representará el puerto del agente que va a estar conectado al canal. Teniendo en cuenta que los canales son unidireccionales en JOYCE+, cada puerto de un agente es de entrada (in) o de salida (out) de mensajes.

Una variable puerto de un agente es entonces una variable local que debe ser declarada dentro de su clase de agentes. A continuación se da, como ejemplo, la declaración de una variable puerto de salida llamada **pa** de tipo " tpto " (tipo definido en el ejemplo del numeral 3.2) :

```
VAR  pa : out tpto
```

Una variable puerto es en realidad un apuntador a ese elemento dinámico representado por el canal, el cual es proporcionado por el entorno del sistema distribuido.

Para crear un canal y conectarlo a una variable puerto, un agente debe utilizar la siguiente instrucción :

```
+ <nombre de variable puerto>
```

El efecto de esta instrucción es la creación de un canal que queda apuntado por la variable puerto indicada, y que es del tipo asociado a esta variable. Por ejemplo, **+pa** asigna a la variable puerto **pa** la "dirección" de un nuevo canal de tipo " tpto " .

Múltiples variables puertos de un agente se pueden agrupar en un arreglo si todas ellas son del mismo tipo : es decir, si van a conectarse a canales del mismo tipo en la misma dirección (i.e todos los canales son de entrada o de salida). Por ejemplo, la siguiente es la declaración de un arreglo **arrpa** de puertos de salida de tipo "tpto" :

```
VAR arrpa : ARRAY [1..max] OF out tpto
```

3.5 ACTIVACION DE UN AGENTE

Suponiendo que una clase ***A*** de agentes contiene la declaración de otra clase ***B*** de agentes, la siguiente instrucción (en el <cuerpo de instrucciones> de ***A***) permite a un agente de clase ***A*** activar a un nuevo agente de clase ***B*** para ser ejecutado en un sitio lógico **i** del sistema distribuido :

```
B ( <parámetros actuales> ) at i
```

Se asume aquí que los sitios lógicos del sistema distribuido se numeran con enteros diferentes (por ejemplo 1, 2, 3, ...) y que su asociación a sitios físicos de la red de comunicación se establece de forma dinámica a través de un catálogo externo al programa. Todos los sitios lógicos del sistema distribuido podrían asociarse a un mismo sitio físico permitiendo probar el sistema en una sola máquina antes de montarlo en una red de varias máquinas. En general, el sitio lógico donde se va a activar un agente se especifica dando una expresión de tipo *integer*.

Cualquier agente del sistema distribuido tiene un ciclo de vida consistente en 3 fases sucesivas:

- **fase de activación** : al ser activado el agente crea una instancia del conjunto de parámetros formales declarado en su clase y le asigna los valores de los parámetros actuales. También crea una instancia del conjunto de variables locales declarado en su clase. La unión de estas dos instancias constituye el conjunto de variables propias del nuevo agente (inaccesibles para cualquier otro agente, aún si es de su misma clase).

- **fase de ejecución** : el agente ejecuta el cuerpo de instrucciones asociado a su clase, trabajando sobre sus variables propias. A través de esas instrucciones puede, entre otras acciones, crear canales y activar a otros agentes (que consituyen sus subagentes).

- **fase de terminación** : al terminar sus instrucciones el agente debe esperar a que sus subagentes finalicen y desaparezcan completamente, y luego destruye sus variables propias y los canales que haya creado. Finalmente desaparece del sistema distribuido.

Un agente de clase *A* puede activar varios agentes de clase *B* : cada uno de los subagentes tendrá un conjunto distinto de variables propias y puede o no recibir diferentes valores en los parámetros actuales. Un agente de clase *A* puede activar a otro agente de su misma clase *A* , lo que constituye una activación recursiva (aspecto que da mucho poder al lenguaje).

3.6 CONEXION DE UN PUERTO A UN CANAL YA EXISTENTE

Una vez que un agente **a** ha creado un canal con conexión a un puerto local **pa** puede activar un subagente **b** pasándole como parámetro el valor de su puerto **pa**, es decir un apuntador al canal. Asumiendo que el subagente **b** es de clase *B* y que va a ejecutarse en el sitio i, su activación es de la forma : `B (pa) at i`

El subagente **b** debe recibir el apuntador al canal en un parámetro formal **pb** del mismo tipo de puerto de **pa**, quedando de esta forma conectado al canal a través del puerto local **pb**. Dependiendo de la dirección y agente destino del canal, los dos puertos **pa** y **pb** pueden ser de salida hacia el canal (en cuyo caso debe existir un tercer agente **c** destinatario del canal) , o el uno de entrada y el otro de salida. Asumiendo por ejemplo que el puerto **pb** es de entrada (i.e. el canal tiene como destino el subagente **b**) y que es de tipo "tpto", el encabezado de la clase *B* de agentes debe ser de la forma :

```
B ( pb : in tpto)
```

Con las suposiciones anteriores, la figura 2 ilustra la asociación de los agentes **a** y **b** a través de puertos conectados a un canal común :



Figura 2 : Conexión del agente b a un canal creado por el agente a

La asociación de los agentes **a** y **b** permite que los dos intercambien mensajes a través del canal común usando las primitivas de comunicación que se presentarán en las siguientes secciones.

3.7 ENVIO DE UN MENSAJE

Asumiendo la asociación de los agentes **a** y **b** mostrada en la figura 2, el agente **a** puede enviar un mensaje *m* al agente **b**, siempre y cuando *m* pertenezca a una de las clases enumeradas en " tpto " que es el tipo de canal sobre el cual se va a transmitir.

Por ejemplo, si la declaración global de "tpto" es la siguiente :

```
TYPE tpto = [número(integer), código(char), ack( ) ]
```

entonces el agente **a** puede enviar al agente **b** el mensaje " número (2345 MOD 47) ", y para ello debe ejecutar la siguiente instrucción : **pa** ¡ número(2345 MOD 47) .

La forma general de la instrucción para enviar un mensaje es la siguiente :

```
<puerto de salida> ¡ <clase de mensaje> ( <expresión> )
```

En tiempo de compilación se puede verificar que la instrucción sea correcta chequeando los siguientes aspectos :

- El < puerto de salida > es una variable puerto o un parámetro formal de tipo puerto del agente que ejecuta la instrucción. En cualquier caso su declaración debe especificar que es un puerto de salida (out). Cuando el <puerto de salida> corresponde a un elemento de un arreglo de puertos, se debe especificar así : <nombre de arreglo de puertos> [<subíndice>] , donde el <subíndice> es una expresión de tipo *integer* .
- La <clase de mensaje> debe estar considerada en la declaración del tipo de puerto correspondiente al <puerto de salida>.
- La <expresión> debe ser del tipo de dato asociado a la <clase de mensaje> . Para aquéllos mensajes que son "señales " la <expresión> es nula.

En tiempo de ejecución de esta instrucción se realizan las siguientes acciones :

- Evaluar la <expresión> (si está presente) para calcular el dato que debe ir en el mensaje.
- Conformar el mensaje, el cual está compuesto de <clase de mensaje> y dato acompañante.
- Transmitir el mensaje a través del canal que está conectado al <puerto de salida> (hay una espera implícita mientras se obtiene turno para transmitir por el canal).

Después de estas acciones, el agente que ejecuta la instrucción de envío (agente "emisor") prosigue con la ejecución de la siguiente instrucción sin requerir antes la confirmación de recepción del mensaje por parte del agente destinatario del canal (i.e. comunicación asincrónica). Por su parte, el canal está encargado de depositar el mensaje en orden FIFO en la cola de recepción asociada a un puerto de entrada del agente destinatario (el puerto de entrada depende de la conexión establecida entre los agentes). El tipo de transmisión del mensaje será local si ambos agentes (emisor y destinatario) están asignados al mismo sitio lógico, o externa si están asignados a sitios lógicos distintos; sin embargo, en los dos casos se usan las mismas primitivas de comunicación.

3.8 DIFUSION DE UN MENSAJE

Cuando se tiene un arreglo de puertos de salida se puede difundir un mensaje a través de todos los canales conectados a esos puertos (que por definición son del mismo tipo). La forma general de la instrucción de difusión es la siguiente :

```
<nombre de arreglo de puertos> [<exp1>..<exp2>] ¡
                <clase de mensaje> ( <expresión> )
```

El rango de los puertos utilizados está delimitado por <exp1> y <exp2> que son 2 expresiones de tipo *integer*. La instrucción de difusión equivale a una instrucción FOR de *PASCAL* en la que se ejecuta una instrucción de envío del mensaje por cada uno de los puertos correspondientes al rango especificado.

3.9 RECEPCION SIMPLE DE UN MENSAJE

Asumiendo de nuevo la asociación de los agentes **a** y **b** mostrada en la figura 2 y siguiendo con el ejemplo de la sección 3.7, el agente **b** puede recibir el mensaje " número (2345 MOD 47) " enviado por el agente **a** ejecutando la siguiente instrucción :

```
pb ¿ número ( cont )
```

donde "cont" es una variable local de tipo *integer* , que es el tipo de dato asociado a la clase de mensaje "número". Esta instrucción implica un bloqueo del agente **b** hasta que el mensaje deseado llegue por su puerto de entrada **pb** (i.e. sea depositado en la cola de recepción asociada) y la posterior asignación del dato del mensaje a la variable "cont ".

La forma general de la instrucción para recibir un mensaje es la siguiente :

```
<puerto de entrada> ¿ <clase de mensaje> (<variable>)
```

En tiempo de compilación se puede verificar que la instrucción sea correcta, de manera análoga a la verificación de una instrucción de envío de un mensaje (explicada en la sección 3.7).

En tiempo de ejecución de esta instrucción se realizan las siguientes acciones :

- El agente que ejecuta la instrucción de recepción se bloquea hasta que algún mensaje correspondiente a la <clase de mensaje > llegue a la cola de recepción asociada al <puerto de entrada> . (El bloqueo en reallidad corresponde a un ciclo continuo en el cual el agente está examinando en orden FIFO el contenido de toda la cola de recepción).
- Cuando el mensaje llega y el agente lo detecta, el dato que lleva (si está presente) es asignado a la <variable> .
- El mensaje recibido es eliminado de la cola de recepción asociada al <puerto de entrada> .

3.10 RECEPCION DE MENSAJES DENTRO DE UNA SELECCION (POLLING)

La recepción simple de un mensaje implica el bloqueo del agente que ejecuta la instrucción esperando que le llegue un mensaje de una clase específica. En aquéllos casos en los cuales no se puede asumir el orden en que van a llegar los mensajes a un agente, es más flexible y eficiente que el agente se bloquee esperando no un mensaje específico sino uno entre varios mensajes posibles de diversas clases. La recepción efectiva de un mensaje específico entre varios presentes podría someterse además a la verificación de alguna condición.

Para lograr esta recepción flexible, muy adecuada para agentes que actúan como procesos "servidores", se debe utilizar la instrucción compuesta **POLL** cuya forma general es la siguiente :

```
Poll
  <recepción condicionada1>   -->  <tratamiento1>
/ <recepción condicionada2>   -->  <tratamiento2>
.................
/ <recepción condicionadan>   -->  <tratamienton>
end
```

Cada <recepción condicionada> es una instrucción de recepción simple acompañada de una condición opcional (donde la condición es cualquier expresión booleana válida en *PASCAL*). Cuando lleva condición, la <recepción condicionada> se especifica de la siguiente manera :

```
<puerto de entrada> ¿ <clase de mensaje> (<variable>) & <condición>
```

Con esta instrucción el agente intenta seleccionar y recibir un solo mensaje entre los varios posibles que aparecen en la lista de " recepciones condicionadas ". Cuando escoge y recibe efectivamente un mensaje, el agente debe ejecutar el tratamiento asociado. Cada < tratamiento > es una secuencia cualquiera de instrucciones JOYCE+ .

La ejecución de una instrucción **POLL** se realiza en 2 fases sucesivas :

- fase "polling": Se realiza un examen cíclico de cada <recepción condicionada> empezando por la primera, hasta lograr escoger una. Al examinar una <recepción condicionada> (en la que se especifica una posible condición y la recepción de un mensaje de cierta clase a través de un puerto determinado) ésta se escoge solo si su condición se cumple y si el mensaje asociado está presente en la cola de recepción del puerto. Si hay varios mensajes de la clase deseada presentes en la cola, se escoge el primero en orden FIFO. Cuando la condición no está presente equivale a una condición booleana siempre cierta (i.e. TRUE). El examen cíclico de las recepciones condicionada implica que si no se ha logrado escoger ninguna en el primer ciclo se vuelven a examinar todas empezando de nuevo por la primera, y así sucesivamente en los ciclos que sean necesarios.

- fase " completion " : Una vez escogida una <recepción condicionada>, en esta nueva fase se realizan las acciones asociadas a la recepción seleccionada (i.e. el dato del mensaje se asigna a una variable local y el mensaje se elimina de la cola de recepción del puerto). En seguida se ejecuta el <tratamiento> que figura en la instrucción POLL a la derecha de la <recepción condicionada> escogida.

Para evitar que un agente se quede indefinidamente en la fase "polling" de una instrcción **POLL** (en el caso que no se logre escoger una < recepción-condicionada > en ciclos sucesivos) se puede usar la claúsula opcional ELSE TIMEOUT dentro de la instrucción POLL, de la siguiente forma :

```
Poll
  <recepción condicionada1>  -->  <tratamiento1>
/ <recepción condicionada2>  -->  <tratamiento2>
.................
/ <recepción condicionadan>  -->  <tratamienton>
  else timeout( <máx> )          -->  <tratamienton+1>
end
```

El símbolo <máx> es una expresion de tipo *integer* que denota el máximo número de ciclos que se pueden realizar en la fase " polling ". Si al completar dicho número de ciclos no se ha logrado escoger una <recepción condicionada > , el agente ejecuta el tratamiento alterno <tratamiento$_{n+1}$> y da por terminada la instrucción POLL. La claúsula ELSE TIMEOUT de la instrucción POLL es muy útil en un ambiente de red de comunicación en el cual existe la posibilidad de sitios caídos o aislados : en este caso un agente puede llegar a encontrarse aislado de los demás agentes, y al no poder recibir ningún mensaje podría ejecutar alguna acción de recuperación en el tratamiento alterno. Otro ejemplo en el cual la claúsula ELSE TIMEOUT es útil es cuando un agente **a** quiere verificar que otro agente **b** recibió un mensaje que le envió previamente : al no recibir el mensaje de confirmación de recepción dentro de un período máximo puede asumir que el mensaje inicial no fue recibido (posiblemente por aislamiento del agente **b**).

3.11 ANALISIS DEL LENGUAJE JOYCE+

En comparación a otros lenguajes utilizados para programar sistemas distribuidos como son CSP *[Hoare 78]* , OCCAM *[Jones 86]* y ADA *[Maekawa 87, Alford 85]* , el lenguaje JOYCE+ presentan las siguientes ventajas (válidas en general también para el lenguaje original JOYCE *[Hansen 87]*) :

- Interfaces definidas entre procesos : Cada agente de JOYCE+ (que corresponde al concepto de proceso) tiene una interfaz definida respecto a los demás agentes. Esa interfaz la constituyen el conjunto de puertos de entrada y de salida utilizados por el agente, y que son declarados como parámetros formales o como variables locales. La interfaz de un agente junto con la declaración de los tipos de sus puertos especifica todas las clases de mensajes que puede recibir y enviar. En CSP y OCCAM la interfaz de cada proceso no está definida, y en ADA solo se define la interfaz respecto a los mensajes que pueden llegar al proceso.

- Activación dinamica y recursiva de procesos : En JOYCE+ los agentes se activan de forma dinámica y eventualmente recursiva, por lo cual no hay que prever desde compilación el número total de agentes que existirán en ejecución. La activación recursiva permite expresar de forma concisa y simple soluciones a problemas en los que intervienen múltiples agentes de la misma clase, sin necesidad de recurrir a subíndices para diferenciar los agentes como ocurre en CSP. Ninguno de los lenguajes CSP, OCCAM o ADA permite activación recursiva de procesos.

- Independencia entre procesos : Los agentes en JOYCE+ no se nombran explícitamente cuando intercambian mensajes; solo requieren nombrar sus puertos locales (que deben estar conectados a canales compartidos entre los agentes que quieren intercambiar mensajes). Este esquema de identificación indirecta permite independizar el diseño de cada agente de los nombres específicos de los demás agentes del sistema. Por otra parte, esta independencia entre agentes es fundamental cuando la activación de los agentes es dinámica y no se conoce en tiempo de compilación cúantos y cúales agentes se van a activar en ejecución. En los lenguajes CSP, OCCAM y ADA los procesos deben nombrarse explícitamente cuando ejecutan primitivas para enviar mensajes.

- Verificación de las conexiones entre procesos en compilación : En JOYCE+ hay declaración de clases de mensajes, de tipos de puertos y de puertos, que son los que permiten asociar agentes a través de canales. Estas declaraciones permiten verificar desde compilación que los mensajes intercambiados por agentes que van a estar conectados a un mismo canal pertenecen a un mismo alfabeto, es decir, a un mismo conjunto de clases de mensajes. También se puede chequear desde compilación que ningún agente en ejecución va a tratar de transmitir por un canal inexistente. Estas verificaciones en tiempo de compilación hacen de JOYCE+ un lenguaje seguro respecto a los lenguajes CSP, OCCAM y ADA.

- Sincronización de procesos únicamente por mensajes : Al igual que CSP, JOYCE+ considera la sincronización de procesos únicamente por mensajes y no por memoria compartida (ADA permite ambos tipos de sincronización). Este tipo de sincronización es el más adecuado para un ambiente en el cual los procesos pueden ejecutarse en sitios lógicos distintos y es impuesto en JOYCE+ por la ausencia de variables globales y de parámetros por referencia.

- Comunicación asincrónica, manejo de timeouts y especificación de sitios lógicos : Las ventajas expuestas hasta aquí son válidas tanto para JOYCE+ como para el lenguaje del cual se derivó : JOYCE. Lo que es propio de JOYCE+ es la comunicación asincrónica entre agentes a diferencia de la comunicación sincrónica "rendez-vous" considerada en JOYCE, y también en CSP, OCCAM y ADA. La comunicación asincrónica es más adecuada para sistemas distribuidos que operan sobre redes de comunicación no confiables, y con ella se busca bloquear a los agentes el menor tiempo posible cuando ejecutan las primitivas de comunicación *[Sloman 87]* . Además el manejo de timeout en una selección de primitivas de recepción (lo que corresponde a la claúsula ELSE TIMEOUT de la instrucción POLL) evita el bloqueo de un agente cuando se encuentra aislado de sus agentes interlocutores (también ADA tiene manejo de timeouts pero no lo tienen CSP, OCCAM ni JOYCE). Otro aspecto propio de JOYCE+ es la especificación del sitio lógico en la activación de un agente (aspecto no considerado en CSP, OCCAM, ADA ni JOYCE), lo cual es necesario en sistemas distribuidos de agentes que pueden ejecutarse en distintos sitios lógicos. Por último JOYCE+ incorpora otros aspectos nuevos respecto a JOYCE, como son la declaración de procedimientos locales a un agente (con lo cual se evita la proliferación de agentes en el sistema y se controla el "overhead" en comunicación) y la instrucción de difusión de un mensaje por puertos de salida pertenecientes a un mismo arreglo, la cual puede implementarse de forma eficiente en una red de comuncación local.

4. LA METODOLOGIA DE PROGRAMACION JOYCE+

En muchos sistemas distribuidos en donde la sincronización de los procesos (agentes) es únicamente por mensajes y no por memoria compartida, los procesos son diseñados como servidores de mensajes . Un proceso servidor de mensajes es aquél que después de su inicialización entra en un "ciclo de atención" casi permanente en el cual está capacitado para servir cualquier mensaje correcto que le llegue dándole el tratamiento adecuado, tratamiento que incluye el envío de mensajes de respuesta correspondientes al servicio. El proceso solo puede salir del ciclo de atención cuando se cumpla alguna condición que indique que debe terminar su ejecución.

En el modelo y lenguaje JOYCE+ un proceso servidor corresponde a un agente cuyo cuerpo de instrucciones consiste principalmente en un ciclo dentro del cual hay una sola instrucción POLL especificando recepciones para todas las clases de mensajes correspondientes a los puertos de entrada del agente. Las condiciones del POLL permiten diferenciar el tratamiento de mensajes de la misma clase según el orden relativo en que llegan.

Los Sistemas Transaccionales Distribuidos que operan sobre redes de comunicación utilizan generalmente el enfoque de procesos servidores de mensajes. Otras clases de sistemas distribuidos no lo utilizan pero podrían transformarse en sistemas destribuidos equivalentes de procesos servidores de mensajes.

La metodología JOYCE+ que expondremos en las siguientes secciones indica cómo programar en lenguaje JOYCE+ sistemas distribuidos de procesos servidores de mensajes. La metodología implica expresar primero la estructura lógica del sistema en el modelo JOYCE+, definir luego la interfaz de cada clase de agentes especificando los tipos de puertos involucrados, programar luego el cuerpo de cada clase de agentes especificando el ciclo de atención, y por último programar la conexión dinámica entre agentes a través de canales. Hemos restringido la metodología para el caso en el cual el agente inicial es el único responsable de la conexión dinámica entre agentes (i.e. el agente inicial es el único que crea canales y activa a los demás agentes). En un futuro pensamos ampliar la metodología para el caso general en el cual la conexión dinámica es responsabilidad de todos los agentes y se puede tener además activación recursiva de agentes.

4.1 ETAPA 1: ESTRUCTURA LOGICA DEL SISTEMA DISTRIBUIDO

La Etapa 1 de la medodología de programación JOYCE+ implica definir la estructura lógica del sistema distribuido que se quiere programar y expresarla en el modelo JOYCE+. Teniendo en cuenta que la estructura lógica es dinámica, nos interesa aquí la que correponde al momento en el cual el agente inicial ya ha activado todos los demás agentes del sistema distribuido. Como se explicó en la parte 2 de este artículo, dicha estructura lógica se puede expresar a través de un grafo en donde se muestran como nodos los agentes que componen el sistema distribuido junto con sus puertos locales, y en donde los arcos representan las conexiones de comunicación a través de canales.

Para nombrar los agentes de los diferentes sitios lógicos, en la estructura lógica se puede asumir como identificador único de un agente la concatenación de :

nombre de su clase, sitio lógico, número de instancia

El *número de instancia* es un orden secuencial relativo para los agentes de la misma clase asignados al mismo sitio lógico. Este esquema de identificación único de agentes solo es válido en esta Etapa 1 puesto que en el programa que describirá el sistema distribuido en lenguaje JOYCE+ solo se definen clases de agentes sin nombrar explícitamente instancias de los mismos.

Teniendo en cuenta que en esta medodología de programación JOYCE+ el agente inicial es el único responsable de la creación de canales, activación y conexión de todos los otros agentes, no es necesario representar en la estructura lógica del sistema distribuido al agente inicial : se asume que es un agente adicional a los representados en el grafo con un puerto de salida por cada canal del sistema

La figura 3 muestra a manera de ejemplo el grafo de la estructura lógica de un sistema distribuido :

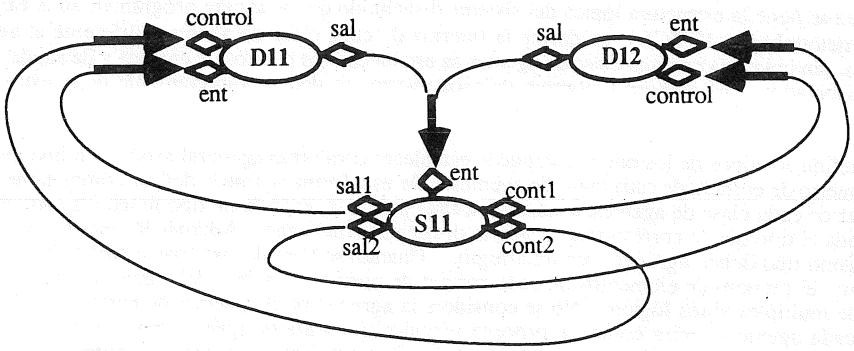


Figura 3. Etapa 1 de la metodología JOYCE+ : estructura lógica del sistema distribuido

El ejemplo de la figura 3 representa un sistema distribuido con dos agentes de clase D ("Despachador") en el sitio lógico 1 y un agente de clase S ("Servidor") en el sitio 2. Suponemos que los agentes D interactúan con los usuarios para ofrecer los servicios del sistema y generan mensajes de peticiones de servicios que envían al agente S. El agente S realiza los servicios accesando una Base de Datos y envía mensajes de respuestas de servicios hacia los agentes D. Para separar las respuestas normales a servicios de los mensajes de control (que indican, por ejemplo, que un servicio no se pudo realizar), cada agente D tiene 2 puertos de entrada. El agente inicial y sus conexiones no están representados en el grafo.

Para el caso de sistemas distribuidos " *homogéneos* " que tienen las mismas clases y número de instancias de agentes en todos los sitios lógicos, es suficiente expresar la estructura lógica para 2 o 3 sitios previendo su generalización a más sitios.

Para construir la estructura lógica de un sistema distribuido se pueden establecer las siguientes reglas generales :

- Cada agente debe tener por lo menos un puerto de entrada con conexión a un canal. Este puerto le servirá para recibir mensajes provenientes de otros agentes, mensajes que corresponden a peticiones de servicios que brinda el agente en cuestión o respuestas a servicios brindados por otros agentes. En el caso de que se quieran discriminar los mensajes de entrada a un agente por prioridad o por razones de claridad, se pueden considerar varios puertos de entrada para ese agente.

- Hay un canal conectado y destinado a cada puerto de entrada de un agente. Por lo tanto el número total de canales en el sistema distribuido es igual al número total de puertos de entrada de los diversos agentes.

- Un agente debe tener tantos puertos de salida como canales (de salida) necesite conectados para enviar mensajes a otros agentes. Estos mensajes corresponden a peticiones de servicios que brindan otros agentes o respuestas a servicios brindados por el agente en cuestión.

- Todos los agentes de la misma clase tienen los mismos puertos de entrada y de salida, y usan los mismos nombres (locales) para esos puertos. Como identificador único de cada agente se asume la concatenación de *nombre de su clase, sitio lógico, número de instancia* .

4.2 ETAPA 2: INTERFAZ DE CADA CLASE DE AGENTES

Una vez se tiene la estructura lógica del sistema distribuido que se quiere programar, en la Etapa 2 de la metodología JOYCE+ se define la interfaz de cada clase de agentes (diferente al agente inicial). En la interfaz de una clase de agentes se establecen los puertos de entrada y de salida y sus tipos asociados. El número y nombre de esos puertos se deduce directamente de la estructura lógica.

Para definir los tipos de los puertos se puede establecer como regla general asociar un tipo único a cada puerto de entrada de cada clase de agentes. De esta forma se puede definir completamente la interfaz de cada clase de agentes, asociando a cada puerto de entrada un tipo único y a cada puerto de salida el tipo que le corresponde al puerto destinatario del canal. Además los puertos de salida del mismo tipo deben agruparse en un arreglo. Cuando se trata de sistemas distribuidos "*homogéneos* " el número de elementos de cada arreglo de puertos de salida debe generalizarse para el caso de múltiples sitios lógicos. No se considera la agrupación de puertos de entrada en arreglos pues cada agente se mira como un proceso servidor que trata de igual forma las peticiones de servicio procedentes de diversos agentes: no requiere entonces un puerto de entrada por cada agente fuente sino uno solo común para todos. Por otra parte, para discriminar las clases de mensajes por prioridades sí se pueden considerar varios puertos de entrada, y en ese caso son de distinto tipo.

La figura 4 ilustra la interfaz de cada clase de agentes del sistema distribuido correspondiente a la estructura lógica que fue mostrada en la figura 3.

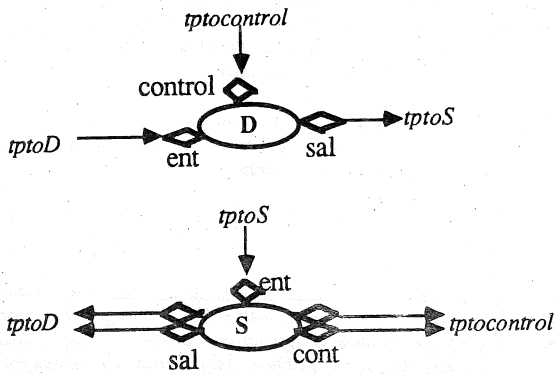


Figura 4 . Etapa 2 de la metodología JOYCE+ : interfaz de cada clase de agentes

Comparando las figuras 3 y 4 se observa que los puertos de salida de la clase de agentes S han sido agrupados en 2 arreglos (sal y cont) de acuerdo al tipo de los puertos.

Al asumir en la metodología de programación JOYCE+ que solo el agente inicial es responsable de la conexión dinámica entre agentes, todos los tipos de puertos deben ser declarados a nivel global del programa que describe un sistema distribuido particular. La declaración misma de cada tipo de

puerto, asociado a un puerto de entrada de una clase de agentes, debe enumerar todas las clases de mensajes que pueden llegar a ese puerto de entrada, las cuales corresponden a la unión de :
- peticiones de servicios que puede brindar un agente de la clase asociada
- respuestas a servicios que otros agentes brindan a un agente de la clase asociada

En la declaración de un tipo de puertos se asocia un tipo de dato a cada clase de mensaje, el cual puede ser cualquier tipo simple o compuesto válido en *PASCAL*. Ese tipo de dato debe resumir los campos (i.e. parámetros) que componen un mensaje de la clase en cuestión. Los tipos de datos se declaran también a nivel global precediendo la declaración de los tipos de puertos.

Cuando se tienen varios puertos de entrada para una misma clase de agentes, los conjuntos de clases de mensajes correspondientes a los tipos de esos puertos no requieren ser disyuntos : por ejemplo, para dar prioridad de tratamiento a un mensaje sobre otro podrían llegar por distintos puertos de entrada aún siendo los dos de la misma clase de mensajes.

Además de los tipos simples de puertos, es necesario declarar tipos arreglos de esos tipos para prever la declaración de los arreglos de puertos de salida que puedan tener las diversas clases de agentes.

Siguiendo con el ejemplo del sistema distribuido correspondiente a las interfaces de clases de agentes mostradas en la figura 4, se podrían tener las siguientes declaraciones globales de constantes, tipos de datos y tipos de puertos (que asocian esos tipos de datos a sus clases de mensajes) :

```
CONST   maxD          = 20;

TYPE tx               = record of
                            cuenta    :  integer;
                            cantidad  :  real;
                            sitio     :  integer
                        end;
     ty               = record of
                            cuenta    :  integer;
                            sitio     :  integer
                        end;

     tptoS            = [ consignar(tx), retirar(tx), saldo(ty) ];
     tptoD            = [ rta_operacion(boolean), rta_saldo(real) ];
     tptocontrol      = [ reactivacion( ) ];
     tarrptoD         = ARRAY [1..maxD] of tptoD;
     tarrptocontrol   = ARRAY [1..maxD] of tptocontrol;
```

Se muestran en estas declaraciones mensajes típicos de una aplicación bancaria : el agente **S** (Servidor) puede recibir peticiones de 3 clases de servicios (considerados en "tptoS") : consignar, retirar o solicitud de saldo. En los 3 casos el agente **S** consulta y/o modifica la Base de Datos de cuentas bancarias y genera un mensaje de respuesta de servicio : en el caso de una consignación o de un retiro la respuesta indica el éxito o fracaso de la operación (mensaje rta_operación) y en el caso de una solicitud de saldo el mensaje rta_saldo informa el saldo requerido. Estos mensaje de respuestas de servicios (considerados en "tptoD") son enviados al agente **D** (Despachador) que hizo la solicitud. El dato asociado a un mensaje que solicita el servicio de consignar o de retirar es un registro de tipo "tx" con 3 campos que indican : la cuenta, la cantidad que se quiere creditar o debitar, y el sitio lógico asignado al agente **D** que hace la solicitud. En el caso de una solicitud de saldo, el dato asociado es un registro de tipo "ty" con 2 campos : cuenta y sitio lógico del agente **D** solicitante.

El mensaje reactivación (considerado en "tptocontrol") es una señal de control enviada por el agente **S** a un agente **D** para indicar que el sitio lógico asignado a **S** está de nuevo en linea después de haber sufrido un aislamiento en términos de comunicación.

En total se declaran 3 tipos simples de puertos, correspondientes a los que aparecen en la interfaces de las clases de agentes (figura 4). Además se declaran 2 tipos arreglos de puertos en base a los anteriores, con el fin de poder declarar posteriormente los arreglos de puertos de salida que aparecen en la interfaz de la clase de agente **S**.

4.3 ETAPA 3 : ESPECIFICACION DE CADA CLASE DE AGENTES

Una vez establecida la interfaz de cada clase de agentes, en la Etapa 3 de la Metodología de programación JOYCE+, se procede a especificar completamente cada clase de agentes diferente al agente inicial.

Teniendo en cuenta que en esta metodología solo el agente inicial activa a los demás agentes, la especificación de una clase de agentes tiene el siguiente esquema general :

```
agent <nombre de la clase> (<declaración de parámetros formales>);
< declaración de constantes >
< declaración de tipos >
< declaración de variables locales >
< declaración de procedimientos locales >
begin
< cuerpo de instrucciones >
end
```

Las diferentes **clases de parámetros formales** (todos por valor) que se pueden declarar en el encabezado de una clase de agentes son los siguientes :

- Puertos de entrada considerados en la interfaz de esa clase de agentes.
- Puertos de salida considerados en la interfaz de esa clase de agentes. Se incluyen los arreglos de puertos de salida definidos en la interfaz.
- Enteros para definir en ejecución el número de elementos de los arreglos de puertos de salida que se van a utilizar (estos enteros pueden depender, por ejemplo, del número total de sitios lógicos).
- El sitio lógico en el que se va a ejecutar un agente específico de esa clase.

Las **declaraciones de constantes, tipos, variables y procedimientos locales** son aquéllas necesarias para poder definir el cuerpo de intrucciones de la clase de agentes. Sin embargo, hay que tener en cuenta las siguientes observaciones :

- La declaración de tipos no incluye nuevos tipos de puertos diferentes a los declarados a nivel global, puesto que se asume que los agentes diferentes al agente inicial no crean canales.

- La declaración de variables debe incluir variables de los diferentes tipos de datos asociados a los puertos de entrada o de salida de la clase de agentes (i.e. tipos de datos asociados a las clases de mensajes enumeradas en los tipos globales de puertos correspondientes).

- Los procedimientos locales pueden incluir instrucciones de envío o difusión de mensajes pero no de recepción : en efecto, al asumir que todo agente es un proceso servidor tiene un único ciclo de atención de recepción de mensajes en su cuerpo de instrucciones, y no en sus procedimientos. Tampoco hay instrucciones de creación de canales ni de activación de agentes, puesto que estas instrucciones son de responsabilidad única del agente inicial.

El **cuerpo de intrucciones** de una clase de agentes tiene el siguiente esquema general :

```
<tratamiento inicial>;
fin :=  false;
while ¬ fin  do
 <instrucción  POLL>
```

En el <tratamiento inicial> están las instrucciones que inicializan las variables locales, y en aquéllas clases de agentes que toman la iniciativa de pedir servicios a otros agentes, se encontrarán aquí instrucciones de envío o difusión de mensajes.

La instrucción WHILE constituye el ciclo de atención de cada agente de la clase especificada : en cada ciclo se ejecuta una instrucción POLL en la cual se describe el tratamiento para cada clase de mensaje recibido por cada puerto de entrada del agente. En la instrucción POLL se pueden enumerar primero las clases de mensajes correspondientes al puerto de entrada al cual se le quiere dar prioridad de atención sobre los demás. Las condiciones sobre las recepciones permiten establecer otros criterios de selección de un mensaje entre varios presentes en la cola asociada a un puerto de entrada.

Como regla general en la instrucción POLL del ciclo de atención de una clase de agentes debe figurar la recepción de todas las clases de mensajes asociadas a los tipos de puertos de entrada que figuran en la interfaz de esa clase de agentes.

En cada uno de los tratamientos de la instrucción POLL del ciclo de atención va una secuencia de instrucciones de los siguientes tipos :

- asignación a variables locales
- invocación de procedimientos locales
- instrucción de control de *PASCAL* como *if* , *while* , etc.
- instrucción para enviar o difundir un mensaje

Gracias a la posibilidad de expresar condiciones sobre las recepciones de la instrucción POLL, no es necesario considerar recepciones simples u otras instrucciones POLL dentro de los tratamientos del ciclo de atención. Tampoco deben figurar instrucciones de creación de canales ni de activación de agentes (que son responsabilidad del agente inicial).

La variable booleana " fin " permite eventualmente terminar la ejecución de un agente cuando asume el valor *false* por efecto de uno de los tratamientos de la instrucción POLL. La claúsula opcional ELSE TIMEOUT de la instrucción POLL permite especificar el tratamiento alterno que debe ejecutar un agente en caso de que no pueda recibir ningún mensaje dentro de cierto período máximo de tiempo (representado por cierto número máximo de ciclos asociados al POLL).

A continuación se muestra un **ejemplo de especificación de la clase de agentes D** (Despachador) correspondiente a las interfaz mostrada en la figura 4 :

```
AGENT D (misitio : integer;
         ent     : in tptoD;
         control : in tptocontrol;
         sal     : out tptoS);

CONST
    max          = 10000;
```

```
VAR
    regx   : tx;
    regy   : ty;
    cta    : integer;
    opcion : char;
    exito  : boolean;
    fin    : boolean;
    monto  : real;

< declaración del procedimiento menu >

PROCEDURE solicitar_servicio;
BEGIN
menu (opcion);
case opcion of
  'C': begin
          read(regx.cuenta, regx.cantidad);
          sal ¡ consignar(regx)
       end;
  'R': begin
          read(regx.cuenta, regx.cantidad);
          sal ¡ retirar(regx)
       end;
  'S': begin
          read(regy.cta);
          sal ¡ saldo(regy)
       end
end
END;

BEGIN (* cuerpo de D *)
fin := false;
regx.sitio := misitio;
regy.sitio := misitio;
solicitar_servicio;
while ¬ fin do
    poll
      control ¿ reactivacion       -->
          writeln('servicio del sistema nuevamente reestablecido');
          solicitar_servicio
    / ent ¿ rta_operacion (exito) -->
          if exito
              then writeln('operación efectuada')
              else writeln ('la operación no se puede realizar ')
          ;
          solicitar_servicio
    / ent ¿ rta_saldo (monto)     -->
          writeln ('el saldo es ', monto);
          solicitar_servicio
      else timeout (max)          -->
          writeln ('no hay servicio por fallas en comunicación')
    end
END;
```

De acuerdo al cuerpo de intrucciones dado en el ejemplo, un agente **D** supone el aislamiento del sitio lógico del agente **S** después de no recibir ningún mensaje en un periodo de 10000 ciclos de su

instrucción POLL. En ese momento informa el hecho a los usuarios y no vuelve a ofrecerles servicios hasta no recibir la señal de control "reactivacion" que indica la superación de la falla.

4.4 ETAPA 4 : CREACION DE CANALES, ACTIVACION Y CONEXION DE AGENTES

En la última etapa de la metodología de programación JOYCE+ se especifica completamente el agente inicial, el cual está encargado de crear todos los canales del sistema y activar todos los agentes conectándolos a los canales. Con esta especificación queda completo el programa en JOYCE+ que describe el sistema distribuido.

La estructura del programa, como se planteó en la parte 3 de este artículo, es la siguiente :

```
program < nombre del programa >;
 < declaración de constantes globales >
 < declaración de tipos globales >
 < declaración del agente inicial >
end .
```

Las declaraciones de constantes y tipos globales (de puertos y de datos asociados a las clases de mensajes) son aquéllas especificadas en la Etapa 2 de la metodología JOYCE+

La declaración de la clase del agente inicial (de la cual solo va a activarse una instancia) tiene el siguiente esquema general :

```
agent  < nombre de la clase inicial >;
  < declaración de clases de agentes >
  < declaración de variables locales >
  < declaración de procedimientos locales >
begin
  < cuerpo de instrucciones >
end
```

La **declaración de clases de agentes** corresponde a las especificaciones hechas en la Etapa 3 de la metodología JOYCE+ .

La **declaración de variables locales** corresponde básicamente a las variables puertos de salida que el agente inicial va a conectar a los canales que va a crear : hay que prever una por cada canal que aparece en la estructura lógica del sistema. También se pueden considerar otras variables locales necesarias para expresar el cuerpo de instrucciones del agente inicial y de sus procedimientos locales.

El **cuerpo de instrucciones** del agente inicial, combinado con las invocaciones a procedimientos locales, debe cumplir con los siguientes 2 objetivos principales :

- Crear todos los canales del sistema distribuido conectándolos a las variables puertos de salida del agente inicial.

- Activar todos los agentes del sistema distribuido conectándolos a los canales de acuerdo a la estructura lógica establecida en la Etapa 1 de la metodología JOYCE+ . Al activar un agente específico, el agente inicial le debe pasar como parámetros efectivos el valor de sus variables puertos conectadas a los canales que el nuevo agente, los enteros necesarios para dimensionar correctamente los arreglos de puertos de salida, y el sitio lógico donde se va a ejecutar el nuevo agente.

Para lograr estos objetivos se utilizan las instrucciones de JOYCE+ de creación de canales y de activación de agentes, las cuales deberán aparecer en el cuerpo de intrucciones y/o en los procedimientos locales del agente inicial.

La asociación de los sitios lógicos a los sitios físicos de la red de comunicación no se establece dentro del programa sino que constituye información del entorno del programa que debe conocerse en el momento de ejecución del sistema distribuido (por ejemplo, a través de un catálogo).

A continuación se presenta la especificación del agente inicial del ejemplo de sistema distribuido presentado en las secciones anteriores (su estructura lógica se presentó en la figura 3 y las interfaces de sus clases de agentes en la figura 4) :

```
AGENT INICIAL;
    AGENT D ( misitio : integer  ;
              ent     : in tptoD;
              control : in tptocontrol;
              sal     : out tptoS );
    ................
    BEGIN
    ...........
    END;

    AGENT S ( misitio : integer;
              numD    : integer;
              ent     : in tptoS  ;
              sal     : out tarrptoD   ;
              cont    : out tarrptocontrol );
    ................
    BEGIN
    ...........
    END;
VAR
    ptoentS      : tptoS;          (* variables de INICIAL *)
    ptoentD      : tarrptoD;
    ptocontrolD : tarrptocontrol;
    numD, i      : integer;
BEGIN (* de INICIAL *)
read (numD);
+ ptoentS;
for i := 1 to numD do begin
    + ptoentD[i];
    + ptocontrolD[i]
end;
S (2, numD, ptoentS, ptoentD, ptocontrolD ) at 2;
for i := 1 to numD do
    D (1, ptoentD[i], ptocontrolD[i], ptoentS ) at 1
;
END.
```

Teniendo en cuenta que el agente inicial tiene puertos de salida hacia todos los canales del sistema, puede utilizarlos para enviar a los demás agentes señales de control. Por ejemplo, para indicar la finalización del sistema distribuido, el agente inicial puede enviar una señal "fin" a cada agente con el objetivo de que finalice su ciclo de atención (lo cual logra cada agente asignando *true* a la variable fin dentro del tratamiento asociado a la recepción de la señal "fin"). De igual manera, el agente inicial puede enviar una señal de "reinicio" a cada agente cuando el sistema distribuido recomienza su actividad después de una interrupción : en este caso, cada agente debe tener previsto

un tratamiento de recuperación asociado a la recepción de esta señal (si es un solo sitio lógico el que recomienza su actividad, la señal "reinicio" debería enviarse solo a los agentes de ese sitio).

5. PROYECTOS RELATIVOS A **JOYCE+**

Actualmente hemos emprendido varios proyectos de software para construir un AMBIENTE DE PROGRAMACION BASADO EN EL MODELO Y LENGUAJE JOYCE+ :

- En el proyecto "**Subsistema de Comunicación para el Ambiente JOYCE+ en DECNET** " *[Torres 88]* se está desarrollando un susbsistema de comunicación basado en el modelo JOYCE+ sobre una red DECNET en la cual están conectados un equipo *Microwax - II* con sistema operacional *UNIX* y otro equipo *Microwax - II* con sistema *VMS* . El susbsistema trabaja sobre 3 catálogos que describen la estructura lógica de un sistema distribuido en el modelo JOYCE+ :

  a) Catálogo de agentes : Indica para cada instancia de agente el nombre de su clase, el sitio lógico de ejecución, el número de instancia (dentro de su clase en ese sitio), el nombre del programa que contiene su secuencia de intrucciones y el identificador del proceso asociado en la red DECNET.

  b) Catálogo de canales : Indica para cada canal del sistema distribuido el agente y el puerto destino.

  c) Catálogo de sitios : Indica para cada sitio lógico el sitio físico asociado y opcionalmente un nombre lógico para la aplicación.

  Con estos catálogos, que deben estar replicados en todos los sitios lógicos del sistema, el Subsistema de Comunicación asegura servicios de comunicación asincrónica entre agentes locales o remotos cuando dichos servicios son solicitados a través de las instrucciones de JOYCE+ para enviar o difundir un mensaje. Los mensajes son colocados en colas de recepción asociadas a los puertos de entrada de los agentes y se asume que la recepción misma es responsabilidad de los agentes.
  En el proyecto " **Traductor JOYCE+ para el Ambiente JOYCE+ en DECNET** " se desarrollará un pseudo-compilador capaz de traducir un programa fuente en lenguaje JOYCE+ en un conjunto de programas en lenguaje C que puedan ejecutarse sobre la red DECNET utilizando los servicios del Subsistema de Comunicación JOYCE+ descrito arriba. El conjunto de programas generados deben lograr los objetivos del sistema distribuido descrito por el programa fuente.

  A grandes rasgos el Traductor JOYCE+ generará un programa en lenguaje C por cada clase de agentes descrita en el programa fuente. Las instrucciones de creación de canales y de activación de agentes se deben traducir en generación de entradas de los catálogos de canales y de agentes. El conjunto de programas generados se deben compilar y estar disponibles en cada uno de los sitios físicos de la red de comunicación (DECNET). Para poner en actividad el sistema distribuido deberá activarse el Subsistema de Comunicación JOYCE+ y ponerse en ejecución el programa correspondiente al agente inicial en cada uno de los sitios lógicos.

  Inicialmente el Traductor JOYCE+ tendrá las mismas restriciones de la Metodología de programación JOYCE+ , es decir, asumirá que solo el agente inicial es responsable de la creación de canales, activación y conexión de los demás agentes. Posteriormente se ampliará en

un nuevo proyecto para permitir que estas funciones sean ejercidas por cualquier agente e incluirá el caso de la activación recursiva de agentes.

- En el proyecto " **Traductor JOYCE+ para el Ambiente JOYCE+ en PC-NET** " se replicarán los mismos objetivos de los 2 proyectos anteriores para una red PC-NET de microcomputadores PC compatibles. En este caso se involucrarán las funciones del Subsistema de Comunicación para Ambiente JOYCE+ (primer proyecto) dentro de los programas generados por el Traductor. De hecho, solo se debe generar un programa para replicarlo en cada microcomputador, el cual debe simular el procesamiento paralelo de los agentes de cada sitio.

- El proyecto " **TESEO : Ambiente de Programación de Sistemas Transaccionales Distribuidos** " *[Franky 87]* tiene como objetivo montar un Sistema Manejador general de Bases de Datos Distribuidas que permita automatizar la programación de sistemas transaccionales distribuidos, asegurando aspectos de atomicidad, recuperación y control de concurrencia de transacciones distribuidas. Nuestro objetivo es dejar una versión definitiva de TESEO sobre la red DECNET de equipos *Vax* (mencionada en el primer proyecto) y para ello planeamos programar TESEO como un sistema distribuido siguiendo el modelo, lenguaje y metodología JOYCE+ . Una vez expresado TESEO a través de un programa en lenguaje JOYCE+ utilizaremos el Traductor JOYCE+ (para Ambiente JOYCE+ en DECNET) para generar los programas definitivos en lenguaje C.

Los anteriores son los proyectos de software relativos a JOYCE+ con los cuales estamos comprometidos a corto, mediano y largo plazo (en nuestro grupo de investigación SINBAD). Todos pertenecen al *área de los Sistemas Distribuidos que operan sobre redes de comunicación.* Sin embargo, existen otras áreas en las cuales se podría explorar la aplicabilidad del modelo y lenguaje JOYCE+, como son las siguientes :

- La programación dirigida por eventos *[Green 86, Hill 86]* típica de las máquinas *Macintosh* y de aquéllas que presentan al usuario interfaces de íconos, ofrece muchas analogías con la programación de un sistema distribuido. En efecto, los dispositivos fuentes de eventos como son el teclado, el ratón, el disco, etc. se pueden considerar agentes que se ejecutan en paralelo y los eventos mismos se pueden considerar mensajes (pertenecientes a clases bien definidas) que son enviados de forma asincrónica al control del programa. Este último, a su vez, se puede ver como un agente "central" que recibe los eventos sacándolos de una cola de recepción, los trata y genera nuevos eventos hacia los otros agentes, quienes finalmente realizan las acciones (tratamientos) que el usuario ve reflejadas en la pantalla. Con todas estas analogías se podría pensar en expresar un programa para este tipo de máquinas en lenguaje JOYCE+ y luego usar un Traductor JOYCE+ específico que generara un programa ejecutable en la máquina. El objetivo de desarrollar tal Traductor sería el de reducir la complejidad de la programación dirigida por eventos ofreciéndole al programador la posibilidad de trabajar sobre el modelo abstracto JOYCE+ .

- En la programación orientada a objetos *[Cox 86]* los objetos que se programan se pueden ver como procesos servidores y podrían expresarse como agentes JOYCE+ . Cada clase de objetos se podría expresar como una clase de agentes con una interfaz bien definida en la cual se tendría un único puerto de entrada para recibir mensajes y varios puertos de salida para enviar mensajes a otros objetos. Los mensajes representarían peticiones de servicios y respuestas a los mismos. El cuerpo mismo de una clase de objetos contendría un ciclo único de atención. Un programa orientado a objetos se puede ver como un sistema distribuido de instancias de objetos que se comunican de forma asincrónica y, en términos generales, en su programación se podrían seguir las Etapas de la Metodología JOYCE+ extendiéndola para considerar la composición de objetos (lo cual corresponde en JOYCE+ a declarar y utilizar clases de objetos dentro de otras clases de objetos). También aquí podría ser útil un Traductor JOYCE+

específico que ofrezca al programador la posibilidad de trabajar sobre el modelo abstracto JOYCE+ esperando así reducir la complejidad de la programación.

- En la programación de sistemas distribuidos para control industrial se podrían utilizar el modelo y el lenguaje JOYCE+ extendiéndolos para manejar tiempo real, lo cual implicaría poder manejar "timeouts" reales, pausas y en general eventos asociados a los relojes de los agentes que componen un sistema distribuido. Se necesitarían entonces traductores JOYCE+ específicos para el tipo de máquinas y de redes de comunicación que se usan en esta área.

- En los problemas de procesamiento paralelo en una sola máquina el lenguaje JOYCE original *[Hansen 87]* presenta grandes ventajas respecto a otros lenguajes generalmente utilizados en esta área. Estas ventajas fueron presentadas en la sección 3.11 de este artículo. En esta área no se justifica utilizar la comunicación asincrónica de JOYCE+ pues en una sola máquina resulta más eficiente la comunicación sincrónica "rendez-vous" de JOYCE, la cual no requiere buffers intermedios. Para utilizar el lenguaje JOYCE en esta área es necesario disponer del traductor (o compilador) asociado : ya existe una versión de compilador JOYCE para un PC el cual mide únicamente 3000 lineas de código fuente y que fue desarrollado por el autor del lenguaje *[Hansen 87b]* .

6. CONCLUSIONES

En este artículo hemos presentado un modelo, un lenguaje y una metodología de programación de sistemas distribuidos que hemos denominado JOYCE+ . Con ellos esperamos facilitar la especificación y programación de sistemas distribuidos que se quieren montar sobre redes de comunicación. El modelo y el lenguaje presentan como principales virtudes la simplicidad, claridad y modularidad, las cuales permiten al programador trabajar a un nivel completamente transparente al ambiente específico de red en el cual va a montar su sistema distribuido. Además proponemos una metodología de programación que indica cómo describir y programar sistemas distribuidos que siguen el enfoque de "procesos servidores".

En primera instancia, el programador tendría que transformar su programa expresado en JOYCE+ a programas ejecutables sobre la red de comunicación específica sobre la cual quiere montar su sistema distribuido. Pero a largo plazo, podremos ofrecer traductores JOYCE+ que eviten al programador tener que conocer cómo se invocan los servicios de comunicación específicos que ofrece la red.

El trabajo que hemos desarrollado alrededor de JOYCE+ tuvo su origen en nuestro conocimiento del lenguaje JOYCE definido por Brinch Hansen *[Hansen 87]* , el cual presenta aspectos muy novedosos respectos a otros lenguajes que son utilizados para programar sistemas distribuidos en una sola máquina.

El desarrollo de ambientes generales basados en JOYCE+ será de gran utilidad para dominar la complejidad que representa la especificación y programación de sistemas distribuidos sobre redes de comunicación. Por otra parte, estos ambientes permitirán construir otros ambientes más sofisticados como son aquéllos relativos a proyectos de Sistemas Manejadores de Bases de Datos Distribuidas (por ejemplo, el proyecto TESEO *[Franky 87]* o el proyecto FCCC en sus etapas futuras *[Abásolo 87]*).

BIBLIOGRAFIA

[Abálsolo 87] : José Abásolo ,
"Protocolos de conversión para el intercambio de información entre Bases de Datos bibliográficas", XIV Conferencia Latinoamericana de Informática (CLEI), Bogotá - Colombia, Nov. 1987.

[Alford 85] : M. W. Alford, J.P. Ansart et al.,
"Distributed Systems", Lecture Notes in Computer Science Nº 190 (capítulo 6), Springer-Verlag, 1987.

[Cox 86] : Brad J. Cox,
"Object oriented programming : an evolutionary approach", Addison Wesley Publishing Co., 1987.

[Franky 87]: M. Consuelo Franky,
"TESEO : Ambiente de programación de sistemas transaccionales distribuidos", XIV Conferencia Latinoamericana de Informática (CLEI), Bogotá - Colombia, Nov. 1987.

[Green 86] : Mark Green,
"A Survey of three Dialogue Models", ACM Transactions on Graphics, Vol. 5 Nº 3, July 1986.

[Hansen 87] : Per Brinch Hansen,
"JOYCE : A programming Language for Distributed Systems", Software - Practice and experience, Vol. 17 Nº 1, Jan. 1987.

[Hansen 87b] : Per Brinch Hansen,
"A JOYCE Implementation", Software - Practice and experience, Vol. 17 Nº 4, April 1987.

[Hill 86] : R. D. Hill,
"Supporting concurrency, communication and synchronization in Human-Computer Interaction - The Sassafras UIMS", ACM Transactions on Graphics, Vol. 5 Nº 3, July 1986.

[Hoare 78] : C.A.R. Hoare,
"Communicating Sequential Processes", Comm. ACM, Vol. 21 Nº 8, Aug. 1978.

[Jones 86] : G. Jones,
"Programming in OCCAM", Prentice-Hall, 1986.

[Maekawa 87] : M. Maekawa, A. Oldehoeft, R. Oldehoeft,
"Operating Systems : advanced concepts" (capítulo 3), The Benjamin/Cummings Publishing Company, 1987.

[Sloman 87] : M. Sloman, J. Kramer,
"Distributed Systems and Computer Networks" (capítulos 3 y 4), Prentice-Hall, 1987.

[Torres 87] : Julio A. Torres,
"Subsistema de Comunicación para un ambiente de procesamiento distribuido basado en JOYCE+ ", tesis de pregrado en Ingeniería de Sistemas ISC-88-I-31, Universidad de los Andes, Sept. 1988.

Capítulo 7

Redes

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa.
Buenos Aires, Setiembre 1988.

ANALISIS DE UN SINCRONIZADOR PARA DATOS CODIFICADOS CONVOLUCIONALMENTE

Oscar N. BRIA, Hugo E. LORENTE y Carlos H. MURAVCHIK

Departamento de Electrotecnia
Facultad de Ingeniería
Universidad Nacional de La Plata

Calle 48 y 116, La Plata BA, (1900) ARGENTINA
Tel. (021) 24-3709, Telex 31151 BULAP ARG

RESUMEN

El propósito de este trabajo es analizar el desempeño de un tipo de sincronizador para datos codificados convolucionalmente.

Se modeliza la máquina de estados del sincronizador como una cadena de Markov. Se define un parámetro de medida del desempeño para cuando se usa el sincronizador en un canal donde se transmiten paquetes, y se encuentra una expresión analítica para el mismo.

Finalmente se presentan resultados numéricos en forma gráfica que muestran la variación del parámetro de desempeño en función del número de estados de la máquina de estados del sincronizador, y de la probabilidad de error al bit en el canal de comunicaciones.

1. INTRODUCCION

En la comunicación por conmutación de paquetes puede ser necesaria una codificación redundante de los datos a transmitir con el propósito de detectar y corregir los posibles errores surgidos durante la transmisión a través de un canal ruidoso. Normalmente se utilizan para la codificación redundante códigos de bloque (Ref. 1). Otra forma de verificar los datos y corregir los errores hace uso de códigos convolucionales. Estos últimos pueden resultar aptos en ciertas aplicaciones donde el canal utilizado es de radiofrecuencia.

Si se recurre a un código convolucional (Ref. 2) los bits de datos y los que servirán para la verificación se envían de forma intercalada. En un código convolucional sistemático de tasa de datos 1/2 se envían en forma alternada un bit de dato y uno calculado por una función convolucional de la información fuente que llamamos de verificación o paridad. Esto obliga a que en el receptor se provean los medios para: a) separar en dos subsecuencias la secuencia recibida, y b) decidir cuál de las subsecuencias corresponde a los datos y cuál a los bits de verificación.

En la Ref. 2 se describe una máquina digital decodificadora (MD), de un código convolucional sistemático de tasa de datos 1/2. MD demultiplexa la secuencia compuesta de bits que recibe,

en dos subsecuencias cada una de ellas formada por una de las dos posibles secuencias de bits alternados . En el procesamiento posterior de verificación y corrección, una de las secuencias es interpretada como secuencia de bits de datos y la otra como secuencia de bits de verificación de la primera.

Cuál de las dos secuencias es interpretada como secuencia de datos y cuál como de verificación depende exclusivamente de la fase de la señal de muestreo de la secuencia compuesta original, señal que llamaremos, como en la Ref.2, reloj de datos. Este reloj de datos se deriva de un reloj llamado de bit que está sincronizado con la secuencia de bits de entrada (y que tiene el doble de frecuencia que el reloj de datos).

Para cada una de las dos posibles fases del reloj de datos, MD puede estar en uno de dos posibles estados. Llamaremos a esos estados enganchado (E) y desenganchado (D). En el primero de ellos se discrimina correctamente cuales son los bits de datos y cuales los de verificación; en el segundo se hace la asignación incorrecta (Fig. 1.1).

En Ref.2 se muestra como se relaciona una variable interna de la MD llamada síndrome con la fase del reloj de datos y la probabilidad de error al bit en el canal de comunicaciones (pe). Esta variable interna tiene la frecuencia del reloj de bit, y se genera en el decodificador para detectar y corregir errores.

El síndrome tiene las siguientes características (Ref.2):

- Es una variable binaria que se actualiza con la llegada de cada bit de la cadena compuesta (es decir su reloj es el de bit).

- Se calcula en forma automática en la misma MD, pues tiene otra aplicación interna en la misma MD, ver la Ref.2.

- El síndrome de cada uno de los bits de datos depende solamente de los errores introducidos por el canal, y en ausencia de estos es cero.

- Para el dispositivo citado el síndrome de cada uno de los bits de verificación depende no sólo de los errores sino también de los bits de datos y/o verificación.

- La probabilidad de que s sea igual a uno durante la recepción de un bit, dado que se calcula el síndrome con los bits de datos es:

Por lo tanto es de esperar, que el comportamiento del síndrome calculado en sincronismo con los datos (síndrome de los bits de datos) sea diferente del calculado en sincronismo con los bits de paridad (síndrome de los bits de paridad). En efecto, en la Ref.2 se demuestra que:

$$Pr(s=1) = 3*pe \qquad (1.1)$$

La probabilidad de que s sea igual a uno durante la recepción de un bit dado que se calcula el síndrome en sincronismo con los bits de paridad es:

$$Pr(s=1) = 1/2 \qquad (1.2)$$

Durante la recepción de un paquete de datos la evolución de la señal de síndrome puede interpretarse como un proceso aleatorio de tiempo discreto (sucesión de variables aleatorias) con espacio de estados (0,1). A este proceso lo llamaremos Sn.

En la Ref.2 se describe una máquina de número de estados finito, diseñada con el propósito de que MD alcance el estado de sincronismo con los bits de datos, que llamaremos enganchado (E) y permanezca en él durante la recepción de un paquete. Aquí definimos una máquina parecida (ver Ref.8), llamada sincronizador de datos (en adelante MQ). Esta máquina tiene como entrada al proceso Sn y como salida la variable binaria que controla la fase del reloj de datos. A ésta última la llamaremos r y cada uno de los dos valores (r1 y r2) que puede tomar corresponde a una de las dos posibles fases del reloj de datos. En la Fig.1.2 se muestra como se relacionan funcionalmente MD y MQ.

En realidad existe una familia de MQs, cada una de ellas con un número de estados distinto. En la fig.1.3 se muestra el diagrama de estados, como la máquina de Moore de una MQ genérica de N estados MQ(N), el cual sirve como descripción de toda la familia. En el diagrama de estados de la Fig.1.4 se ha supuesto que r1 es la fase correcta para el reloj de datos.

De las fórmulas 1.1 y 1.2 y de las figuras 1.2 y 1.3 se deduce, que si la fase del reloj de datos es la correcta, MQ incrementa su cuenta con probabilidad 3*pe, y la decrementa con probabilidad un medio. Como se prohibe que MQ disminuya su cuenta por debajo del estado uno, es de esperar que MQ tienda a permanecer en el estado uno o sus inmediaciones, para el caso usual de que pe<<1/2.

De igual manera se deduce, que si la fase del reloj de datos es la incorrecta, MQ incrementa su cuenta con probabilidad un medio y la decrementa con probabilidad 3*pe. En esta situción es de esperar si pe<<1/2; que MQ aumente en promedio su cuenta. Pero al llegar al estado N/2 (umbral de decisión) se invierte la fase del reloj de datos y se pasa a un estado similar al uno pues se supone que la nueva fase es la correcta.

Expongamos ahora otro aspecto del problema que nos ocupa. Si por ejemplo r2 es la fase correcta para un paquete en particular, no lo será necesariamente para el próximo (o más generalmente para otro cualquiera). Es decir que la MQ debe estimar cuál es la fase a utilizar al comienzo de cada paquete. A tal efecto la información útil del paquete viene precedida por unos cuantos bits de sincronismo, de modo que la MQ pueda estimar en ese lapso la fase del reloj de datos que mantendrá a la MQ en el estado E.

El propósito de este trabajo es estudiar analíticamente el desempeño de las MQs excitadas por el proceso aleatorio Sn durante la recepción de un paquete; suponiendo que los paquetes están compuestos por L+1 bits de información útil (bits de datos y bits de verificación) precedidos por M-1 bits de sincronismo. Se utiliza como herramienta analítica la teoría de las cadenas de Markov.

En la sección 2 se demuestra que se puede describir la evolución de MQ como una cadena de Markov. En la sección 3 se discute un índice que califica el desempeño de las MQ. Los resultados obtenidos para la máquina de la Ref.2 son presentados en la sección 4.

2. CADENAS DE MARKOV DE LAS MQs.

Dadas las características de la familia de máquinas de estado finito MQs, abordaremos el estudio del comportamiento de las MQs excitadas por el proceso Sn. Para ello en ésta sección definimos el proceso aleatorio que describe la evolución de los estados de las MQs y demostramos que se trata de una cadena de Markov.

Definimos un proceso aleatorio de tiempo discreto:

- Xn, n=1,2,....; es un proceso aleatorio (sucesión de variables aleatorias) donde cada Xi es una variable aleatoria cuyos valores pertenecen al conjunto de estados de MQ. Este conjunto es X=(x1,x2,.....,xN).

Veamos como se relaciona el proceso Xn con el hecho de que MD se halle enganchada (fase correctadel reloj de datos) o no:

- Durante toda la recepción de un paquete sólo una de las dos fases será la correcta.
- Al conjunto X=(x1,x2,....,xN) por comodidad lo desdoblamos en (e1,e2,....,eN/2) U (d1,d2,.....,dN/2) = E U D. EE es el conjunto de todos aquellos estados de MQ que tienen como salida la fase correcta, y D es el conjunto de todos aquellos estados de MQ que tienen como salida la fase incorrecta.
- Si suponemos en lo que sigue que r1 es la fase correcta. Entonces cuando Xn tome valores en E la fase será r1. Y cuando Xn tome valores en D la fase será r2, que es la fase incorrecta, y por lo tanto será Zn=D.

Con las definiciones, suposiciones y asignación numérica secuencial hecha a los estados (ver Fig.1.3) podemos ahora escribir:

$$E = (e1,e2,..,eN/2) = (x1,x2,.....,xN/2) \qquad (2.1)$$

$$D = (d1,d2,..,dN/2) = (xN/2+1,xN/2+2,.....xN) \qquad (2.2)$$

E y D se definen así si y sólo si r1 es la mejor respuesta.

En lo que sigue se tomarán las siguientes convenciones:

- El signo " " significa intersección lógica.
- CD = 1 significa que el flanco negativo del reloj de datos ocurre durante un bit de de datos.
- CD = 0 significa que el flanco negativo del reloj de datos ocurre durante un bit de paridad

Recordando las características del proceso aleatorio Sn podemos escribir (ver fórmulas 1.1 y1.2):

$$Pr(Sn=1/Xn \text{ pertnc } E \;.\; CD=1) = p = 3pe \qquad (2.3)$$

$$Pr(Sn=1/Xn \text{ pertnc } D \;.\; CD=1) = 1/2 \qquad (2.4)$$

Como así también:

$$Pr(Sn=1/Xn \text{ pertnc } E \ . \ CD=0) = 1/2 \qquad (2.5)$$

$$Pr(Sn=1/Xn \text{ pertnc } D \ . \ CD=0) = p = 3pe \qquad (2.6)$$

Como lo que interesa es caracterizar el proceso (Xn), deberíamos conocer la distribución conjunta de las Xn o lo que equivalente (Ref.5), para todo n:

$$Pr(Xn+1=xn+1/Xn=xn, Xn-1=xn-1, Xn-2=xn-2, \ldots, X0=x0) \qquad (2.7)$$

con xi pertnent a X.

El estado futuro de una máquina de número de estados finito depende únicamente del estado actual (Xn) y la entrada actual (Sn) (ver fig. 2.1). Como el valor de Sn depende del estado presente de X y CD (2.3-2.6), es fácil demostrar (Ref.8) que:

$$- Pr(Xn+1=xn+1/Xn=xn, Xn-1=xn-1, Xn-2=xn-2, \ldots, X0=x0) = Pr(Xn+1=xn+1/Xn=xn) \qquad (2.8)$$

Un proceso con la propiedad dada por la fórmula 2.8 es un proceso de Markov de tiempo discreto y como tiene un espacio de estados denumerable es una cadena.

Una cadena de Markov tiene la propiedad de que, la probabilidad del comportamiento futuro de la sucesión dado que el estado presente es conocido exactamente, es independiente del conocimiento del comportamiento pasado.

A la probalidad de que Xn+1 sea el estado j, dado que Xn es el estado i o probabilidad de transición de un paso, la denotamos:

$$P_{i,j}^{n,n+1} = Pr(Xn+1=j/Xn=i) \qquad (2.9)$$

Cuando las probabilidades de transición de un paso (2.9) son independientes de n decimos que la cadena de Markov es homogénea. Podemos entonces simplificar la notación:

$$P_{i,j}^{n,n+1} = P_{i,j} \qquad (2.10)$$

Una cadena de Markov homogénea queda completamente definida por:

- La distribución de probabilidades iniciales, que puede ponerse en forma de vector.

- El conjunto de probabilidades de transición de un estado a otro, para todos los posibles pares de estados, que puede arreglarse en forma de matriz. Esta matriz es la matriz de transición de Markov.

El proceso Xn que describe la evolución de los estados de MQ resulta ser una cadena finita de Markov homogénea. A continuación hallaremos la matriz de transición para una MQ genérica.

Si seguimos considerando a r1 como la fase correcta y tenemos presentes las las fórmulas 2.3, 2.4, 2.5 y 2.6, podemos escribir el siguiente conjunto de relaciones (2.11), con p=3pe; m=1/2 y q=1-p:

Si i es par,

Para j=i+1:
-Si i pertnc a (EE)-eN/2, entonces Pr(Xn+1=j/Xn=i)=p
-Si i pertnc a (DD)-dN/2, entonces Pr(Xn+1=j/Xn=i)=m

Para j=i-1:
-Si i pertnc a (EE), entonces Pr(Xn+1=j/Xn=i)=q
-Si i pertnc a (DD), entonces Pr(Xn+1=j/Xn=i)=m

Si i es impar,

Para j=i+1:
-Si i pertnc a (EE)-e1, entonces Pr(Xn+1=j/Xn=i)=m
-Si i pertnc a (DD)-d1, entonces Pr(Xn+1=j/Xn=i)=q

Para j=i-1:
-Si i pertnc a (EE)-e1, entonces Pr(Xn+1=j/Xn=i)=m
-Si i pertnc a (DD)-d1, entonces Pr(Xn+1=j/Xn=i)=p

Además:

Pr(Xn+1=e2/Xn=e1)=1
Pr(Xn+1=d2/Xn=d1)=1
Pr(Xn+1=d2/Xn=eN/2)=p
Pr(Xn+1=e2/Xn=dN/2)=m

Si tenemos presentes las fórmulas 2.1 y 2.2

$$E = (e1,e2,..,eN/2) = (x1,x2,.....,xN/2)$$

$$D = (d1,d2,..,dN/2) = (x(N/2+1),x(N/2+2),.....xN)$$

se puede ver fácilmente que la matriz de Markov correspondiente a una MQ genérica, de N estados, con probabilidad de error de un bit pe, es (2.12):

$$P(N,p)=\begin{array}{c|cccccccccccccccccc}
 & 1 & 2 & 3 & 4 & \cdots & N/2 & \cdots & N \\
1 & 0 & 1 & 0 & 0 & \cdots & \# & \cdots & \\
2 & q & 0 & p & 0 & \cdots & \# & \cdots & \\
3 & 0 & m & 0 & m & \cdots & \# & \cdots & \\
\vdots & & & & & & \# & & \\
 & & & & & & \# & & \\
 & & & & & & \# & & \\
 & & & & & & \# & & \\
 & & & & & m\ 0\ m & \# & 0\ 0\ 0 & \cdots & \\
N/2 & & & & & 0\ q\ 0 & \# & p\ 0\ 0 & \cdots & \\
 & \# & \# & \# & \# & \# & \# & \# & \# & \# & \# & \# & \# & \# & \# & \# & \# & \# & \# \\
 & & & & & 0\ 0\ 0 & \# & 0\ 1\ 0 & \cdots & \\
 & & & & & 0\ 0\ 0 & \# & m\ 0\ m & \cdots & \\
 & & & & & & \# & & \\
 & & & & & & \# & & \\
 & & & & & & \# & & \\
 & & & & & & \# & & \\
 & & & & & & \# & & \cdots & m\ 0\ m\ 0 \\
 & & & & & & \# & & \cdots & 0\ p\ 0\ q \\
N & m & \cdots & & & & \# & & \cdots & 0\ 0\ m\ 0
\end{array}$$

Otra forma de representar las probabilidades de transición consiste en construir el diagrama de estados y transiciones de la cadena. En la figura 2.2 se ve el diagrama de estados y transiciones de la cadena de Markov que representa a MQ(12). Los nodos del diagrama representan los estados y los arcos dirigidos las probabilidades de transición de un estado a otro. Si bien las máquinas de número de estados finito también admiten una representación de diagramas de estados y transiciones, ambas representaciones no deben confundirse; ni tampoco pensarse que una representación se infiere directamente de la otra.

Por simple inspección (Refs.3 y 4) se pueden ver algunas características de las cadenas de Markov de las MQs. En la Ref.8 se arriba a la conclusión que las cadenas Xns que describen la sucesión de los estados de las MQs, son irreducibles, periódicas de período dos, y recurrentes.

3. INDICE DE DESEMPEÑO PARA LAS MQs

Con el objeto de evaluar el desempeño de las MQ debemos definir una función de costo o índice que permitan juzgar su comportamiento.

A primera vista parece que un buen objetivo de funcionamiento, para cada una de las realizaciones, es que la cantidad de bits recibidos con la MQ enganchada sea máximo, para lo cual debe darse que la MQ permanezca enganchada (estado E) la mayor cantidad de tiempo posible (no olvidemos que la mínima unidad de tiempo es el tiempo de duración de un bit).

Si bien la proporción de tiempo en que la máquina está enganchada puede ser un índice adecuado para comparar el desempeño de distintas MQs; su bondad o utilidad dependerá del contexto en que sean utilizadas.

No debemos olvidar que estamos interesados en recibir paquetes de información (conjuntos de bits en secuencia y no bits individuales); por lo tanto

> La MQ debe permanecer en el estado E durante toda la recepción (todos los bits) de información del paquete para que sea considerado aceptable. (3.1)

En consecuencia definimos los eventos (disjuntos y complementarios) PE y PD:

PE: El procesamiento de cada uno los bits de información del paquete se hace con la MQ enganchada.

PD: El procesamiento de al menos uno de los bits de información (uno o más bits de información) del paquete se hace con la MQ desenganchada.

Claramente PE y PD son mutuamente exclusivos y su unión es el evento cierto; por lo tanto:

$$Pr(PE) = 1 - Pr(PD) \qquad (3.2)$$

Dado que para considerar aceptable un paquete debemos recibir todo el paquete engachando, concluimos Pr (PE) es mejor que la proporción de tiempo que MQ está enganchada como índice de desempeño.

A continuación calcularemos Pr(PE) valiéndonos del hecho que el comportamiento de MQ puede ser descripto por la cadena de Markov Xn.

Precisando un poco más la sentencia 3.1 la MQ debe:

- Estar en E durante la recepción del primer bit de información del paquete.

- Permanecer en E durante todo el tiempo que demora la recepción de cada uno de los restantes bits de información del paquete.

Es conveniente hacer la precisión anterior si suponemos que la recepción de paquetes debe realizarse en forma aperiódica. Más aún si se pierde el sincronismo durante el tiempo en que no se reciben paquetes.

Habitualmente los paquetes de comunicaciones tienen un "encabezamiento" en su comienzo, parte del cual está destinado a lograr sincronización. A continuación de dicho encabezamiento se transmiten los bits de información. Por lo tanto la MQ debe arribar al estado E antes de que finalice la parte de sincronización del encabezamiento de cada paquete.

Podemos considerar al paquete como dividido en (M-1) bits para sincronización y los restantes (L+1) bits como de información. Entonces la probabilidad de que la MQ esté desenganchada en al menos un bit de información está dada por:

$$\begin{aligned} Pr(PD) &= Pr(PD.(XM=E)) + Pr(PD.(XM=D)) = \\ &= Pr(PD.(XM=E)) + Pr(PD/XM=D) * Pr(XM=D) = \\ &= Pr(PD.(XM=E)) + Pr(XM=D) = \end{aligned} \qquad (3.3)$$

pues por definición Pr(PD/XM=0)=1. Con (3.2) se tiene:

$$Pr(PD) = Pr(PD.XM=E) + (1 - Pr(XM=E)) \qquad (3.4)$$

En la Ref.8 se calculan Pr(PD.XM=E) y Pr(XM=E). A continuación transcribimos las ecuaciones finales:

$$Pr(PD.XM=E) = \qquad (3.5)$$

$$\left[p0,p1,..pN\right] P^{M} \left[1(e1),\ 1(e2),..\right] \ (P(E,E)-I)^{-1} \ (P^{L}(E,E)-I)\ P(E,D)\ 1$$

y

$$Pr(XM=E) = \left[p0,p1,..pN\right] P^{M}\ 1(E) \qquad (3.6)$$

De las ecuaciones (3.5) y (3.6) obtenemos una expresión para Pr(PE):

(3.7)

$$Pr(PE) = [p0, p1, .. pN]\ P^{M}\ (1(E) - [1(e1),\ 1(e2), ..]\ (P(E,E)-I)^{-1}\ (P^{L}(E,E)-I)\ P(E,D)\ 1)$$

donde:

ei	Estado enganchado número i.
di	Estado desenganchado número i.
E	Unión de todos los ei.
D	Unión de todos los di.
$[p0, p1, .. pN]$	Vector fila de distribución de probabilidades iniciales. Expresa la probabilidad de estar en el estado i al comienzo del primer bit de sincronismo de un paquete. Como no hay motivo para favorecer un estado de la MQ sobre otro, suponemos la situación más desfavorable (en el sentido de máxima entropía) que es la distribución uniforme.
1(x)	Vector columna de dimensión Nx1, con todos ceros excepto un uno en la posición (x).
1	Vector columna de todos unos, de dimensión N/2x1.
P	Matriz de Markov de la máquina de estados del sincronizador

A fin de simplificar los cálculos y la notación se ha dividido a P en cuatro submatrices:

$$P = \begin{vmatrix} P(E,E) & P(E,D) \\ P(D,E) & P(D,D) \end{vmatrix}$$

4. RESULTADOS NUMERICOS

En la fórmula 3.7 pueden distinguirse los siguientes parámetros:

a. N, número de estados de la MQ.
b. pe, probabilidad de error al bit en el canal de comunicaciones
c. M-1, número de bits de sincronismo del paquete.
d. L+1, número de bits de información del paquete.

En la gráfica 1 se muestran los resultados de la evaluación de la fórmula 3.7 para:

a. N variando entre 4 y 32 estados;

b. pe variando entre 10^{-6} y 10^{-1} ;
c. M-1 igual a 100 bits;
d. L+1 igual a 1000 bits;

o sea que la gráfica 1 muestra Pr(PD) en función de pe, con N como parámetro, y M y L fijos. Si bien el índice de desempeño Pr(PE) debe estar más cerca a a uno cuanto mejor sea el comportamiento de la MQ, de (3.2) se ve es lo mismo que Pr(PD) sea lo más pequeño posible. Analizamos Pr(PD) para facilitar la visualización de los resultados.

En la gráfica 1 se ve que Pr(PD) crece en función de pe para todos los valores de N. Esto significa que el índice de desempeño baja a medida que la probabilidad de error aumenta.

Para un valor dado de pe y para los valores bajos de N, se ve que el desempeño mejora al aumentar en número de estados del sincronizador. Sin embargo para igual pe y valores mayores de N el desempeño desmejora. Esta situación es más evidente para valores bajos de la probabilidad de error.

Se observa que para una cierta "calidad" del canal de comunicaciones, es decir un valor fijo de pe, el desempeño presenta un óptimo con relación al número de estados de la máquina MQ. Esto puede visualizarse claramente en la gráfica 2.

Por lo tanto para cada aplicación especifica (tipo de canal de comunicaciones) del sincronizador donde se prevea una franja de valores de pe, existirá una franja de valores de N que obtienen los mejores desempeños.

Este comportamiento se explica del siguiente modo. Un N elevado favorece el comportamiento en estado E, adquirido durante la sincronización, a lo largo de la parte de información. Sin embargo si N es demasiado elevado el sincronizador se hace lento y no alcanzará el tramo de sincronización para alcamzar los estados de enganche.

La ventaja de usar N elevado radica en su robustez, o sea el poder desempeñarse muy bien para valores muy distintos de la probabilidad de error de bit (canales muy variables), al precio de tener que alargar el tramo de sincronización del paquete.

5. CONCLUSIONES

Se ha estudiado un tipo de sincronizadores de datos para códigos convolucionales fácilmenente implementables como máquinas de número de estados finito.

Se demostró que la secuencia de estados del sincronizador constituía una cadena de Markov, y este conocimiento sirvió de base para el análisis del comportamiento de estos sincronizadores en función del nro de estados.

Del análisis de los resultados numéricos se concluye que, si se conoce el valor máximo de la probabilidad de error al bit en el canal, existe un valor óptimo para el número de estados del sincronizador.

Utilizando la teoria de las cadenas de Markov pueden estudiarse otros tipos de sincronizadores con distinta estructura de la máquina de estados.

REFERENCIAS

1. Sloane N.J.A.. A Short Course on Error Correcting Codes. CISM, Udine, Italia.

2. Lorente Hugo. Un sincronizador de datos para códigos convolucionales. CeTAD. Facultad de Ingeniería. U.N.L.P. Manuscrito del autor aceptado para su publicación en el boletín del Programa Nacional de Informática y Electrónica.

3. Cinlar E. Introduction to Stochastic Processes. Prentice - Hall. New Jersey. 1975.

4. S. Karlin y H.M. Taylor. A First Course in Stochastic Processes. Second Edition. Academic Press. Vol I.

5. Koroliuk, V. S.. Manual de la Teoria de Probabilidades y Estadistica Matemática. Editorial Mir. Moscú. 1981.

6. Papoulis A. Probability, Random Variables, and Stochastic Processes. McGraw Hill. 1965.

7. Stern J. et.al.. Méthodes Pratiques d´Etude des Fonctions Aléatoires. Dunod. Paris. 1967.

8. O. N. Bria, H. E. Lorente y C. H. Muravchik. Análisis de un sincronizador para datos codificados convolucionalmente. Informe interno 4/88. Laboratorio de Comunicaciones. Departamento de Electrotecnia. Facultad de Ingeniería. U.N.L.P. 1988.

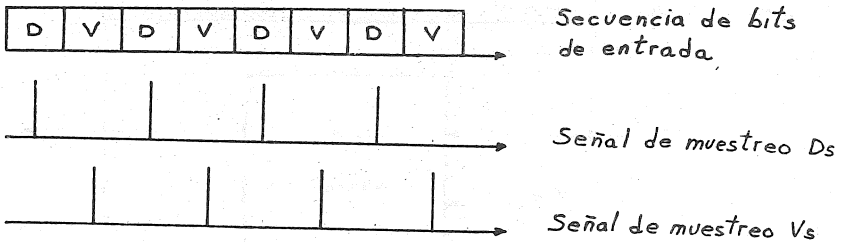


FIG 1.1. Señales de muestreo con la fase correcta para una determinada secuencia de entrada. D= dato, V= verificación.

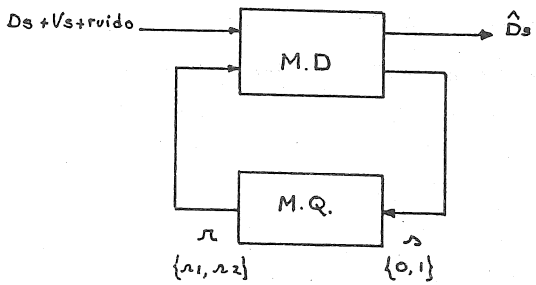


FIG 1.2. Relación funcional entre MD y MQ, y variables binarias

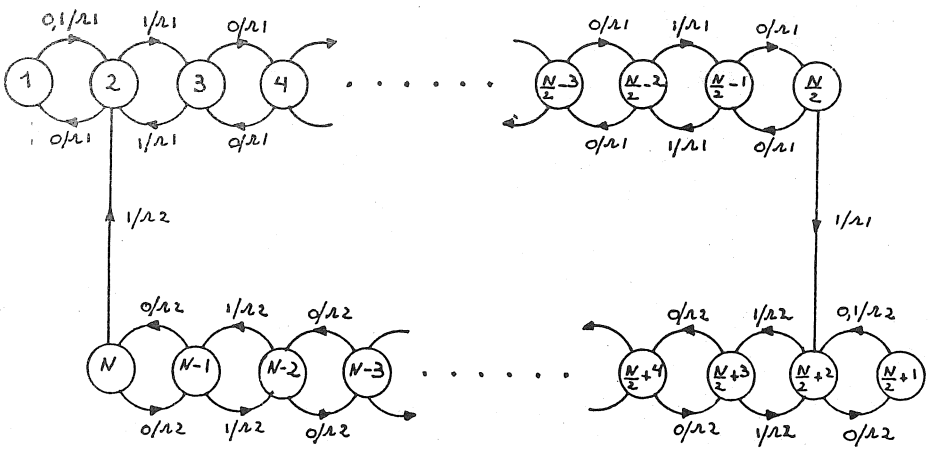


FIG. 1.3 Diagrama de estados y transiciones de una MQ genérica

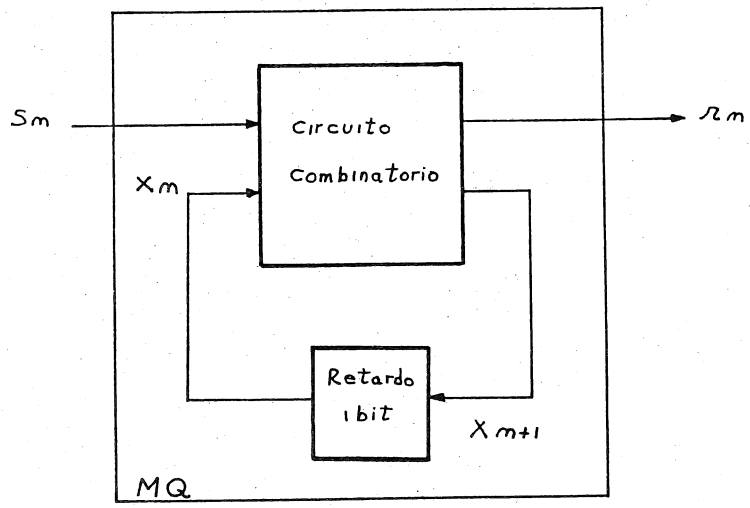


FIG. 2.1. MQ como máquina de estados donde se ponen en evidencia los procesos aleatorios intervinientes.

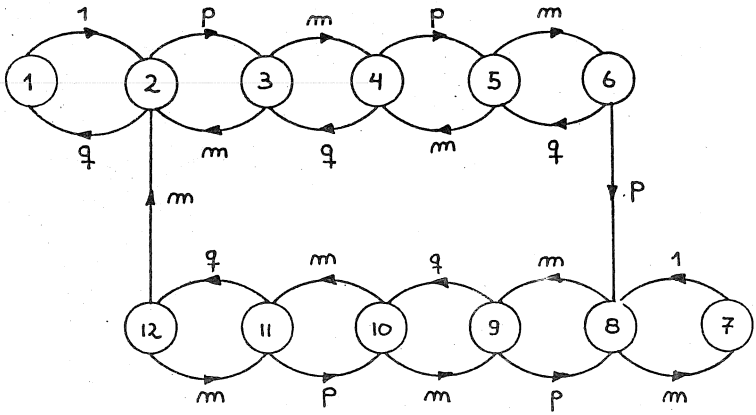
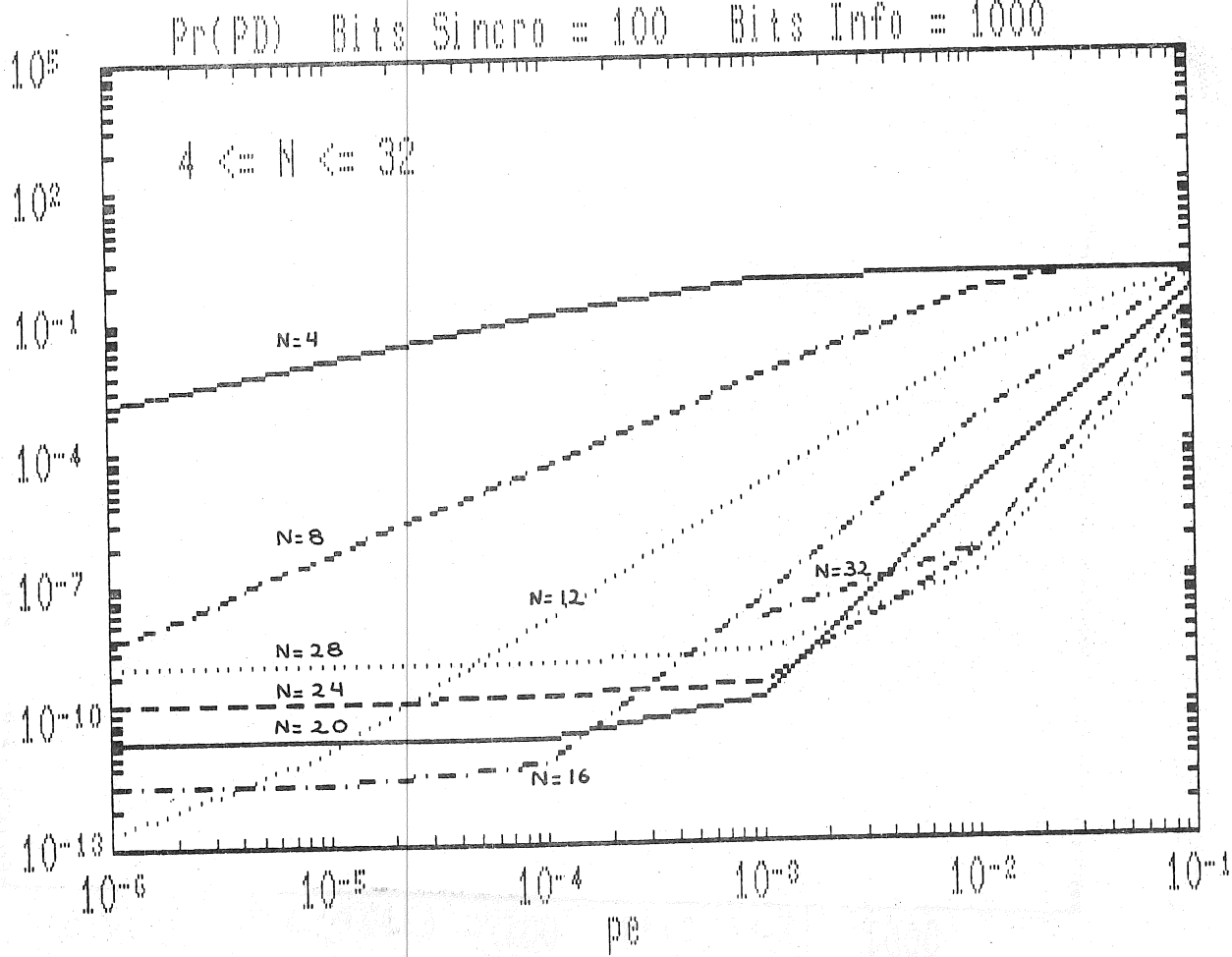
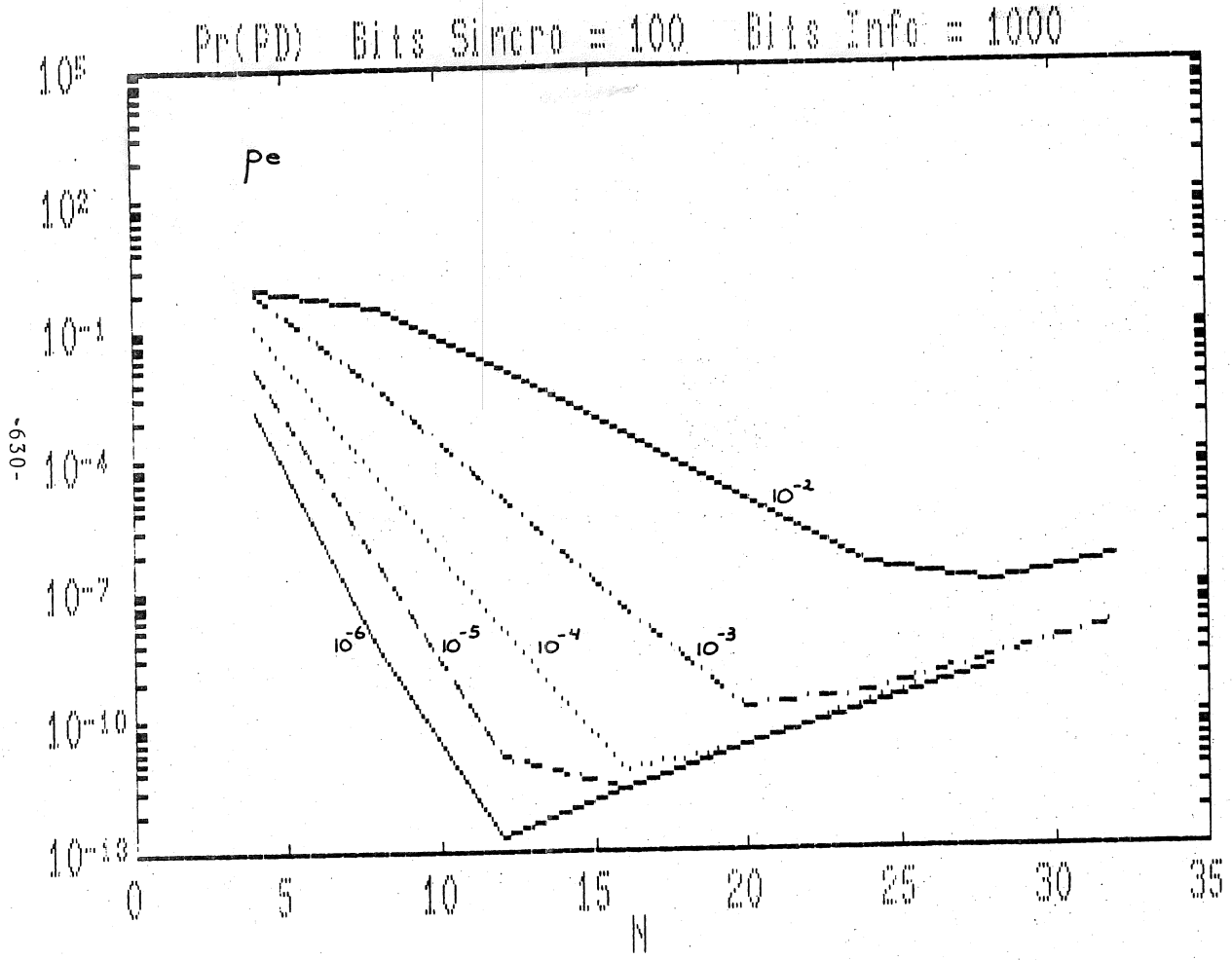


FIG. 2.2. Diagrama de estados y transiciones de la cadena de Markov de MQ(12), considerando a r1 como la respuesta correcta.


Pr(PD) Bits Sincro = 100 Bits Info = 1000
4 <= N <= 32
10^5
10^2
10^-1
10^-4
10^-7
10^-10
10^-13
N=4
N=8
N=12
N=16
N=20
N=24
N=28
N=32
10^-6
10^-5
10^-4
10^-3
10^-2
10^-1
pe

GRAFICA 1


Pr(PD) Bits Sincro = 100 Bits Info = 1000
Pe
10^{5}
10^{2}
10^{-1}
10^{-4}
10^{-7}
10^{-10}
10^{-13}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
5
10
15
20
25
30
35
N

GRAFICA 2

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

EFECTO DEL USO DE PRE-ENFASIS Y DE-ENFASIS EN ENLACES DIGITALES DE AFSK SOBRE FM

Hugo Lorente

CETAD, Facultad de Ingenieria

LIDI, Facultad de Ciencias Exactas

Universidad Nacional de La Plata

EFECTO DEL USO DE PRE-ENFASIS Y DE-ENFASIS EN ENLACES DIGITALES DE AFSK SOBRE FM

RESUMEN

El presente articulo estudia la ventaja ó desventaja del empleo de pre-ènfasis y de-ènfasis en enlaces digitales del tipo AFSK sobre FM (manipulaciòn por desplazamiento de audio frecuencia sobre una portadora de modulaciòn de frecuencia).

El uso de pre-ènfasis y de-ènfasis es habitual en enlaces de voz sobre FM debido a la mejora en la relaciòn señal a ruido obtenida a la salida del receptor.

Para el caso de AFSK sobre FM se muestra que el uso de pre-ènfasis y de-ènfasis resulta en una disminuciòn en la relaciòn señal a ruido a la salida del receptor sobre la que se obtiene en un enlace con respuesta de video plana.

INTRODUCCION

El creciente empleo del procesamiento digital està resultando en un constante aumento de la necesidad de medios seguros de comunicaciòn de datos.

En algunos casos, como en el servicio mòvil terrestre ó en enlaces punto a punto entre lugares de difícil acceso, los enlaces de radio con modulaciòn de FM, diseñados originalmente para la transmisiòn de voz, estàn siendo adaptados para la transmisiòn digital.

La pràctica habitual en estos equipos es el empleo de pre-ènfasis en la transmisiòn conjuntamente con el correspondiente de-ènfasis en la recepciòn debido a la mejora en la relaciòn señal a ruido obtenida en el enlace. Dicha mejora se debe principalmente a dos hechos: al espectro de las señales de voz que decrece con la frecuencia y al espectro de ruido que en enlaces de FM tiene una caracteristica cuadràtica con la frecuencia [1].

Al crecer la demanda de enlaces digitales sobre los medios de comunicaciòn existentes es natural que las pràcticas habituales se mantengan. Sin embargo, si se emplea el mètodo de AFSK para la transmisiòn de datos no puede afirmarse a priori que existan ventajas en el uso de pre-ènfasis y de-ènfasis debido a las diferencias considerables entre el espectro de las señales de voz y el de las señales de AFSK.

A continuaciòn se calculan las relaciones de señal a ruido a la salida de un enlace de AFSK sobre FM para el caso particular en que la desviaciòn màxima està acotada y los filtros de canal y de video pueden considerarse rectangulares. Los resultados indican la desventaja del uso de tècnicas de pre-ènfasis/de-ènfasis con respecto una modulaciòn de frecuencia plana por la señal de AFSK.

AFSK SOBRE FM SIN PRE-ENFASIS/DE-ENFASIS

En la figura 1 se muestra un enlace digital de AFSK so-

bre FM. Se supone que la fuente digital entrega un flujo continuo de datos binarios con una representación del tipo:

$$x(t)= \sum_{k} ak*Pg(t-k*T) \quad \text{..... Ec 1}$$

donde x(t) es la tensión a la salida de la fuente digital, ak ($-\infty <= k <= \infty$) depende del simbolo binario a transmitir según:

$$ak= \begin{cases} +1 & \text{k-ésimo digito = 1} \\ -1 & \text{k-ésimo digito = 0} \end{cases} \quad \text{.....Ec 2}$$

y Pg tiene la forma mostrada en la figura 2, donde T es la inversa de la tasa de bit.

El generador de AFSK convierte la entrada digital en un tono de audio modulado en frecuencia según:

$$y(t) = \cos[\ 2*PI*Fo*t + Kv\int x(t)dt\ ]$$

$$= \cos[\ \theta\ ] \quad \text{.....Ec 3}$$

donde Fo es la frecuencia central de la portadora de audio y Kv es la sensibilidad de modulación.

La frecuencia instantánea a la salida del generador de AFSK es:

$$Fi = (1/2*PI)*d\theta/dt =$$

$$= Fo + (\ Kv/2*PI\ )*x(t)$$

$$= Fo \pm (Kv/(2*PI))*V \quad \text{.....Ec 4}$$

Esto indica que tenemos dos tonos distintos de audio que dependen del simbolo binario a transmitir.

La constante Kv se elige de tal manera que:

$$Kv*V/(2*PI) = Dv \quad \text{.....Ec 5}$$

donde Dv es la desviación de frecuencia de la portadora de AFSK. Se destaca que, como la frecuencia instantánea debe ser mayor que cero, Dv es siempre menor que Fo.

Reescribiendo la EC. 3 resulta:

$$y(t) = \cos[\ 2*PI*\{Fo*t+Dv\int x(t)dt\}] \quad \text{.....Ec 6}$$

Esta señal se aplica a un oscilador controlado por tensión (VCO) el cual genera la portadora de FM. El mismo se su-

pone que tiene una relaciòn entre la tensiòn de entrada y la frecuencia de salida tal como:

$$F(t) = Fp + Kf*y(t)/(2*PI) \quad \text{.....Ec 7}$$

donde Fp es la frecuencia central de la portadora de FM, F(t) es la frecuencia instantànea de la misma y Kf es la sensibilidad de modulaciòn de frecuencia del VCO.

En estas condiciones la señal a la salida del transmisor de FM es:

$$S(t) = A*\cos[\ 2*PI*\int F(t)dt] =$$

$$= A*\cos[\ 2*PI*Fp*t+Kf\int y(t)dt\ ] \quad \text{.....Ec 8}$$

de la EC. 7 la desviaciòn instantànea de la portadora de FM resulta:

$$D(t) = F(t) - Fp = Kf*y(t)/(2*PI) \quad \text{.....Ec 9}$$

y de la ecuaciòn 6 la desviaciòn pico es:

$$Dmàx = Kf/(2*PI) \quad \text{.....Ec 10}$$

Como Dmàx està fijada de antemano la constante Kf queda establecida.

En el canal de comunicaciòn a la señal se agrega ruido aditivo. Este ruido se supone de densidad espectral unilateral constante N, por lo menos en el ancho de banda del canal.

El mòdulo de la funciòn de tranferencia del filtro de canal se muestra en la figura 3. El ancho de banda de este filtro (2*Bc) es normalmente bastante mayor que la desviaciòn pico a pico de la portadora de FM.

Se supone que el ruido que pasa por el filtro de canal admite una representaciòn en componentes cuadràticas tal como:

$$n(t)=nc(t)\cos[2*PI*Fp*t]-ns(t)\cos[2*PI*Fp*t] \quad \text{.....Ec 11}$$

donde nc(t) y ns(t) tienen una densidad espectral tal como se muestra en la figura 4.

La señal màs el ruido filtrados son aplicados al discriminador donde las variaciones de frecuencia a su entrada son convertidas nuevamente en variaciones de tensiòn segùn:

$$Vs(t) = Kd*[F(t)-Fp] = Kd*D(t) \quad \text{.....Ec 12}$$

donde Vs es la tensiòn de salida del discriminador, F(t) es la frecuencia instantànea de la señal màs el ruido y Kd es la ganancia del discriminador.

Finalmente la salida del discriminador se aplica al filtro de video. Este filtro debe ser simètrico alrededor de la frecuencia de la portadora de audio y debe permitir el paso de las componentes de la modulaciòn digital [2]. Por simplicidad se supone que este filtro es rectangular como se muestra en la figura 5, donde 2*Bv es su ancho de banda.

En estas condiciones es posible calcular la relaciòn señal a ruido a la salida del enlace. Para ello se supone que el efecto del ruido es aditivo a la salida del discriminador. Este anàlisis simplificado es adecuado cuando la relaciòn señal a ruido a la entrada del discriminador es elevada [3].

Suponiendo que no hay ruido la señal a la salida del sistema resulta segùn las Ec. 6, 9 y 12:

$$Vs = Kd*Dmàx*\cos[2*PI*\{Fo*t+Dv\int x(t)dt\}] \quad \text{...Ec 13}$$

y la potencia de señal:

$$S = ((Kd*Dmàx)^2)/2 \quad \text{.....Ec 14}$$

Si ahora se supone que no hay modulaciòn, a la entrada del discriminador tenemos la portadora de FM màs el ruido:

$$S(t) + n(t) = A*\cos[2*PI*Fp*t] + n(t) \quad \text{.....Ec 15}$$

reemplazando el ruido por la Ec 11, tenemos:

$$S(t) + n(t) =$$

$$=(A+nc)\cos[2*PI*Fp*t]-ns*sen[2*PI*Fp*t]$$

$$=R(t)*\cos[2*PI*FP*t+\varphi(t)] \quad \text{.....Ec 16}$$

donde:

$$\varphi(t) = Arctang[\, ns(t)/(A+nc(t)] \quad \text{.....Ec 17}$$

como estos càlculos se han realizado en la suposiciòn de una alta relaciòn señal a ruido a la entrada del discriminador, podemos escribir:

$$\varphi(t) \cong ns(t)/A \quad \text{.....Ec 18}$$

y la desviaciòn instantànea de la portadora de FM màs el ruido resulta:

$$D(t) = F(t)-Fp = (1/(2*PI))*d\varphi/dt=$$

$$= (1/(2*PI*A))*dns(t)/dt \quad \text{.....Ec 19}$$

de la Ec 12 a la salida del discriminador tenemos:

$$Vs = Kd*(1/(2*PI))*dns(t)/dt \quad \text{.....Ec 20}$$

esta tensiòn de salida del discriminador tiene una densidad espectral:

$$S(f) = 2*N*((Kd/(2*PI*A))*(2*PI*f))^2 \quad \text{....Ec 21}$$

para f menor que Bc, donde f es la frecuencia.

Esta densidad espectral se muestra en la figura 6 en la cual se ha superpuesto la transferencia del filtro de video. Con estos datos podemos calcular la potencia de ruido a la salida del sistema de comunicaciòn como:

$$No = \int_{Fo+Bv}^{FO-Bv} S(f)df =$$

$$= 4*N*(Kd/A)^2*(Fo^2*Bv+Bv^3/3) \quad \text{.....Ec 22}$$

y la relaciòn señal a ruido como:

$$(So/No) = (Dmàx*A)^2/(8*N*(Fo^2*Bv+Bv^3/3) \quad \text{.....Ec 23}$$

PRE-ENFASIS Y DE-ENFASIS

En la figura 7 se muestra el sistema de la figura 1 modificado de tal manera de incluir el pre-ènfasis en el transmisor y el de-ènfasis en el receptor. Se supone que el circuito de de-ènfasis es un derivador con una transferencia:

$$Hp(f) = j*2*PI*f \quad \text{.....Ec 24}$$

con $j^2=-1$, y el circuito de de-ènfasis es un integrador con

una transferencia:

$$Hd(f) = 1/(j*2*PI*f) \quad \text{.....Ec 25}$$

En estas condiciones podemos escribir la ecuaciòn 8 como:

$$S(t) = A*\cos[2*PI*Fp*t+Kf'*y(t)] \quad \text{.....Ec 26}$$

donde Kf' es la nueva sensibilidad del VCO distinta, en general, a la anterior. La desviaciòn instantànea de la portadora de FM resulta:

$$D(t) = (Kf'/(2*PI))*dy(t)/dt =$$

$$= -Kf'*(F0+Kv*x(t)/(2*PI))*\text{sen}[\ \theta\ ] \quad \text{....Ec 27}$$

con una desviaciòn màxima:

$$Dmàx = Kf'*(Fo+Kv*V/(2*PI)) =$$

$$= Kf'*(Fo+Dv) \quad \text{.....Ec 28}$$

y una desviaciòn mìnima:

$$Dmin = Kf'*(Fo-Dv) \quad \text{.....Ec 29}$$

Queda claro que al emplear pre-ènfasis la desviaciòn de la portadora de Fm depende del simbolo binario transmitido. De todas maneras, como Dmàx està fijada de antemano, el segundo tèrmino de la Ec 28 queda establecido.

Reescribiendo la Ec 27 tenemos:

$$D(t) = (Dmàx/(2*PI*(Fo+Dv))*dy(t)/dt \quad \text{.....Ec 30}$$

Luego que la señal pasa por el discriminador y por el circuito de de-ènfasis la tensiòn de salida resulta:

$$Vs(t) = Kd\int D(t)dt =$$

$$= (Kd*Dmàx/(2*PI*(Fo+Dv))*y(t) \quad \text{.....Ec 31}$$

con una potencia de señal:

$$S = ((Kd*Dmàx/(2*PI*(Fo+Dv))\hat{}2)/2 \quad \text{.....Ec 32}$$

y el ruido a la salida resulta:

$$Vs(t) = Kd*(1/2*PI*A)*ns(t) \quad \text{.....Ec 33}$$

Como la densidad espectral de ns(t) es conocida podemos calcular la potencia de ruido a la salida del filtro de video:

$$No = 4*N*(Kd/(2*PI*A))^2*Bv \quad \text{.....Ec 34}$$

de donde resulta la relación señal a ruido:

$$(So/No)* = (A*Dmax)^2/((Fo+Df)^2*8N*Bv) \quad \text{.....Ec 35}$$

donde (So/No)* indica la relación señal a ruido calculada incluyendo pre-ènfasis y de-ènfasis en el sistema.

RESULTADOS

Si a la relación señal a ruido dada por la ecuación 35 la dividimos por la obtenida en la ecuación 23, tendremos evaluada la ventaja ó desventaja de emplear pre-ènfasis y de-ènfasis.

$$R = (So/No)*/(So/No) =$$

$$= (Fo^2+Bv^2/3)/((Fo+Dv)^2) \quad \text{.....Ec 36}$$

Si existe alguna ventaja al emplear pre-ènfasis y de-ènfasis R debe ser mayor que la unidad, esto es:

$$R = (Fo^2+Bv^2/3)/((Fo+Dv)^2) > 1$$

ó lo que es lo mismo:

$$Bv^2 > 6*Fo*Dv+3*Dv^2 \quad \text{.....Ec 37}$$

De las ecuaciones 4 y 5:

$$Dv < Fo \quad \text{.....Ec 38}$$

reemplazando en la Ec 37:

$$Bv > 3*Dv \quad \text{.....Ec 39}$$

La ecuaciòn 39 indica que para que haya alguna ganancia en el empleo de pre-ènfasis y de-ènfasis es necesario que el ancho de banda de video sea por lo menos tres veces la desviaciòn de la portadora de audiofrecuencia. Por otra parte de la ecuaciones 23 y 35 se deduce que la relaciòn señal a ruido disminuye al aumentar el ancho de banda de video, siendo por lo tanto deseable que este ancho de banda sea el menor posible.

De los espectros de la señales de FSK [2] se puede afirmar que tal ancho de banda de video es innecesario y que por lo tanto en la pràctica no es ventajoso el empleo de pre-ènfasis y de-ènfasis.

A titulo de ejemplo supongamos un sistema de AFSK-FM con una frecuencia central de portadora Fo=1700 Hz, una desviación pico a pico de la portadora de AFSK 2*Dv=1000Hz y un ancho de banda de video 2*Bv=1500 Hz (Modem Bell 202) . En estas condiciones la pèrdida causada por el empleo de pre-ènfasis y de-ènfasis resulta:

$$1/R = ((Fo+Dv)^2)/(Fo^2+Bv^2/3) =$$

$$= 1.573 = 1.97 \text{ dB} \quad \text{.....Ec 40}$$

REFERENCIAS

[1] Panther, P. F.: "Modulation, Noise, and Spectral Analysis", Mc Graw-Hill Book Company, New York.

[2] Lucky, R. E. et al: "Principles of Data Communication", Mc Graw-Hill Book Company, New York.

[3] Taub, H. et al: "Principles of Communications Systems", Mc Graw=Hill Book Company, New York.

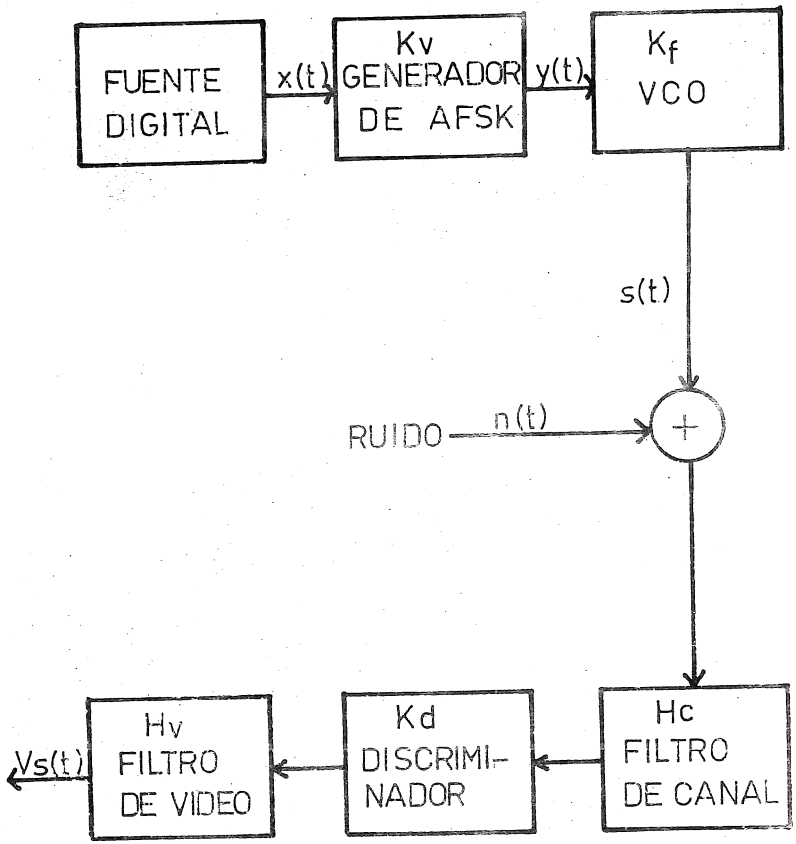

FUENTE DIGITAL
x(t)
Kv GENERADOR DE AFSK
y(t)
Kf VCO
s(t)
RUIDO
n(t)
+
Hc FILTRO DE CANAL
Kd DISCRIMI-NADOR
Hv FILTRO DE VIDEO
Vs(t)

FIGURA 1

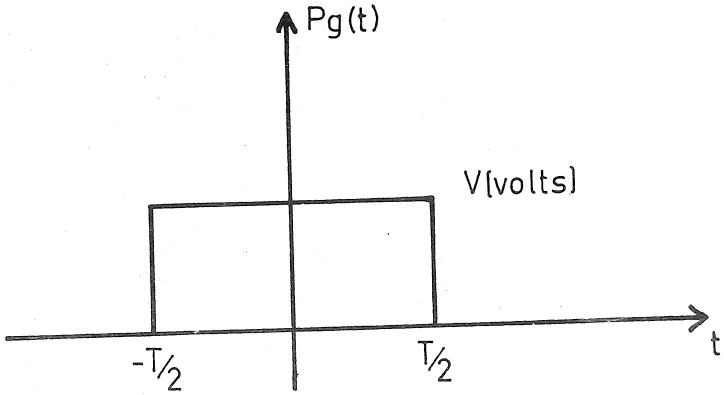
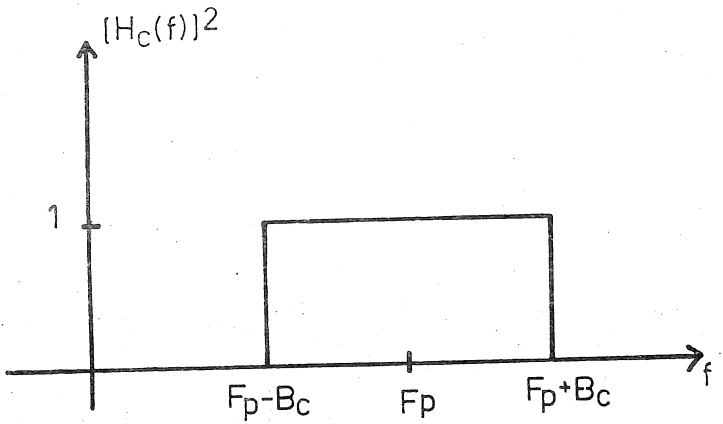
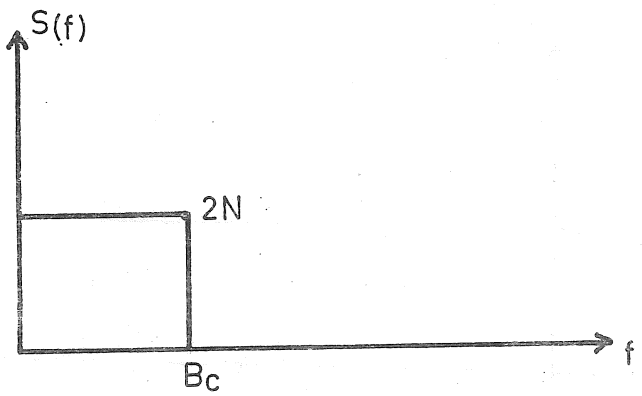


FIGURA 2



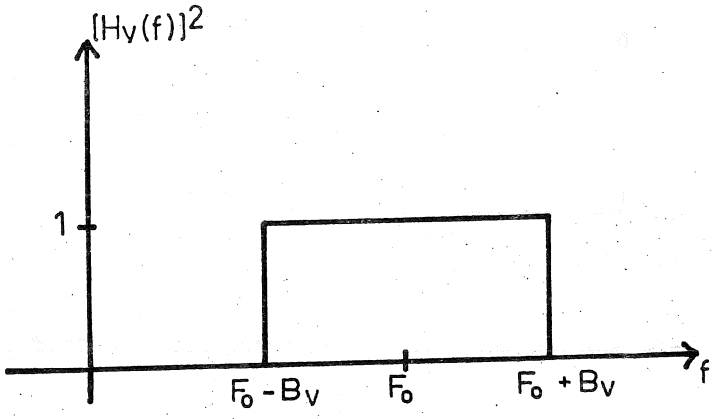
TRANSFERENCIA DEL FILTRO DE CANAL

FIGURA 3



DENSIDAD ESPECTRAL DE LAS COMPONENTES DE RUIDO

FIGURA 4



FILTRO DE VIDEO

FIGURA 5

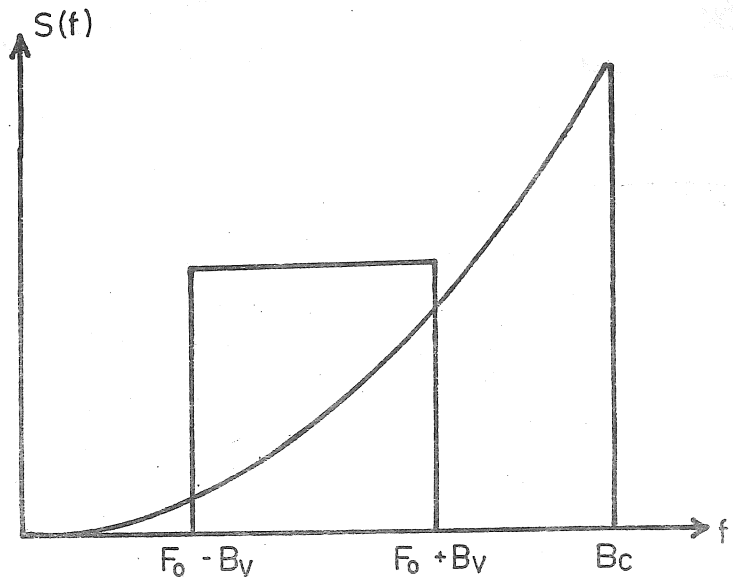


FIGURA 6

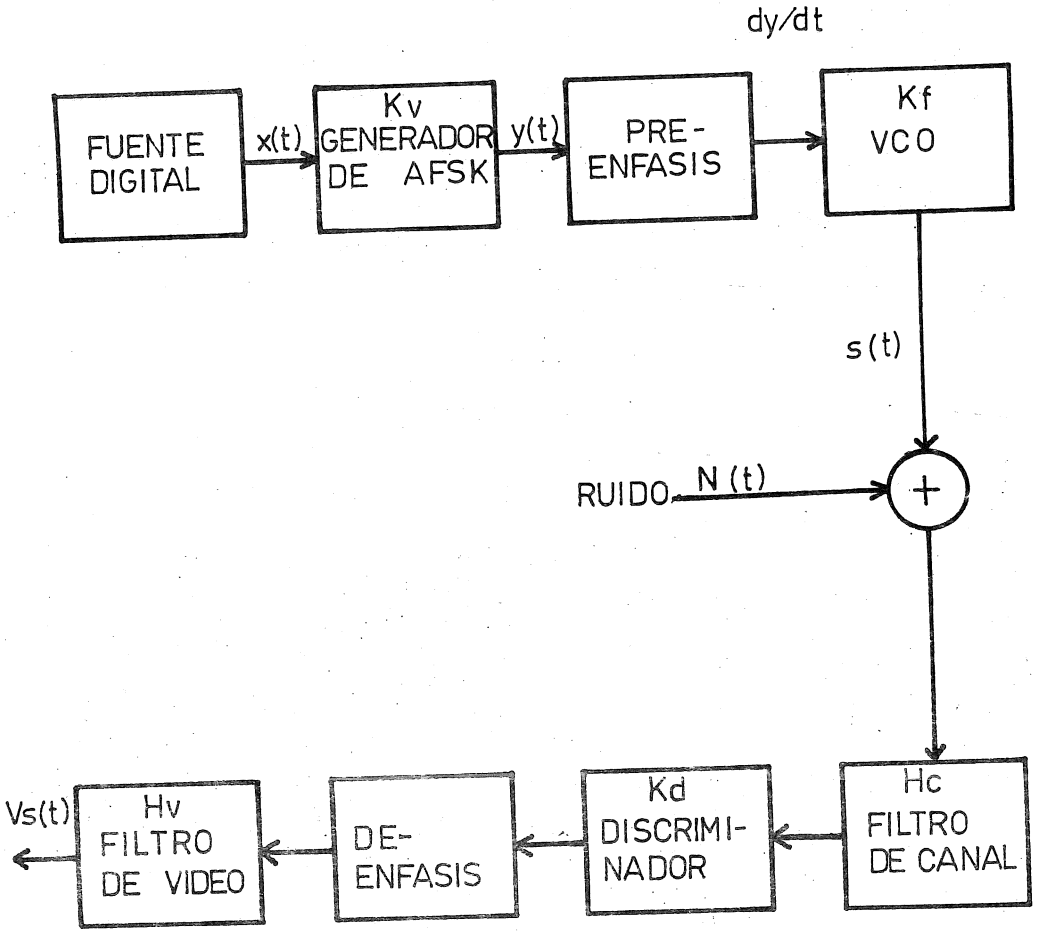

dy/dt
FUENTE DIGITAL
x(t)
Kv GENERADOR DE AFSK
y(t)
PRE-ENFASIS
Kf VCO
s(t)
RUIDO N(t)
+
Hc FILTRO DE CANAL
Kd DISCRIMI-NADOR
DE-ENFASIS
Hv FILTRO DE VIDEO
Vs(t)

FIGURA 7

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

EVALUACION DE COMPORTAMIENTO DE PROTOCOLOS DE REDES LOCALES NORMALIZADAS EN AMBIENTES CRITICOS

J. Santos, J. Orozco* y O. Alimenti
* Becario del CONICET
Dep. de Ingeniería Eléctrica, Universidad Nacional del Sur
8000 Bahía Blanca, Argentina

RESUMEN: En el presente trabajo se realiza una evaluación comparativa de los procolos de redes locales normalizados por el IEEE: Barra con Ficha (802.4), Anillo con Ficha (802.5) y una modificación del Acceso Múltiple por Sensado de Portadora con Detección de Colisión (802.3), operando en ambientes con constricciones de tiempo. Se analizan las disciplinas de prioridades necesarias para que el sistema esté libre de crisis según las relaciones que vinculan el tiempo de ranura, el número de nodos de la red y sus tiempos de crisis. En cada caso se obtienen los rendimientos (como un cierto % del ancho de banda) y la producción del sistema (en Mb/s), como una función de la longitud del campo de datos transmitido.

INTRODUCCION

En algunos casos, el flujo de información entre nodos de una red local que comparten un medio de comunicación tiene ciertas constricciones de tiempo; éstas surgen del hecho de que un mensaje generado en un cierto nodo debe llegar al nodo receptor dentro de un cierto intervalo de tiempo. Este intervalo se denomina *tiempo de crisis* del nodo emisor y se dice entonces que la red opera en un *ambiente crítico* o *con constricciones de tiempo*.

El estudio de los parámetros que gobiernan el comportamiento de tales sistemas no es nuevo. Por ejemplo, en [1] y [2], de 1967 y 1971 respectivamente, las relaciones entre el número de unidades del sistema, sus tiempos de crisis y el tiempo que cada unidad puede usar el medio ***(tiempo de ranura)*** están analizados en relación con las disciplinas de prioridades a ser aplicadas para que el sistema esté libre de crisis.

Con el advenimiento de las redes locales y su uso en ambientes críticos típicos, como por ejemplo control en tiempo real, se ha producido un renovado interés en el tema [3, 4, 5, 6, 7, 8 y 9]. En [9] se construyó primero el mapa de un conjunto de protocolos de redes locales sobre el conjunto de disciplinas de prioridades y luego el mapa de éstas sobre el conjunto de relaciones ***número de unidades/tiempos de crisis/tiempo de ranura***.

Es evidente que en un ambiente crítico, los nodos deben tener un tiempo garantizado de acceso al medio; en consecuencia existe un tiempo máximo entre el final de dos mensajes sucesivos cuando un nodo está transmitiendo y por lo menos otro tiene un mensaje generado, listo para ser transmitido; ese tiempo es casualmente el tiempo de ranura e incluye no sólo el tiempo necesario para transmitir y propagar el mensaje sino también un cierto "gasto general" empleado principalmente para determinar cuál es el nodo que adquiere el derecho a transmitir (Fig. 1).

Una ***disciplina de prioridades*** es una relación binaria de orden definida en el conjunto de nodos. El conjunto ordenado se denomina ***pila de prioridades***; al comienzo de cada tiempo de ranura, el medio es asignado a un cierto nodo sobre la base de información contenida en la pila.

En un ambiente crítico es posible establecer tres relaciones [2]:

$$nTr \leq TCmin \qquad (1)$$

$$\frac{TCmax}{Tr} \geq \sum_{i \in N} \backslash \frac{TCmax}{TCi} \qquad (2)$$

$$\frac{TCmax}{Tr} \geq \sum_{i \in N} \frac{TCmax}{TCi} \qquad (3)$$

en los que: n, número de nodos de la red; Tr, tiempo de ranura; TCmin, tiempo de crisis mínimo; TCmax, tiempo de crisis máximo; TCi, tiempo de crisis del nodo i; \, operador monádico techo o de redondeo por exceso; N={1,2,..,i,..n}, conjunto de nodos.
Si se cumple (1), una disciplina de Rueda Cíclica (en inglés Round Robin) asegura que el sistema está libre de crisis. En el segundo y tercer caso se requiere en cambio una disciplina de Prioridades Fijas y una disciplina de Menor Tiempo hasta Crisis, respectivamente.

Las expresiones anteriores tienen dos hipótesis subyacentes: La generación de mensajes es sincrónica con el comienzo de una ranura, y el régimen de generación de mensajes en cada nodo es, a lo sumo, la inversa del tiempo de crisis del nodo.

La diferencia aritmética entre el miembro de la derecha y el de la izquierda en (1), da el número de ranuras ociosas entre dos mensajes sucesivos del nodo con el tiempo de crisis mínimo luego del peor caso de carga del sistema; éste ocurre con la generación simultánea de mensajes en todos los nodos al comienzo de una ranura. La diferencia aritmética entre los miembros de la izquierda y la derecha en (2) y (3) da el número de ranuras ociosas entre dos mensajes sucesivos del nodo con el tiempo de crisis máximo, luego del peor caso de carga del sistema.

El rendimiento R del sistema puede ser expresado en términos del ancho de banda del medio como

$$R = Ri\ Re\ W$$

en la que Ri, rendimiento interno o intrarranura; Re, rendimiento externo o interranura; V, ancho de banda del medio.

El rendimiento intrarranura es la relación entre el tiempo realmente dedicado a la transmisión del campo de datos en la trama del mensaje y el tiempo de ranura; depende de la disciplina de prioridades y del protocolo. La disciplina de prioridades determina el gasto general necesario para establecer cuál nodo tiene el derecho a acceder al medio; el protocolo determina la relación entre la longitud del campo de datos y la longitud total de la trama del mensaje (incluyendo el preámbulo, direcciones, verificación de redundancia cíclica, etc.).

El rendimiento externo depende esencialmente del régimen de llegada de los mensajes. Si 1/TCi es el régimen de llegada de mensajes al nodo i y MCM es el mínimo común múltiplo de todos los TC, el rendimiento interranura es

$$Re = \sum_{i \in N} (MCM/TCi)/MCM$$

Evidentemente, 100 Ri Re V da el rendimiento del sistema como un porciento de su ancho de banda. 1-Re es la relación entre el número de ranuras utilizadas y el número total de ranuras. Es evidente que cuando el número de ranuras ociosas tiende a cero, Re tiende a 1; se dice entonces que el sistema está saturado.

Las ecuaciones (2) y (3) permiten una amplia distribición de los tiempos de crisis. De hecho

$$\lim_{TCmin \to Tr} TCmax = \lim_{TCmin \to Tr} ((n-1)/(1/Tr - 1/TCmin)) = \infty$$

es decir, el TC de un nodo puede tender al tiempo de ranura si el tiempo de crisis de todos los otros nodos tiende a oo.

En lo que sigue se evaluará, en ambientes críticos, el comportamiento de las tres redes locales normalizadas por el IEEE (802.3, 802.4 y 802.5). La evaluación será realizada con los protocolos respectivos sirviendo las disciplinas Rueda Cíclica (RC) y Menor Tiempo hasta Crisis (MTC), que caracterizan las condiciones de menor y mayor exigencia del sistema, respectivamente. En cada caso se harán ciertas suposiciones con respecto a la longitud del medio, retardos en la propagación a lo largo del mismo, retardos en los nodos, en los dispositivos repetidores, etc. Debe notarse que la evaluación de comportamiento supone condiciones estacionarias sin fallas en la red y por lo tanto los resultados obtenidos representan una cota superior de comportamiento.

No será considerado en el presente trabajo la disciplina intermedia, Prioridades Fijas, utilizable cuando se cumple (2); la misma puede ser servida en cuatro niveles de prioridad por el protocolo Barra con Ficha y en ocho niveles por el protocolo Anillo con Ficha.

BARRA CON FICHA (802.4)

Se suponen los siguientes valores:

Régimen de transmisión de datos: 10 Mb/s (0,8 us/byte)
Longitud del cable: 2.500 m
Número de repetidores: 4
Retardo por repetidor: 3 µs
Retardo por nodo: 10 µs
Retardo inducido por el medio: 4 ns/m

La peor configuración topológica de la red desde el punto de vista de la ubicación de los nodos a lo largo de la barra ocurre cuando dos nodos

sucesivos en el anillo lógico están en extremos opuestos del cable, de manera que entre ellos existe el tiempo máximo de propagación. El tiempo de propagación entre extremos es

$$4\ ns/m * 2.500\ m + 3\ ns/rep * 4\ rep = 22\ \mu s$$

En el análisis que sigue se usará esta configuración, la más desfavorable.

En la norma 802.4 la ficha tiene 14 bytes; el número de bytes en el campo de datos de la trama del mensaje será notado B y se supondrá que, de acuerdo a la norma, la trama tiene 12 bytes además de los correspondientes al campo de datos. Los nodos, aunque conectados a una barra, forman un anillo lógico en el cual cada estación tiene una antecesora y una sucesora. La ficha pasa de nodo en nodo en orden descendente; cuando una estación recibe una ficha direccionada para ella, toma posesión de la misma y puede enviar una trama. Cuando termina la transmisión de la trama pasa la ficha a la estación siguiente.

Rueda Cíclica

En este caso, el tiempo de ranura es la suma de:

a) Tiempo para transmitir la ficha: $0,8 * 14 = 11,2\ \mu s$
b) Retardos introducidos por el medio y los repetidores: $22\ \mu s$
c) Retardo inducido en el nodo: $10\ \mu s$
d) Tiempo para transmitir la trama: $9,6 + 0,8\ B$

Por lo tanto, el tiempo de ranura, en μs, es

$$Tr = 52,8 + 0,8\ B \qquad (4)$$

Como puede verse Tr es independiente de n.

Por otra parte, de acuerdo a (1), para un tiempo de ranura dado el tiempo de crisis mínimo admisible por el sistema es función únicamente del número de nodos. En la Fig. 2 se representa TCmin en función de la longitud del campo de datos, para varios n. En Barra con Ficha/Rueda Cíclica (BF/RC), la eficiencia intraranura resulta ser

$$Ei = 0{,}8\ B\ /\ (52{,}8 + 0{,}8\ B) \qquad (5)$$

En la Fig. 3 se representa el rendimiento del sistema como porciento del ancho de banda en función de la longitud del campo de datos para el caso en el cual

$$\forall i \quad TCi = TCmin = n\ Tr \qquad (6)$$

es decir no hay ranuras ociosas, el sistema está saturado y Re = 1.

Menor Tiempo hasta Crisis

Para servir la disciplina MTC, antes de la transmisión del mensaje es necesaria una ronda de reservas en la cual cada nodo con un mensaje listo para ser transmitido declara el tiempo que le queda hasta entrar en crisis; si el nodo no tiene mensaje generado, este tiempo será infinito; las tramas de reserva tendrán un campo de datos de cuatro bytes, haciendo un total de 18 bytes en la trama. Luego de la ronda de reserva, la ficha pasará de nodo en nodo; éstos se abstendrán de transmitir hasta alcanzar aquél con el tiempo mínimo para entrar en crisis, el cual tendrá el derecho de acceder al medio. La transmisión del mensaje es seguida por la transmisión de una ficha vacía la cual circulará hasta alcanzar el nodo más alto del sistema y comenzar a partir de él una nueva ronda de reserva. Por lo tanto, el tiempo de ranura,

en us, será la suma de:

a) El tiempo para transmitir y propagar n tramas de reserva y n fichas: (0,8*18 + 0,8*14 + 22 + 10)n
b) El tiempo para transmitir y propagar n fichas: (0,8*14 + 22 + 10)n
c) El tiempo para transmitir la trama mensaje: 9,6 + 0,8 B

Por lo tanto, el tiempo de ranura en μs es

$$Tr = 9,6 + 108,8\ n + 0,8\ B \qquad (7)$$

Debe notarse que la disciplina RC impone TOmin = n Tr como una cota inferior absoluta en los tiempos de crisis mientras que la disciplina MTC permite una distribución más flexible; el TOmin en este caso puede estar por debajo del mínimo permitido en la disciplina RC siempre que los otros nodos tengan TO tales que se cumpla (3). En la Fig. 4 se representa, para B=1.500 y diferentes n, TOmin en función de TOmax cuando en el sistema hay sólo dos tiempos de crisis diferentes; un nodo tiene TOmin y todos los demás TOmax; en cada caso se marca también el TOmin para la disciplina RC.

La cota superior en el rendimiento del sistema BF/MTC ocurrirá cuando Re=1; Ri será

$$0,8\ B/(9,6 + 108,8\ n + 0,8\ B) \qquad (8)$$

En la Fig. 5 se representa el rendimiento como % del ancho de banda, en función de la longitud del campo de datos, para n diferentes, en condiciones de saturación.

ANILLO CON FICHA (802.5)

Se suponen los siguientes valores:

Régimen de transmisión: 4 Mb/s (o 2 µs/byte)
Longitud del anillo: 2.500 m
Retardo inducido por el nodo: 250 ns
Retardo inducido por el medio: 4 ns/m
Los nodos están igualmente espaciados a lo largo del anillo.

En la norma 802.5, la ficha tiene una longitud de 3 bytes; la trama de mensaje tiene un "gasto general" de 13 bytes sobre los B bytes del campo de datos.

Rueda Cíclica

En esta disciplina de prioridades, el tiempo de ranura es la suma de:

a) Tiempo para transmitir el mensaje: 2 (13 + B) µs
b) Retardo inducido por el medio: 4 ns/m * 2.500 m = 10 µs
c) Retardo inducido por los nodos: 0,25 n µs
d) Tiempo para transmitir la nueva ficha: 3 bytes * 2 µs/byte = 6 µs
e) Tiempo para propagar la ficha hasta el próximo nodo: 4 ns/m *2.500/n m = 10/n us

Por lo tanto, el tiempo de ranura, en µs, es:

$$Tr = 42 + 0{,}25\ n + 10/n + 2\ B \qquad (9)$$

El rendimiento intrarranura será en consecuencia

$$Ri = 2\ B\ /\ (42 + 0{,}25\ n + 10/n + 2\ B) \qquad (10)$$

En la Fig. 6 se representa el rendimiento del sistema en función de la longitud del campo de datos, bajo condiciones de saturación, para diferentes n.

Menor Tiempo hasta Crisis

Al igual que en el caso de la barra con ficha, una ronda de reserva debe preceder la transmisión del mensaje; siguiendo la norma, esto implica que cada nodo en posesión de la ficha transmite su trama de reserva; la trama recorre todo el anillo y al alcanzar de vuelta el nodo emisor, es retirada de circulación;a continuación, el nodo entrega la ficha libre al nodo siguiente. Luego de la ronda, la ficha debe ser pasada a lo largo del anillo hasta alcanzar el nodo que tiene el derecho de acceso al medio; este nodo transmite su mensaje,lo retira luego de haber dado la vuelta al anillo y pasa la ficha al nodo siguiente para comenzar una nueva ronda de reserva.

El tiempo de ranura es la suma de:

a) Tiempo para transmitir n tramas de reserva, con un campo de datos de 4 bytes: $2(13 + 4)\ n\ \mu s = 34\ n\ \mu s$
b) Tiempo para propagar n tramas de reserva a lo largo del anillo: $10\ n\ \mu s$
c) Retardo inducido en los nodos: $0,25\ n^2\ \mu s$
d) Tiempo para transmitir la ficha: $6\ \mu s$
e) Tiempo para propagar la ficha: $10\ \mu s$
f) Retardo inducido en los nodos: $0,25\ n\ \mu s$
g) Tiempo para transmitir y propagar el mensaje en us (como en el caso Rueda Cíclica): $42 + 0,25\ n + 10/n + 2\ B$

Por lo tanto, Tr en μs es:

$$Tr = 42 + 61\ n + 0,25\ n^2 + 10/n + 2\ B \qquad (11)$$

En la Fig. 7 se representa TCmin en función de TCmax, con las mismas condiciones del caso previo similar, para B =1.500 y diferentes n. Finalmente, en la Fig. 8, se representa el rendimiento del sistema en porciento del ancho de banda en función de la longitud del campo de datos, en condiciones de saturación (Re=1).

CSMA/CD (802.3)

CSMA/CD es un protocolo contencioso y por lo tanto no puede garantizarse un tiempo máximo de acceso. Sin embargo, en MTC es posible operar los nodos de tal manera que el acceso sea libre de colisiones. Como en los casos anteriores, una ronda de reservas precede la transmisión del mensaje.

Los nodos son designados de acuerdo a su posición ordenada en la barra, por ejemplo 1,2,,...,n de izquierda a derecha. La ronda de reserva arranca con el nodo 1 transmitiendo su trama de reserva, en la cual declara su nombre y el tiempo que le queda hasta entrar en crisis; la trama puede ser del tamaño mínimo permitido (72 bytes). De acuerdo a la norma, 9,6 us después que el último bit de la reserva del nodo 1 ha sido percibido por el nodo 2, éste transmite su propia reserva. El proceso continúa hasta la reserva n-ésima; al recibirla, todos los nodos saben cuál es el que tiene el derecho a transmitir y difieren sus propias transmisiones.

Siguiendo a [10], la implementación electrónica puede ser modificada de tal manera que en cada nodo se puede simular la presencia de portadora en el medio. Como consecuencia, el nodo tendrá la posibilidad de acceder al medio sólo cuando no perciba portadora, ni real ni simulada. La portadora simulada se controla por contadores dedicados que comienzan a contar cuando cesa la actividad en el medio y vuelven a cero cuando se reanuda la actividad. Por lo tanto, el contador de cada nodo cuenta realmente silencio en el medio y al mismo tiempo, mediante la portadora simulada, inhibe a su propio nodo para acceder al mismo. Mediante un adecuado desplazamiento de la cuenta final en la cual cada contador se

detiene, es posible ofrecer a los nodos un acceso al medio libre de conflictos. El mecanismo, una especie de pasaje de ficha virtual, permite el arranque de la ronda de reserva aún en el caso de que haya nodos fuera de servicio.

La trama de mensaje tendrá entre 72 y 1.526 bytes, de los cuales 46 a 1.500 corresponden al campo de datos y 26 al gasto general (preámbulo, direcciones, verificación de redundancia cíclica, etc.).

Se supone lo sigúiente:

Régimen de transmisión: 10 Mb/s (0,8 µs/byte)
Longitud del cable: 2.500 m
Número de repetidores: 4
Retardo introducido por cada repetidor: 4 ns/m

Menor Tiempo hasta Crisis

El tiempo de ranura será la suma de:

a) n tramas de reserva más n intervalos de 9,6 µs entre tramas:
   67,2 n µs
b) Retardo introducido en la ida y vuelta entre extremos del cable por los repetidores y el medio: 26 µs
c) Tiempo de transmisión del mensaje: 0,8 (26+B) us

Por lo tanto, el tiempo de ranura, en µs, es:

$$Tr = 46.8 + 67,2\ n + 0,8\ B \qquad (12)$$

En la Fig. 9 se representa representa TCmin en función del TCmax, con las condiciones de los casos anteriores similares.

Finalmente, en la Fig. 10 se representa el rendimiento del sistema en porciento del ancho de banda en función de la longitud del campo de datos para distintos n.

EVALUACION COMPARATIVA

En el caso del protocolo Barra con Ficha/disciplina Rueda Cíclica (BF/RC), el rendimiento de la red en función de la longitud del campo de datos, medida en términos porcentuales del ancho de banda, es independiente del número de estaciones (Fig. 3). Crece rápidamente y a los 600 bytes se encuentra ya prácticamente en el 90%, superando el 95% para 1.500 bytes.

Con el mismo protocolo y la disciplina Menor Tiempo hasta Crisis (BF/MTC), el rendimiento decae considerablemente (Fig. 5) debido al gasto general que impone la ronda de reserva; el tiempo que ella demanda es una función lineal del número de estaciones y disminuye el rendimiento interranura de modo que el rendimiento total disminuye a medida que n aumenta. Para 1.500 bytes, por ejemplo, el rendimiento cae al 53% para n=10 y apenas alcanza el 10% para n=100.

No hay duda, en consecuencia, que la disciplina Rueda Cíclica induce un rendimiento mejor que la de Menor Tiempo hasta Crisis. De hecho, si se cumple (1) es obviamente innecesario complicar el algoritmo de acceso al medio introduciendo la ronda de reserva característica de la disciplina MTC. Sin embargo, basta que una estación tenga un tiempo de crisis menor que n Tr para que la aplicación de RC sea imposible. MTC, en cambio, permite bajar el TCmin a condición de que las otras estaciones tengan tiempos de crisis que hagan cumplir (3). Como puede verse en la Fig. 4, para n = 50 y una longitud de 1.500 bytes en el campo de datos, el TCmin en RC es de 60,2 ms; en MTC puede ser bajado a menos de 27 ms en una estación siempre y cuando todas las restantes tengan tiempos de crisis de 400 ms.

En el caso del protocolo Anillo con Ficha/disciplina Rueda Cíclica (AF/RC), la Fig. 6 muestra que el rendimiento es muy poco dependiente del número de estaciones. Para una longitud del campo de datos de 350 bytes, todas las curvas se encuentran ya por encima del 90%; para 1500 bytes, el rendimiento es superior al 97%.

Cuando se estudia el mismo protocolo con la disciplina MTC, el rendimiento, siempre en % del ancho de banda, desciende considerablemente y, como era de esperar, se hace muy dependiente del número de estaciones. Como en el protocolo anterior, esto es atribuible a la ronda de reservas -como se vió, una suerte de gasto general que disminuye el rendimiento inter-ranura que es una función lineal del número de estaciones.

De todas maneras, el TCmin puede estar por debajo del permitido por la disciplina RC si los restantes nodos tienen tiempos de crisis lo suficientemente grandes como para que se cumpla (3), como lo muestra la Fig. 7 para B=1.500 y distintos n.

Debe notarse que en términos absolutos y a igualdad de las demás condiciones, BF permite tiempos de crisis menores que AF, debido fundamentalmente a la diferencia de anchos de banda (10 Mb/s y 4 Mb/s, respectivamente).

En la Fig.11 se muestra la producción P, en Mb/s, de los protocolos BF y AF, con la disciplina RC. Aunque medido en % del ancho de banda el protocolo AF es ligeramente superior, la producción en Mb/s de BF es casi el doble debido a la diferencia de sus anchos de banda.

Las Figs. 12 a 15 muestran la producción de los tres protocolos en función de la longitud del campo de datos para n=10, 20, 50 y 100, respectivamente. En todos los casos, aunque el protocolo AF tiene mejor producción que BF para valores bajos del campo de datos, a partir de un cierto valor del mismo se ubica por debajo de los otros dos protocolos. Esto es la penalidad que paga por el limitado ancho de banda.

Debe tenerse en cuenta que las curvas halladas representan cotas superiores y que tanto los rendimientos como las producciones experimentarán disminuciones al bajar el rendimiento interranura por mantenimiento, fallas, etc.

Referencias

[1] H.Hellerman, "Digital Computer System Principles", McGraw-Hill Book Co., N. York, 1967, pp.230-233.

[2] J.Santos and I. Otero, "On priorities and transferences in computer systems and networks", Proc. Symposium on Computers and Automata, MRI Symposia Series, Vol. XXI, Brooklyn Polytechnic Press, N. York, 1971, pp. 265-275.

[3] J. F. Kurose, M. Schwartz and Y. Yemini, "Multiple-Access Protocols and Time-Constrained Communications", Computing Surveys, Vol. 16, Nr. 1, March 1984, pp. 43-70.

[4] L. Kleinrock and M. Scholl, "Packet Switching in Radio Channels: New Conflict-Free Multiple Access Schemes", IEEE Trans on Comm, Vol. Comm-28, Nr. 7, July 1980, pp. 1015-1029.

[5] Y. I. Gold and W. R. Franta, "An Efficient Collision-Free Protocol for Prioritized Access-Control of Cable or Radio Channels", Computer Networks, North Holland Publ. Co., Nr. 7, 1983, pp. 83-98.

[6] J. F. Kurose and M. Schwartz, "A Family of Window Protocols for Time-Constrained Applications in CSMA Networks", Proc. of the INFOCOM 83, San Diego, Calif., Apr. 18-21, 1983, pp. 405-413.

[7] J. F. Kurose M. Schwartz and Y. Yemini, "Controlling Time-Window Protocols for Time-Constrained Communications in Multiple Access Environments", Proc. 8thI International Data Communication Sysmposium. North-Falmouth, Mass., Oct. 3-6, 1983, pp.75-84.

[8] F. A. Tobagi, "Multiaccess Protocols in Packet Communication Systems", IEEE Trans on Comm, Vol. Comm-28, Nr. 4, april 1980, pp. 468-488.

[9] J. Santos, J.Orozco y O.Aliment. " Protocolos y Disciplinas de Prioridades en Ambientes Críticos", Lab. Sistemas Digitales, Dep. Ing. Eléctrica, Universidad Nacional del Sur, Bahía Blanca, Dic. 1987.

[10] F. Borgonovo, L. Fratta, F. Tarini and P. Zini, "L-EXPRESS-NET: The Communications Subnetwork for C-Net Project", IEEE Trans on Comm., Vol. Comm-33, Nr. 7, July 1985.

[11] IEEE, "Local Area Networks, 802.3 Standard, Carrier Sense Multiple Access with Collision Detection", IEEE, N. York, 1985.

[12] IEEE, "Local Area Networks, 802.4 Standard, Token-Passing Bus Access Method", IEEE, N. York, 1985.

[13] IEEE,"Local Area Networks, 802.5 Standard, Token Ring Access Method", IEEE, N. York, 1985.

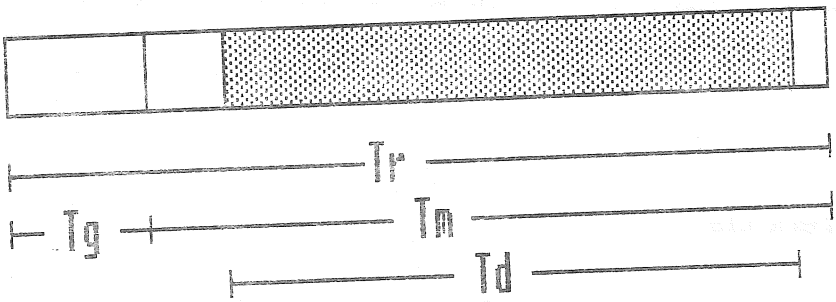


Fig. 1. Tr, tiempo de ranura; Tg, tiempo de "gasto general"; Tm, tiempo de mensaje; Td, tiempo del campo de datos.

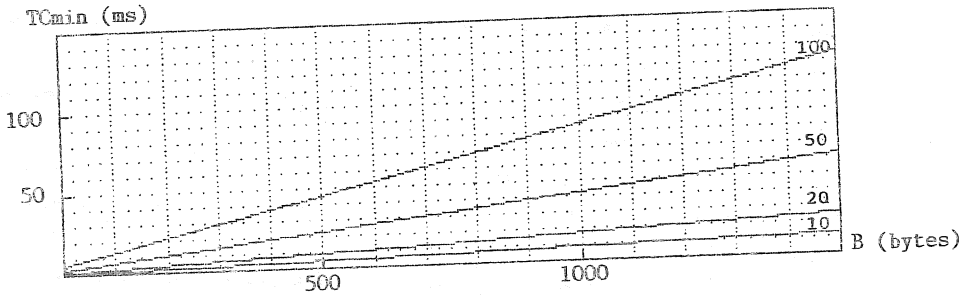


Fig. 2. BF/RC. Tiempo de crisis mínimo en función de la longitud del campo de datos.

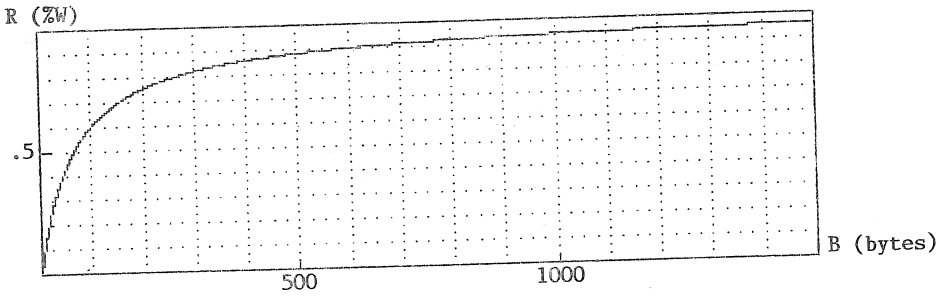


Fig. 3. BF/RC. Rendimiento del sistema, en % del ancho de banda, en función de la longitud del campo de datos, en condiciones de saturación.

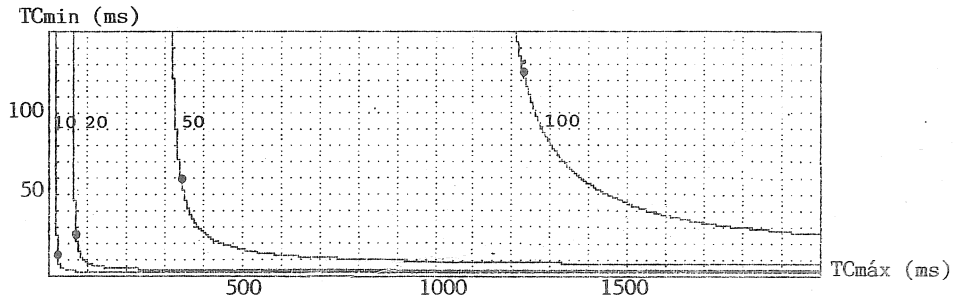


Fig. 4. BF/MTC. Tiempo de crisis mínimo en función del tiempo de crisis máximo, para n=10, 20, 50 y 100. En cada curva se marca el TCmin correspondiente a la prioridad RC.

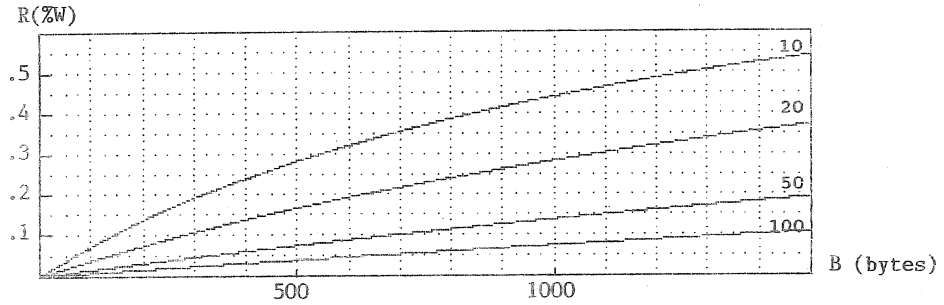


Fig. 5. BF/MTC. Rendimiento del sistema, en % del ancho de banda, en función de la longitud del campo de datos, para n=10, 20, 50 y 100, en condiciones de saturación.

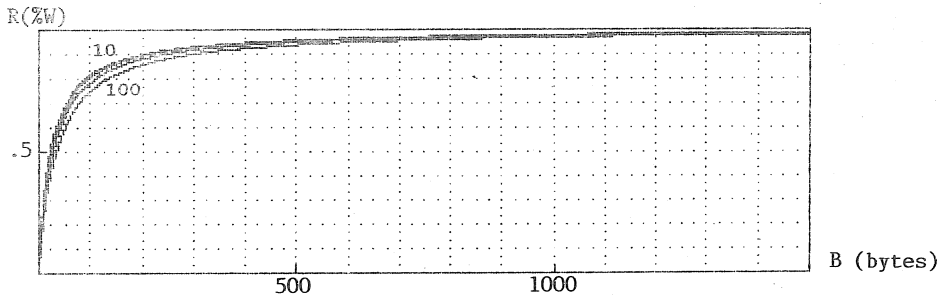


Fig. 6. AF/RC. Rendimiento del sistema, en % del ancho de banda, en función de la longitud del campo de datos, en condiciones de saturación. Las curvas para n=10, 20, 50 y 100, son prácticamente coincidentes.

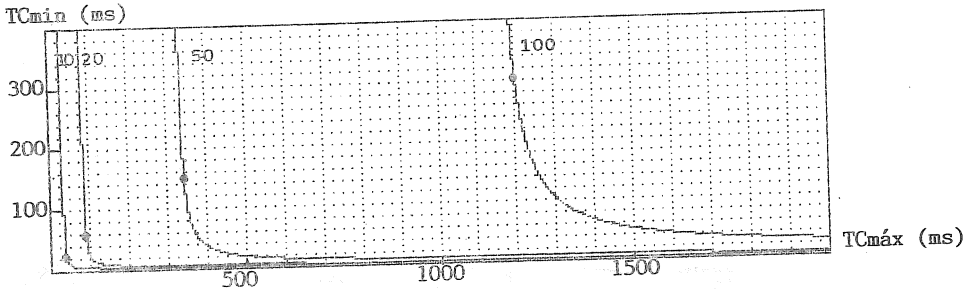


Fig. 7. AF/MTC. Tiempo de crisis mínimo en función del tiempo de crisis máximo, para n=10, 20, 50 y 100. En cada curva se representa en TCmin correspondiente a la disciplina RC.

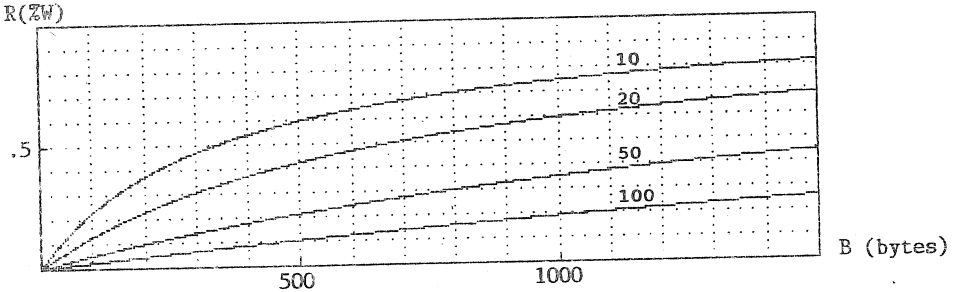


Fig. 8. AF/MTC. Rendimiento del sistema, en % del ancho de banda, en función de la longitud del campo de datos, para n=10, 20, 50 y 100, en condiciones de saturación.

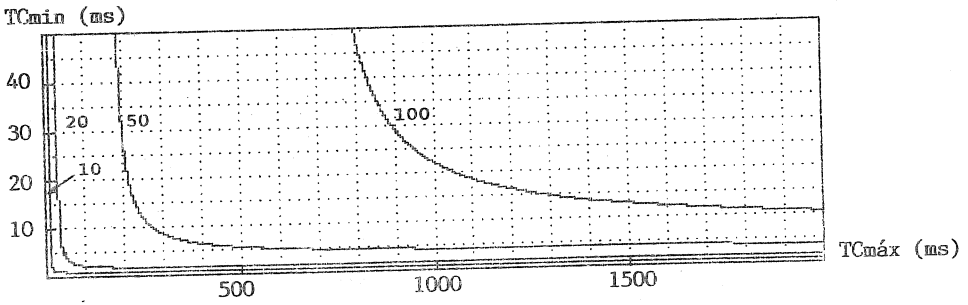


Fig. 9. OSMA/MTC. Tiempo de crisis mínimo en función del tiempo de crisis máximo para n=10, 20, 50 y 100.

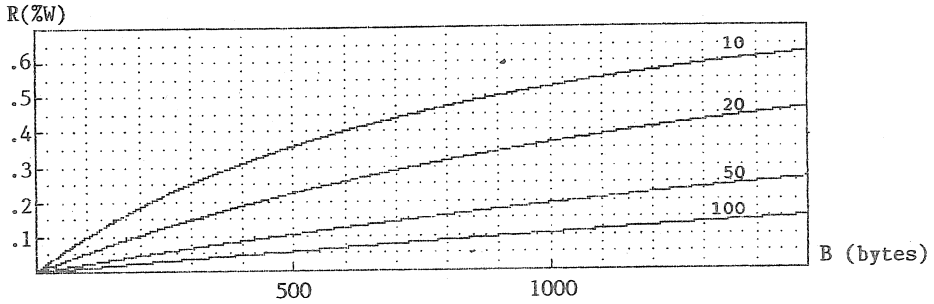


Fig. 10. CSMA/MTC. Rendimiento del sistema, en % del ancho de banda, en función de la longitud del campo de datos, para n=10, 20, 50 y 100.

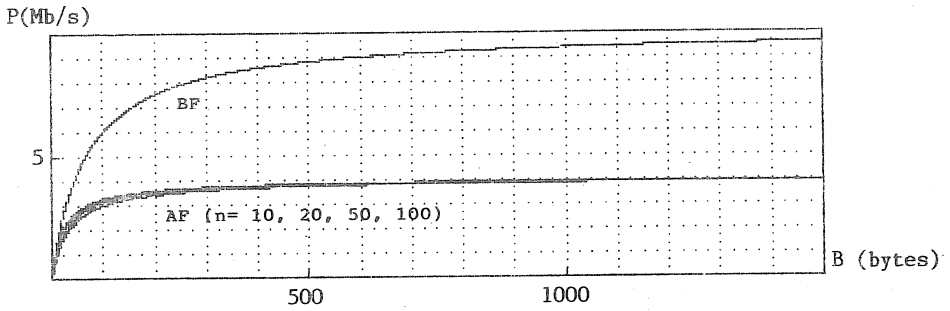


Fig. 11. Producción del sistema, en Mb/s, en función de la longitud del campo de datos, para los protocolos BF y AF, n=10, 20, 50 y 100.

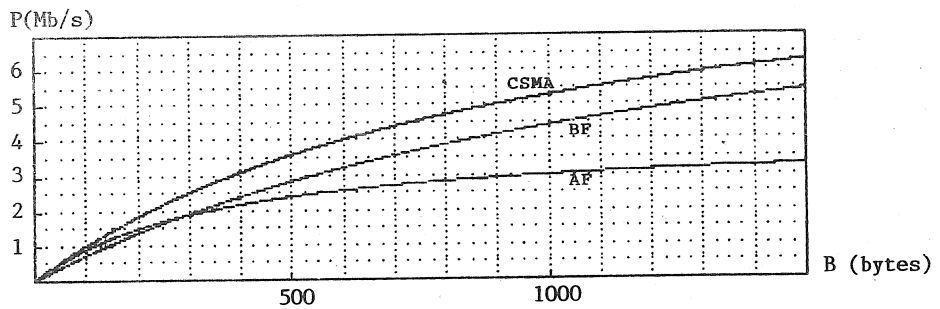


Fig. 12. Producción del sistema, en Mb/s, en función de la longitud del campo de datos, para CSMA, BF y AF, n=10.

AMIGAT
AGITAMIGA

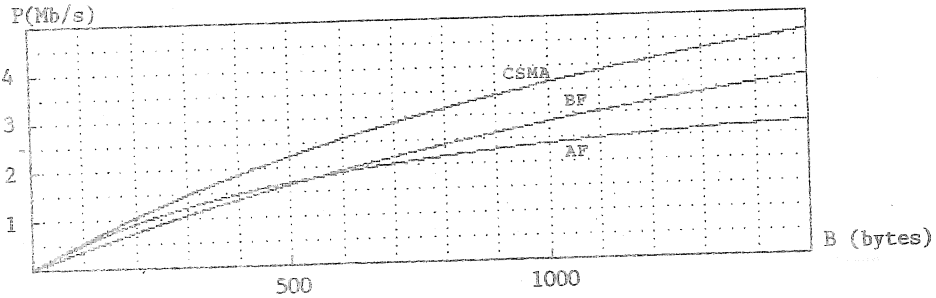


Fig. 13. Producción del sistema, en Mb/s, en función de la longitud del campo de datos, para CSMA, BF, AF, n=20.

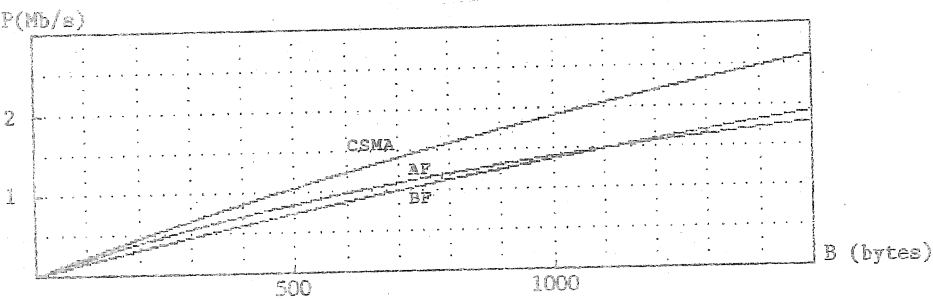


Fig. 14. Producción del sistema, en Mb/s, en función de la longitud del campo de datos, para CSMA, BF, AF, n=50.

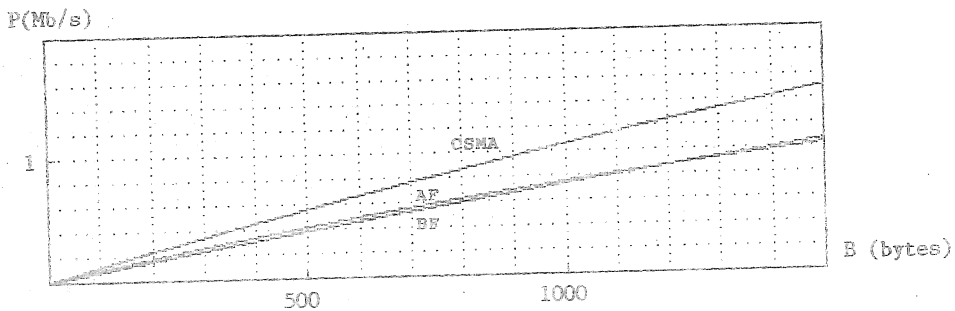


Fig. 15. Producción del sistema, en Mb/s, en función de la longitud del campo de datos, para CSMA, BF, AF, n=100.

PEDECIBA
INFORMATICA

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

UMA ANÁLISE DO SERVIÇO QUARENTENA DE DADOS LOCAL COM MODO DE DIÁLOGO SEMI-DUPLEX

Maria Izabel Cavalcanti Cabral

Departamento de Sistemas e Computaçao

e

Grupo de Redes de Computadores

Universidsade Federal da Paraíba

Rua Aprígio Veloso, s/n - Campina Grande - Paraíba

CEP: 58.100 - Brasil

Telefone: (083)321-7222 (Ramal:510) - Telex: 833191

RESUMO

Apresenta-se a modelagem do protocolo da camada de sessão considerando os serviços gerência de interação com modo de diálogo semi-duplex e quarentena de dados com a opção local.

Investiga-se a participação dos atrasos médios introduzidos por esses serviços na medida de desempenho atraso médio fim-a-fim do protocolo de sessão. Investiga-se também os efeitos da participação conjunta desses serviços na medida de desempenho referenciada.

1. INTRODUÇÃO

A camada de sessão fornece os meios necessários para que dois Usuários do Serviço de Sessão (Usuários SS), organizem e sincronizem a sua troca de dados [1]. No modelo RM/OSI, usuários SS correspondem às entidades de apresentação pares que solicitam serviços à camada de sessão. Essa camada, por sua vez, pode utilizar os serviços da camada de transporte para atender a esses usuários.

Estudos em modelagem e avaliação de desempenho do protocolo de sessão permitem o conhecimento de informações efetivas sobre a participação desse protocolo, e consequentemente a participação dos serviços oferecidos pela camada de sessão no desempenho de uma rede de computadores. Contribuições em modelagem do protocolo de sessão considerem essencialmente os serviços gerência de interação e quarentena de dados [2-3].

O serviço gerência de interação (ou controle de diálogo), permite aos usuários SS controlarem explicitamente quem deve, num dado momento, executar certas funções de controle. Os seguintes tipos de interação (ou modos de diálogos) são definidos:

.Interação bidirecional simultânea (ou diálogo duplex): Unidades de Dados do Serviço de Sessão (UDSSs) podem ser enviadas simultaneamente entre usuários SS pares;

.Interação bidirecional alternada (ou diálogo semi-duplex): somente o usuário SS com a posse da ficha de dados pode enviar UDSSs. A ficha de dados é um atributo de uma conexão de sessão.

É dinamicamente atribuída a um dos usuários SS e permite que o mesmo tenha o direito de transferir dados normais; e,

. interação unidirecional (ou diálogo simplex): somente o usuário SS inicialmente com a posse da ficha de dados pode enviar UDSSs.

O serviço quarentena de dados da camada de sessão, proposto pelo NBS [4], uma variante do serviço sincronização de sessão da ISO [5], permite a uma entidade de apresentação submeter unidades de dados de comprimento variável, denominadas Unidades de Quarentena de Dados (UQDs), que podem ser armazenadas na camada de sessão e fornecidas como um bloco de dados único à entidade de apresentação par. Conforme as UQDs sejam compostas na entidade de sessão de referência ou par, tem-se respectivamente, quarentena de dados local ou quarentena de dados remota.

No serviço gerência de interação com modo de diálogo semi-duplex, UDSSs que chegam a uma entidade de sessão de referência podem ficar bloqueadas aguardando a recepção da ficha de dados para serem liberadas à entidade de sessão par. Denomina-se atraso médio de admissão ao tempo médio de bloqueio dessas UDSSs na entidade de sessão de referência.

O serviço quarentena de dados introduz atraso na liberação de Unidades de Dados do Protocolo de Sessão (UDPSs) de uma entidade de sessão devido ao tempo necessário ao recebimento do número de UDSSs, na composição de UQDs. Esse atraso médio, denominado atraso médio de quarentena, é uma medida de desempenho que juntamente com a medida atraso médio de admissão,

introduzida pelo serviço gerência de interação com o modo de diálogo semi-duplex, devem ser investigadas no desempenho do protocolo de sessão com os serviços referenciados.

O presente trabalho apresenta a modelagem do protocolo de sessão com os serviços gerência de interação com o modo de diálogo semi-duplex, e quarentena de dados com a opção local. O modelo considerado, doravante denominado quarentena local semi-duplex, direciona-se às aplicações que envolvem atomicidade transacional, isto é, transações que devem ser enviadas em um bloco único de dados. O serviço telecompras é um exemplo de uma dessas aplicações.

As medidas de desempenho de interesse são: atraso médio de admissão, atraso médio de quarentena e atraso médio fim-a-fim. A medida de desempenho atraso médio fim-a-fim do protocolo de sessão corresponde ao atraso médio de admissão e ao atraso médio de quarentena adicionados ao tempo médio de transmissão de UDPSs na conexão de sessão.

Esse trabalho investiga a participação das medidas atraso médio de admissão e atraso médio de quarentena na medida atraso médio fim-a-fim do protocolo de sessão. Investiga também, os efeitos da participação conjunta desses serviços no modelo referenciado.

O restante desse trabalho é organizado da seguinte forma: as seções 2 e 3 apresentam respectivamente a descrição e a solução do modelo quarentena local semi-duplex; a seção 4 apresenta um exemplo numérico para esse modelo, e, finalmente, a seção 5

apresenta as conclusões do presente trabalho.

2. O MODELO QUARENTENA LOCAL SEMI-DUPLEX

O modelo quarentena local semi-duplex considera uma conexão de sessão na fase de transferência de dados com os serviços gerência de interação com o modo de diálogo semi-duplex, e quarentena de dados com a opção local.

A figura 1 apresenta o modelo quarentena local semi-duplex. Nesse modelo, UDSSs chegam à entidade de sessão de referência com tempos de interchegadas distribuídos exponencialmente com média λ UDSSs por segundo. Caso essa entidade de sessão não possua a ficha de dados, esta pode bloquear UDPSs até a recepção dessa ficha que permitirá a liberação de UDPSs na conexão de sessão. Essa liberação de UDPSs se processa conforme uma disciplina de liberação de UDSSs adotada. Contudo, se a ficha de dados chega a esse lado e a quantidade de UDPSs armazenadas é menor que o comprimento de uma UQD, esta ficha fica a espera da chegada de UDSSs, que serão associadas a UDPSs, a fim de completarem a UQD desejada. A partir daí, inicializa-se a liberação das UDPSs dessa unidade, juntamente com a ficha de dados.

Adota-se a disciplina de liberação de UDPSs do tipo limitada, isto é, até uma quantidade limitada de UDPSs que se encontram bloqueadas são liberadas com a chegada da ficha de dados, desde que esta quantidade componha uma ou mais UQDs. Essa disciplina, conforme definição dada, é função da participação conjunta dos serviços gerência de interação e quarentena de

dados.

A disciplina de liberação de UDPSs limitada engloba dois casos particulares de disciplinas: a não exaustiva e a comutada. A disciplina não exaustiva corresponde a liberação de UDPSs que compõem apenas uma UOD, e a disciplina comutada corresponde à liberação das UDPSs que compõem todas as UODs bloqueadas até a recepção da ficha de dados.

A entidade de sessão par à medida que recebe UDPSs entrega-as (como UDSSs) ao usuário SS receptor. A entrega da última UDSS a esse usuário permite a geração de uma UDSS de resposta (e/ou reconhecimento) na entidade de sessão par. Essa UDSS juntamente com a ficha de dados, são enviadas pela conexão de sessão à entidade de sessão de referência.

Uma Conexão de transporte é atribuída a uma conexão de sessão durante toda a fase de transferência de dados. Cada conexão de sessão é servida por uma conexão de transporte duplex que transmite as UDPSs entre entidades de sessão pares.

Modela-se uma conexão de transporte duplex por duas conexões de transporte simplex. Considera-se como modelo de uma conexão de transporte simplex um servidor exponencial com taxa média de serviço μ_1 ou μ_2 Unidades de Dados do Serviço de Transporte (UDSTs) por segundo (conforme esta conexão de transporte sirva a uma entidade de sessão de referência ou par respectivamente). A taxa média de serviço corresponde a taxa média de transmissão de UDSTs na conexão de transporte.

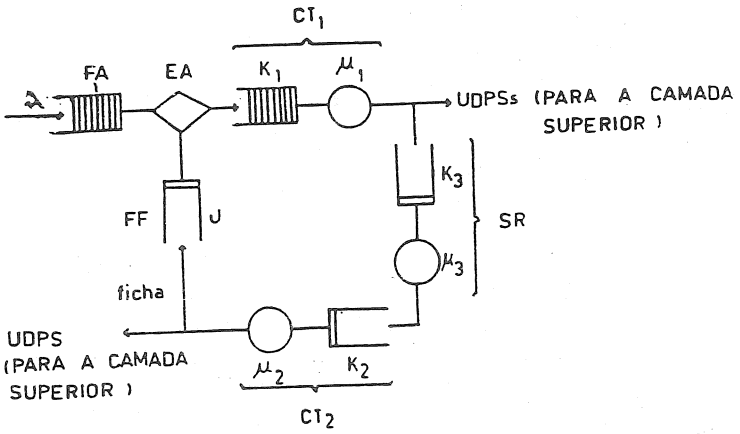


Figura 1: Modelo Quarentena Local Semi-Duplex.

Na figura 1, λ representa a taxa média de chegada de UDSSs à entidade de sessão de referência; FA representa a Fila de Admissão que contém UDPSs bloqueadas na entidade de sessão de referência em processo de quarentena ou como UQDs, aguardando a chegada da ficha de dados para serem liberadas; FF representa a Fila da Ficha de Dados que se apresenta vazia se a entidade de sessão de referência não possui esta ficha, e apresenta comprimento 1 (um), caso contrário; EA representa o Escalonador de Admissão que controla a emissão de UDPSs na conexão de sessão adotando a disciplina de liberação de UDPSs do tipo limitada; SR representa o Sistema de Resposta (ou o gerador de UDSSs à entidade de sessão par) que emite uma UDSS de resposta à entidade de sessão par, sendo ativado após esta entidade enviar à camada superior todas as UDPSs recebidas. Modela-se SR como um servidor exponencial com taxa média de serviço μ_3 UDSSs por segundo,

correspondendo ao serviço oferecido pela camada de sessão adicionado aos serviços das camadas superiores fornecidos à aplicação considerada, e, finalmente, CT (CT) representa a conexão de transporte simplex que serve à entidade de sessão de referência (par) com taxa média de transmissão μ_1 (μ_2) UDSTs por segundo.

O comprimento de uma UQD pode variar durante a troca de dados entre usuários SS pares. Esse comprimento deve respeitar o limite máximo negociado entre esses usuários durante a fase de estabelecimento da conexão da sessão. Contudo, para efeito de simplificação do modelo abordado, considera-se que uma UQD corresponde a um número fixo de UDPSs. Considera-se também, as seguintes suposições: independência estatística nos processos de chegada e de serviço, disciplinas de filas do tipo "First Come First Served" (FCFS) [6], e correspondência uma a uma entre UDSSs, UDPSs e UDSTs.

A notação utilizada no presente trabalho é a seguinte:

λ : Taxa média de chegada de UDSSs na entidade de sessão de referência.

μ : Taxa média de transmissão de UDSTs na conexão de transporte.

N : Número máximo de UDPSs que a camada de sessão pode armazenar.

W : Número máximo de UDSTs que a conexão de transporte pode armazenar.

K : Comprimento em UDPSs de uma unidade de quarentena de dados.

Ia : Comprimento médio da fila de admissão.

Ic : Comprimento médio da fila da conexão de transporte.

At : Atraso médio de transmissão na conexão de transporte.

Aq : Atraso médio de quarentena.

Ad : Atraso médio de admissão.

Af : Atraso médio fim-a-fim.

Ad ^ Aq : Sobreposição de Ad e Aq.

3. SOLUÇÃO

O modelo quarentena local semi-duplex pertence a categoria de redes mistas, apresentando características que inviabilizam a procura de soluções sem restrições para esse modelo. Essas características são: bloqueio de UDPSs em FA e chegadas em lote em CT_1.

Adota-se para esse modelo uma solução usando Cadeias de Markov, conforme método de solução iterativa apresentado em [7]. Essa solução é exata, contudo apresenta a restrição de tornar-se inpraticável para um grande número de estados do modelo considerado.

O modelo quarentena local semi-duplex apresenta $N . (W + 2) + K + W + 2$ estados possíveis.

Seja $Pr[i,j,k_1,k_2,k_3]$ a probabilidade que esse modelo está no estado (i,j,k_1,k_2,k_3), isto é, há i, j e k_3 UDPSs em FA, FF, e SR, e k_1 e k_2 UDSTs em CT_1 e CT_2, respectivamente.

As variáveis de estados referenciadas têm os seguintes limites:

$0 <= i <= N$

$0 <= j <= 1$

$0 <= k1 <= W$

$0 <= k_2 <= 1$

$0 <= k_3 <= 1$

As seguintes restrições entre as variáveis de estados existem:

$i >= K \rightarrow j = 0$

$j > 0 \rightarrow i <= K$ e $k_1 = k_2 = k_3 = 0$

$k_1 > 0 \rightarrow k_2 = k_3 = 0$

$k_2 > 0 \rightarrow k_1 = k_3 = 0$

$k_3 > 0 \rightarrow k_1 = k_2 = 0$

$k_2 + k_3 <= 1$

As medidas de desempenho de interesse para esse modelo considerando os limites e as restrições acima mencionadas, são as seguintes:

$Ia = \sum_i i . Pr[i,j,k_1,k_2,k_3]$, para $1<=i<=N,\ j=0,\ 1<=k_1<=W,\ k_2=0,\ k_3=0;$

$1<=i<=N,\ j=0,\ k1=0,\ k2=1,\ k3=0;$

$1<=i<=N,\ j=0,\ k_1=0,\ k_2=0,\ k_3=1$

$$i<K, j=0,\ k_1=0,\ k_2=0,\ k_3=0.$$

$$Ic = \sum_{k_1} k_1 \cdot Pr[i,j,k_1,k_2,k_3],\ \text{para } 1<=k_1<=W,\ 0<=i<=N,\ j=0,\ k_2=0,\ k_3=0.$$

$$Ad \hat{} Aq = Ia / Va$$

$$At = Ic / Va$$

$$Af = (Ad \hat{} Aq) + At$$

4. EXEMPLO NUMÉRICO

Considera-se uma conexão de sessão com modo de diálogo semi-duplex e quarentena local. A geração de UDSSs na entidade de sessão de par é conforme distribuição exponencial com taxa média 20 UDSSs por segundo. A capacidade de armazenamento na entidade de sessão de referência é de 6 UDPSs. Adota-se a disciplina de liberação de UDPSs do tipo comutada. A taxa média de transmissão na conexão de transporte que serve a essa conexão de sessão é de 10 UDSTs por segundo. A capacidade de armazenamento na conexão de transporte é de 6 UDSTs.

Para esse exemplo, tem-se: N = 6 UDPSs, μ_3 = 20 UDSSs por segundo, $\mu_1 = \mu_2$ = 10 UDSTs por segundo, e W = 6 UDPSs (UDSTs).

A figura 2 apresenta as medidas de desempenho atraso médio de admissão sobreposta à medida atraso médio de quarentena em função da taxa média de chegada de UDSSs. A figura 3 apresenta a medida atraso médio fim a fim em função da referida taxa.

Na figura 2 para K = 1 as medidas sobrepostas atraso médio de admissão e atraso médio de quarentena apresentam um

comportamento típico, isto é, aumentam com a taxa média de chegada de UDSSs. Para K > 1, essa medida apresenta inicialmente um comportamento atípico, sendo muito significativa para os valores menores dessa taxa, e diminuindo com o aumento da referida taxa até um valor limite, quando então volta a apresentar o comportamento típico mencionado, tendendo a um valor de saturação que depende do valor atribuído a K.

Os efeitos da participação conjunta dos serviços gerência de interação e quarentena de dados estão presentes nas medidas de desempenho apresentadas. A interseção total dessas medidas para K = 1,2,3 e 6 na figura 2, significa que a medida atraso médio de quarentena é menor ou igual a medida atraso médio de admissão, isto é, quando a ficha de dados chega à entidade de sessão de referência, esta entidade tem completado a sua capacidade de armazenamento de UDPSs, podendo liberar as UQDs formadas. Nesse caso, todas as UDPSs bloqueadas formam UQDs, não ficando portanto, UDPSs bloqueadas nesta entidade. Essa é uma situação particular em que o valor de K é um divisor da capacidade de armazenamento de UDPSs da entidade de sessão de referência.

Se o valor de K não é um divisor da capacidade de armazenamento de UDPSs da entidade de sessão de referência, quando a ficha de dados chega a essa entidade, e esta se encontra no limite de sua capacidade de armazenamento de UDPSs, as UQDs formadas são liberadas, ficando bloqueadas aquelas UDPSs que não formaram UQDs. Essas UDPSs somente poderão ser liberadas com a próxima chegada da ficha de dados, desde que essas unidades, agrupadas com outras UDPSs formem UQDs.

Essa situação ocorre no presente modelo nos casos em que K = 4 e K = 5. No primeiro caso tem-se o bloqueio de 2 UDPSs e no segundo caso, o bloqueio de 1 UDPS. Assim sendo, conforme mostra a figuras 2, para uma mesma taxa de chegada de UDSSs de referência, os maiores valores das medidas sobrepostas atraso médio de admissão e atraso médio de quarentena, ocorrem no caso em que k = 4, seguido daquele em que K = 5.

Denomina-se tempo médio de interrupção de quarentena, o tempo médio de bloqueio de UDPSs que não formaram UQDs devido à limitação de espaço de armazenamento na entidade de sessão de referência. Esse tempo é somado a medida atraso médio de quarentena significando uma interrupção no processo de quarentena local.

Na figura 2, a partir da interseção total das medidas sobrepostas atraso médio de admissão e atraso médio de quarentena para K = 1, 2, 3 e 6, pode-se obter o tempo médio de interrupção de quarentena para K = 4 (K = 5), como a diferença entre o valor dessas medidas para K = 4 (K = 5), com esse valor para K = 1 (2, 3 ou 6), considerando uma mesma taxa média de chegada de UDSSs de referência.

Para taxas médias de chegadas de UDSSs mais altas, a medida atraso médio fim-a-fim apresenta valores muito próximos para os valores atribuídos a K. Essa proximidade deve-se às medidas que compõem a medida atraso médio fim-a-fim, que assumem comportamentos opostos com o aumento dessas taxas. As medidas sobrepostas atraso médio de admissão e atraso médio de quarentena

apresentam valores maiores conforme a ordem de valores de K sejam respectivamente 4,5 e 6 (1, 2 ou 3), e a medida atraso médio de transmissão na conexão de transporte apresenta valores menores para essa ordem de valores mencionada.

O comportamento típico das medidas sobrepostas atraso médio de admissão e atraso médio de quarentena e também da medida atraso médio fim-a-fim, para K = 1, era esperado uma vez que para este valor é nula a participação da medida atraso médio de quarentena nas medidas mencionadas.

A figura 4 mostra a participação das medidas sobrepostas atraso médio de admissão e atraso médio de quarentena como significativa na medida atraso médio fim-a-fim da conexão de sessão. Nessa figura, pode-se obter a participação da medida atraso médio de transmissão da conexão de transporte, subtraindo da participação total dessas medidas, a participação das medidas sobrepostas referenciadas.

Conforme mostra a figura 4, para uma mesma taxa média de chegada de UDSSs de referência, por exemplo, 42 UDSSs por segundo, essa figura mostra que a menor participação referenciada é de 65,0 %, para K = 6 (extensivo para K = 1,2 e 3), e que a maior participação é de 75,0 %, para K = 4 . A diferença entre esses valores representa a participação do tempo médio de interrupção de quarentena para K = 4.

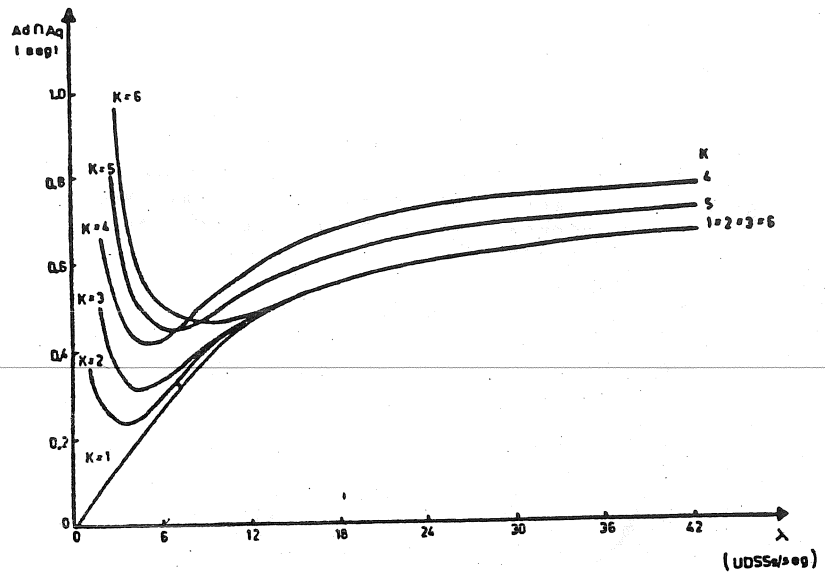


Figura 2: Atraso Médio de Admissão e Atraso Médio de Quarentena vs Taxa Média de Chegada de UDSSs para o Modelo Quarentena Local Semi-Duplex.

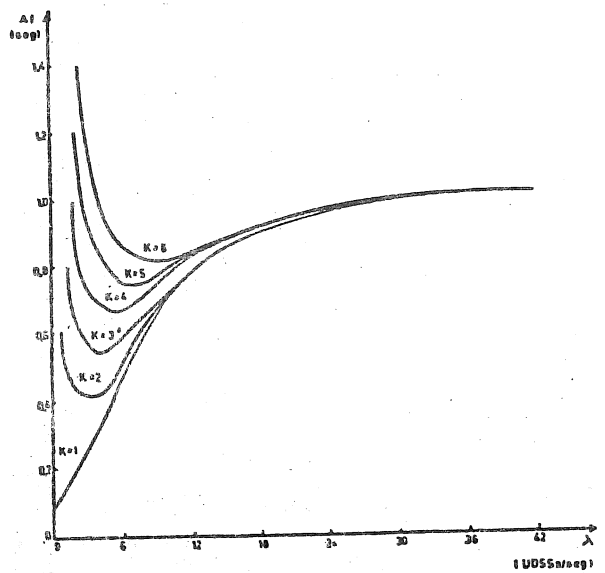


Figura 3: Atraso Médio Fim-A-Fim vs Taxa Média de Chegada para o Modelo Quarentena Local Semi-Duplex.

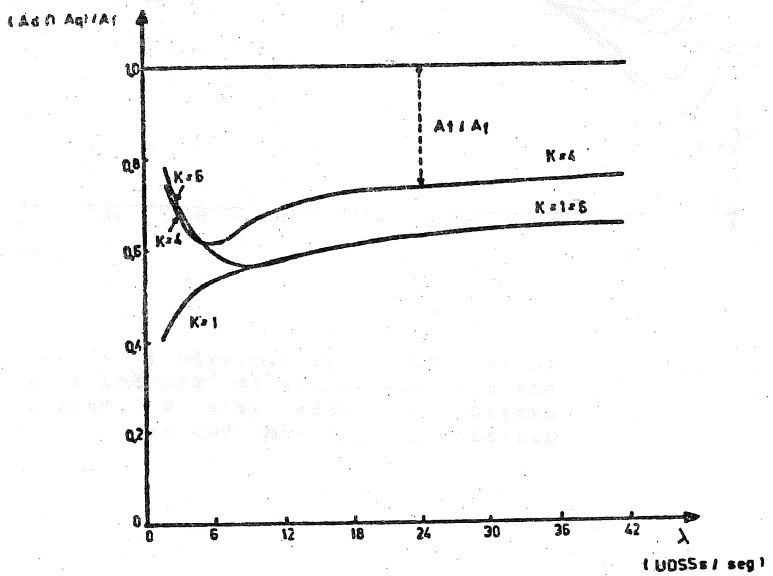


Figura 4: Participação das Medidas Atraso Médio de Admissão e Atraso Médio de Quarentena na Medida atraso Médio Fim-A-Fim para o Modelo Quarentena Local Semi-Duplex.

5. CONCLUSÕES

Os serviços gerência de interação com modo de diálogo semi-duplex e quarentena de dados com a opção local introduzem as medidas atraso médio de admissão e atraso médio de quarentena na medida atraso médio fim-a-fim do protocolo de sessão.

No modelo considerado, denominado quarentena local semi-duplex, os efeitos da participação conjunta dos serviços gerência de interação e quarentena de dados foram detectados: inicialmente na definição da disciplina de liberação de UDPSs considerada do tipo limitada, e, finalmente, na obtenção das medidas de desempenho atraso médio de admissão e quarentena de dados, que se apresentam sobrepostas devido a essa participação conjunta de serviços.

A participação das medidas atraso médio de admissão e atraso médio de quarentena, conforme exemplo numérico apresentado neste trabalho, apresentam percentuais significativos na composição da medida atraso médio fim-a-fim do protocolo de sessão. Esses percentuais evidenciam a necessidade de avaliar o desempenho do protocolo de sessão nos projetos de redes de computadores com aplicações que solicitem os serviços de sessão referenciados.

6. REFERENCIAS BIBLIOGRÁFICAS

[1] Emmons, W.F.. & Chandler, A.S.: OSI Session Layer: Services and Protocols, Proc. of the ItEE, Vol.71, No.12, December

1985, 1387-1400.

[2] Cabral, M.I.C. & Sauvé, J.P.. Modelling of Dialogue Management in the Session Layer, IEEE Global Telecommunications Conference, Vol. 2, Houston, Texas, E.U.A. , December 1986, p. 22.6.1-22.6.4.

[3] Cabral, M.I.C. & Sauvé, J.P.: Modelagem do Serviço Quarentena de Dados Local, Anais do XIV SEMISH, Salvador, Ba, Julho de 1987 p. 581-589.

[4] Nielsen, F.H. & Heafner,J.F. Description of a Planned Federal Information Processing Standard for the Session Protocol, Proc. COMPCON, Spring 82 (IEEE Catalog No. 82 CH 1739-2), february 1982, p. 272-278.

[5] DP8327 Open Systems Interconnection Basic Connection Oriented Session Protocol Specification, Doc. ISO/TC 97/SC 16N1493, march 1987.

[6] Moura, J.A.B. et alli: Redes Locais de Computadores - Protocolos de Alto Nível e Avaliação de Desempenho McGraw-Hill Ltda., 1986.

[7] Sauer, C.H. & Chandy, K.M.: Computer Systems Performance Modeling, Prentice-Hall, Inc., Englewood Cliffs, N.J.,1981.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa.
Buenos Aires, Setiembre 1988.

PROTOCOLO DE COMUNICACION

ENTRE HOST Y FRONT-END PROCESSOR

Definición

Diseño

Implementación

Autores:

Ing. José Mechoulam

Ing. Marcela Adriana Pérez

Institución:

UNIVERSIDAD DE BUENOS AIRES - FACULTAD DE INGENIERIA
DEPARTAMENTO DE ELECTRONICA
LABORATORIO DE ELECTRONICA

Domicilio:

Av. Paseo Colón 850 - (1063) Capital Federal
Repúblico Argentina.
T.E.: 34-1441 Int.: 215

PROTOCOLO DE COMUNICACION

ENTRE HOST Y FRONT-END PROCESSOR

RESUMEN:

El objetivo de este desarrollo es la definición, diseño e implementación del Procedimiento de conexión entre un computador principal y el controlador de comunicaciones, el que deberán ajustarse ambos extremos del enlace para transferir información.

El presente trabajo es parte de un Proyecto de Trasmisión de Datos promovido por la S.E.C.Y.T. y consiste en la interconexión de computadoras de Centros Educativos y de Investigación a fin de compartir sus recursos. Dichas computadoras se interconectarán en red, utilizando ARPAC como medio físico de trasmisión. Se adopta la utilización de un controlador de comunicaciones cuya función es liberar al host de las tareas de comunicación.

El Protocolo diseñado se estructura en tres niveles según las recomendaciones de las normas ISO y los datos se estructuran en paquetes de longitud variable con un máximo de 128 bytes.

- Primer Nivel: está definido por la norma RS-232C, adoptado con el criterio de utilizar modos de trasmisión standard tanto en el host como en el controlador de comunicaciones.

- Segundo Nivel: su función es administrar y controlar el flujo de datos en el canal de comunicación, e implementar el control de errores del canal de trasmisión. Se provee además transparencia a la información y retrasmisión automática de paquetes erróneos.

- Tercer Nivel: implementa y administra múltiples canales lógicos sobre un único medio físico de trasmisión.

I- INTRODUCCION

I-1. Antecedentes

El presente trabajo forma parte del proyecto sobre Sistemas de Trasmisión de Datos, promovido por la Secretaría de Informática.

Específicamente consiste en la interconexión de todos los computadores pertenecientes a los Centros Educativos y de Investigación con el fin de compartir los recursos que estos poseen en el área de software. El medio físico que se utiliza como vínculo para la interconexión de las computadoras es la Red Pública de Trasmisión de Datos ARPAC.

I-2 Finalidad

La implementación del presente trabajo tiende a satisfacer dos objetivos básicos:

a) la necesidad de interconectar en la Red computadoras heterogéneos

b) relevar a los computadores principales de las tareas relacionadas con el mantenimiento de las comunicaciones

El análisis de las soluciones posibles que satisfacen los objetivos enunciados lleva a la decisión de adoptar el uso de un front-end processor. En la primera etapa del desarrollo del presente trabajo el front-end processor se desempeñará como controlador de comunicaciones, haciéndose cargo de todo lo inherente a las tareas de mantenimiento de la Red y las comunicaciones. En la segunda etapa de la implementación se adicionará al front-end processor las funciones de convertidor de protocolos con el fin de hacer transparente la Red al computador principal, facilitanto por lo tanto la conexión de computadores de distintas marcas a la red.

El front-end processor en la primera etapa tiene dos interfaces de comunicación con el mundo exterior, una que lo comunica con la red Pública ARPAC y la otra que lo comunica con el computador principal.

La finalidad del trabajo que se detallará a continuación es la implementación de la interfaz de comunicación con el computador principal (host).

I-3 Consideraciones generales sobre la interfaz controlador-host

Esta interfaz como ya lo hemos expresado es una parte circuital (hardware) , una parte consistente en el protocolo de comunicaciones (software).

En lo relacionado con el hardware se adopta el criterio de utilizar un modo de transmisión que posee standard tanto para el host como para el controlador de comunicaciones. El serial que cumple con los requisitos especificados es el RS-232 C disponible en forma serial en ambos extremos de la conexión.

El cuerpo del protocolo de comunicaciones su objetivo es lograr una transferencia de datos inteligente, fluida y libre de errores. Además este debe caracterizarse por su flexibilidad para implementar y soportar nuevas funciones en la medida en que las exigencias operacionales así lo requieran, sin la necesidad de modificar la estructura existente.

Para satisfacer los requerimientos mencionados se adopta para el protocolo una estructura de niveles según las recomendaciones de la I.S.O.. Para facilitar las comunicaciones con el protocolo de la interfaz de salida X-25 se adopta una configuración equivalente al mismo estructurándolo en 3 niveles.
Cada nivel implementa una série de funciones propias para su desenvolvimiento a la vez que brinda una cantidad de servicios que son utilizados por el nivel superior.

1.- El primer nivel es el físico y como se mencionó anteriormente está definido por la norma RS-232 C que determina las condiciones físicas, tipo de conector, niveles de tensión y señales, que deben cumplirse para lograr la transferencia de información.

2.- El segundo nivel es el que tiene a su cargo la administración y control de flujo de datos a través del canal y la implementación del control de errores para convertir el ruidoso medio físico utilizado en un canal libre de errores.

3.- El tercer nivel tiene la función de implementar y administrar varios canales lógicos sobre el mismo canal físico de trasmisión con el propósito de permitir que el computador principal se pueda comunicar con mas de un destino (host de la Red a la vez). En esta implementación se prevé la posibilidad de contar con hasta 8 canales lógicos.

Los datos se trasmiten en estructura de paquete de longitud variable con un máximo de 128 bytes (coincidente con la norma X-25). A los efectos de la administración y regulación del flujo de datos de los distintos canales este nivel está dotado de una serie de comandos y direccionamientos.

II- PRIMER NIVEL

El primer nivel denominado físico es el que debe proveer las funciones mecánicas, eléctricas, funcionales y procedurales para establecer, mantener y desconectar el enlace físico entre dos equipos de datos posibilitando la trasmisión de un flujo trasparente de bits entre los equipos.

Por la definición surge que este nivel está caracterizado por cuatro parámetros fundamentales:

- mecánico
- eléctrico
- funcional
- procedural

Existen muchas normas que definen claramente estos parámetros, en nuestro caso en función de las premisas establecidas se adoptó la norma RS-232 C.

En relación a las características principales esta norma establece:

1.- Características mecánicas:

Estas características se refieren a la interfaz mecánica y específicamente al conector utilizado y a la asignación de los contactos del mismo a las señales de intercambio. La norma RS-232 C establece el conector definido por la I.S.O. 2110 que es un conector de 25 contactos con la asignación de los mismos de acuerdo a las especificaciones de la norma.

2.- Características eléctricas:

Lógica de salida del driver c/ 3 K a 7 K de carga	-15V > Voh > 5V
Tensión de salida del driver a circuito abierto	[Vo] < 25V
Corriente de salida de cortocircuito	[Io] < 0,5 A
Impedancia de entrada del receptor	7 Kohm < Rin < 3 Kohm
Tensión de entrada receptor	+ 15 V - 15 V
Salida del receptor a circuito abierto	Marca
Salida del receptor con + 3V entrada	Espacio
Salida del receptor con - 3V entrada	Marca

+ 15 V
+ 5 V

" 0 " lógico= espacio

- 5 V

" 1 " lógico= marca

3.- Características funcionales:

Las funciones de los circuitos de intercambio se clasifican en las siguientes categorías: datos, control, temporización y tierras.

La norma RS-232 C adopta los circuitos de intercambio de la norma V 24.

4.- Características procedurales:

Definen el conjunto de procedimientos que utiliza el nivel físico para poner en funcionamiento los circuitos de intercambio. Estos son procedimientos que deben ser ejecutados para habilitar la trasmisión de bits con el fin que las funciones del nivel superior puedan tener lugar. La norma RS-232 C adopta las recomendaciones de la V 24 para circuitos de intercambio DTE/DCE.

Como puede verse, este nivel tiene la función de compatibilizar los aspectos físicos, mecánicos y eléctricos estableciendo los stardard correspondientes para que todo equipo que se ajuste a ellos pueda conectarse sin inconvenientes. Además brinda al nivel superior los servicios de trasmisión de bits trasparentes para que éste pueda implementar las funciones que le corresponden.

III- SEGUNDO NIVEL

III-1 Consideraciones generales

Este nivel tiene a su cargo la implementación de las siguientes funciones:

1.- administración y control de la comunicación. Esto implica las siguientes subfunciones:

a. iniciación de la comunicación
b. mantenimiento de la comunicación
c. finalización de la comunicación

2.- administración y control del flujo de datos. Incluye la implementación de las siguientes subfunciones:

a. manejo de la trasmisión y recepción de datos y comandos
b. implementación del procedimiento de control de errores y recuperación de la información.

Para satisfacer los requerimientos necesarios para el cumplimiento de las funciones que tiene a su cargo este nivel, y teniendo como meta el propósito general de utilizar medios standard, se adopteron las siguientes características para la trasmisión de la información: modo de trasmisión full-dúplex a una velocidad de 9600 bits/seg en forma asincrónica.

III-2 Administración y control de la comunicación

El establecimiento y control del canal físico entre el host y su controlador de comunicaciones consta de tres fases, disponiéndose de comandos específicos para cada una de ellas.

a. Iniciación:

El establecimiento del canal físico o sea el inicio de la comunicación puede ser realizado únicamente por el host. Para tal fin se cuenta con un comando de pedido de conexión que es emitido por el host. La respuesta por parte del controlador de aceptación de la conexión deja establecido el vínculo físico de comunicación, el cual permanecerá activo a lo largo de todo el período de trasmisión.

b. Mantenimiento:

Una vez establecido el canal físico y a lo largo de todo el período activo se tiene flujo de información (datos y comandos) circulando por el mismo. Para poder mantener las comunicaciones y no perder información este nivel cuenta con una serie de registros para el almacenamiento de la misma tanto en trasmisión como recepción. En el caso de que alguno de estos registros de recepción colmara su capacidad se emite un comando de pedido de interrupción del envío de datos. En el momento en que los registros dispongan nuevamente de espacio se emite al otro extremo del enlace un comando de solicitud de reanudar la trasmisión. Mientras dura la comunicación se dispone de un comando de solicitud de averiguación del estado en que se encuentra el otro extremo.

c. Finalización:

El pedido de finalización de las comunicaciones puede ser efectuado por cualquiera de los extremos de la conexión. El motivo de finalización de la conexión puede ser el no estar dispuesto a seguir con el vínculo por no haber más comunicaciones previstas o pendientes, en el caso que lo solicite el host, o por causa de problemas en el vínculo físico que aconsejen tal actitud, y en este caso puede ser adoptada por ambos extremos. Una vez emitido el comando de finalización del vínculo físico y recibida la respuesta de notificación del comando, el canal físico queda interrumpido y solo puede ser restablecido por el host.

III-3 Administración y control del flujo de datos

La trasmisión de los datos se efectúa en paquetes cuya longitud puede ser variable, pero con un máximo de 128 bytes (para tener la equivalencia con la longitud del paquete de datos del protocolo X-25). Este paquete de datos que proviene del nivel superior se ve incrementado por el encabezamiento y delimitadores que lo agrega este nivel para tener bien definido cuál es el campo de los datos. Para hacer la comunicación segura y libre de errores se implementa el control de errores mediante el agregado de dos bytes de Control de Recundancia Cíclico al final de los datos. La configuración final del paquete se obtiene con el agregado del delimitador de final de paquete.

Para un efectivo control del flujo de datos el encabezamiento de cada paquete contiene el número de paquete que se está enviando y el número de paquete que se espera recibir. De esta manera si se recibe algún paquete fuera de secuencia, éste es rechazado.

La trasmisión de paquetes se efectúa de a un paquete por vez y luego se procede a esperar la confirmación de la recepción del mismo antes de enviar el siguiente: para tal fin este nivel cuenta con un comando de confirmación positiva o negativa del paquete recibido.

El mecanismo general del funcionamiento en recepción de este nivel consiste en, una vez recibido el paquete de datos, comprobar en primer término si el número del paquete recibido es el correspondiente, luego verificar si no hubo errores debidos a ruido en el canal mediante la implementación del polinomio de corrección y su comprobación con los bytes de CRC recibidos. En caso de que se verifique algún error se emite el comando de confirmación negativa de la recepción. La recuperación del paquete erróneo recibido se logra mediante la retrasmisión automática del paquete por parte del otro extremo. En caso de no haber errores se trasmite la confirmación positiva del mismo, hecho que habilita al trasmisor a emitir otro paquete.

En el caso de que se reciban paquetes erróneos en forma reiterada y superado un número preestablecido, el nivel tiene la facultad de pedir mediante un comando la desconexión del canal físico, avisando al nivel superior de la decisión adoptada. En este caso, el host luego de tomar las medidas correctivas correspondientes iniciará nuevamente el establecimiento del enlace.

La trasparencia de los datos se obtiene mediante la trasmisión de la longitud de los datos en el encabezamiento. De esta manera el nivel 2 tiene el pleno control de dónde comienzan los datos y cuál es su longitud.

Escapan a esta configuración de paquete solamente los comandos propios de este nivel que no se ajustan a este formato sino tienen una longitud de un byte de composición definida y están codificados de modo de tener una distancia 4, de manera que no pueda convertirse un comando en otro por problemas de ruido.

Lo visto hasta aquí se refiere a las funciones que implementa este nivel; en cuanto a los servicios que el mismo presta al nivel superior, ellos son los de trasmisión y recepción de información libre de errores.

Como se vio,el nivel 2 del Protocolo está conformado por:

1.- Comandos propios del Protocolo:

a) Comandos de establecimiento de la conexión
   CONEC= pedido de establecimiento de la comunicación
   ACONEC= aceptación del pedido de conexión

CARG= cargar el sistema operativo
DESCONEC= pedido de desconexión
ADESCONEX= aceptación del pedido de desconexión

b) Comandos de control del flujo de datos
CONFIR= confirmación de recepción correcta de un paquete

NOCONFIR= implica la no confirmación de un paquete recibido

CONFIRES= confirmación con pedido de espera, lo que implica que se suspenda el envío de paquetes de datos

PEDIN= pedido de información, enviado por cualquiera de los extremos para conocer el status del otro o para avisar que se puede reanudar el envío de datos

Estos comandos cumplen con la función del establecimiento y control del flujo de datos de la comunicación. Se trasmiten en forma directa no ajustándose a la estructura general de paquete que tienen los datos. Tienen una longitud de 1 byte y pueden enviarse en todo momento sin restricciones.

2.- Delimitadores:

Son usados en la estructura del paquete para separar las distintas partes que lo componen.

COEN= comienzo del encabezamiento

CODA= comienzo de los datos

FINTR= final de la trasmisión

FINDA= final de los datos.

Estructura del paquete de datos

El paquete se conforma de la siguiente manera:

COEN	ENC	CODA	DATOS	CRC	FINDA

donde: ENC es el encabezamiento con una longitud de 2 bytes, 1 byte destinado para identificar el número de paquete y 1 byte que contiene la longitud de los datos
CRC son 2 bytes de control de redundancia cíclico

IV- TERCER NIVEL

IV-1 Consideraciones generales

Como se vio en la introducción, la tarea principal de este nivel es implementar varios canales lógicos sobre un canal físico y administrar el flujo de información de cada uno de ellos. Para lograr este propósito es necesario definir e implementar varias funciones, pero es fundamental tener en cuenta que este nivel mantiene intercambio de información con dos entes diferentes. En primer lugar intercambia datos y comandos con su equivalente nivel 3 del otro extremo de la conexión (host), y por el otro lado intercambia información con el software de supervisión (supervisor) que hace de nexo con el nivel 3 del protocolo X-25 correspondiente a la otra interfaz de salida del controlador de comunicaciones.

Las funciones que debe configurar este nivel son:

1. gestionar la activación y desactivación del enlace X-25
2. implementar la iniciación y finalización de una comunicación con el host a través de un canal lógico
3. controlar el flujo de datos de todos los canales lógicos

IV-2 Gestión de la activación y desactivación del enlace X-25

Una vez activado el enlace entre el host y su controlador de comunicaciones, función que es privativa del nivel 2 del protocolo, el host puede si lo desea activar el enlace X-25 entre el controlador de comunicaciones y la red ARPAC.

Los comandos del nivel 3 correspondientes a la activación del enlace X-25 son:

AX25= petición de activación del enlace X-25

RX25= petición de conexión del enlace X-25

CFX25= confirmación de la activación del enlace X-25

DX25= petición de desactivación del enlace X-25

CFDX25= confirmación de la desactivación del enlace X-25

IDX25= informe de la caída del enlace X-25 por parte del controlador de comunicaciones

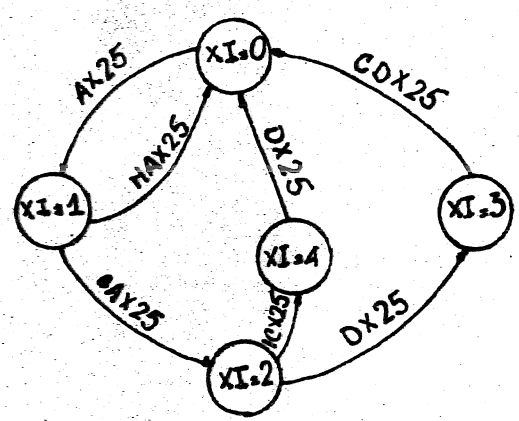
La activación del enlace X-25 la recibe el host enviando un comando de activar el enlace X-25 (AX25).

El nivel 3 recibe el comando y lo pasa al supervisor para ser comunicado al nivel 3 del protocolo X-25 para gestionar la activación. Mientras tanto el nivel 3 queda en estado de espera. Cuando la activación se efectiviza el supervisor se lo comunica al nivel 3, el cual emite la respuesta de CAX25 hacia el host. A partir de este momento puede comenzar el flujo de datos entre host y controlador, lo que implica que si el enlace X-25 no está activado no se aceptará el envío de datos.

La desactivación que también es privativa del host sigue una mecánica similar a la activación.

En caso de que, estando activado el enlace X-25, se produzca una caída del mismo, el controlador emitirá el comando ICX25 para notificar al host, el que deberá resolver el procedimiento a seguir.

El mecanismo de activación/desactivación está representado en el siguiente diagrama de estados.



El estado XI=0 corresponde a Enlace X-25 no activado, el estado XI=2 corresponde al estado de enlace X-25 activado y es el estado normal para que se puedan efectuar otras transacciones. Los estados XI=1, XI=3, XI=4 son estados intermedios de espera.

IV-3 Procedimiento de apertura y cierre de un canal lógico

Todo proceso de intercambio de datos entre el controlador de comunicaciones y el host se debe realizar a través de un canal lógico de comunicaciones. Con tal fin, el uso de dicho canal puede ser solicitado por cualquiera de los extremos de la comunicación. Las condiciones previas que se deben cumplir para que un proceso de intercambio de datos tenga lugar son:

a) canal físico host-controlador funcionando
b) enlace X-25 activado, lo que implica que en diagrama de estados mencionado anteriormente se esté en el estado XI=2

La gestión de pedido de un canal lógico se efectúa por medio del comando de pedido de canal (PCAN) que puede provenir tanto del host como del supervisor del controlador de comunicaciones.

El administrador de canales que se encuentra en el controlador y que oficia de receptor, consulta la tabla de estado de los distintos canales y procede a otorgar un canal al pedido recibido. Mientras dura el trámite de otorgamiento y confirmación el canal en cuestión pasa a estado de espera.
En el caso de que no haya ningún canal disponible para ser utilizado, el pedido formulado es rechazado y el ente que lo solicitó deberá esperar para reiterar la solicitud.

Los diferentes estados en que pueden encontrarse los canales lógicos son:

a. DI=1 lo que significa canal libre para ser utilizado
b. DI=2 canal en espera de algún trámite
c. DI=3 canal ocupado , es decir, asignado a algún proceso de intercambio de datos

En nuestro caso se disponen de 8 canales lógicos, o sea se pueden efectuar hasta 8 transferencias de datos entre el host y el controlador. Como cada transferencia de datos corresponde a una comunicación del host con otro similar por intermedio de la interfaz X-25 de salida esto significa que para utilizar eficientemente los 8 canales lógicos deberá disponerse de los correspondientes 8 canales lógicos de salida X-25 ARFAC.

El intercambio de comandos entre host y controlador no requiere del uso de un canal lógico.

IV-4 Control y trasmisión del flujo de datos

Este procedimiento consta del siguiente conjunto de funciones:

a- Función atención de tareas de recepción:
   1. atención de comandos
   2. atención de datos

b- Función atención de tareas de trasmisión:
   1. atención de comandos
   2. atención de datos

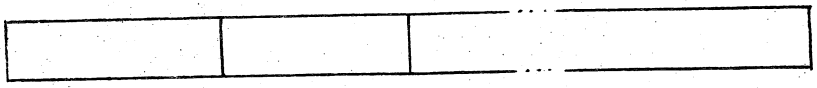
c- Función control de flujo de trasmisión y recepción de datos para cada canal

Como puede observarse este procedimiento es el núcleo del funcionamiento del Protocolo de Comunicaciones. Cada vez que le corresponde una intervención a este nivel, éste procede a efectuar la siguiente rutina:

lee un registro de recepción recibido por el nivel 2 y determina si su contenido es un comando o dato, actuando en consecuencia

luego dá lugar a las tareas de trasmisión, verificando si hay algún comando de respuesta que deba ser trasmitido y a continuación procede a trasmitir un registro de datos del canal lógico cuyo turno le toca en este momento dentro de la cola de espera.

La estructura general del paquete de este nivel es la siguiente:



byte 1 byte 2 bytes 3 al 128

byte #1:
b_0 - b_3= identificador de comando/dato
b_4 - b_7= número de canal

byte #2:
b_0 - b_3= número de último paquete aceptado
b_4 - b_7= número de último paquete trasmitido

En caso de que el paquete sea un comando el byte #3 es el que corresponde al código del comando y es el último byte trasmitido.

a. Función atención tareas de recepción

Esta tarea consiste en leer un registro recibido y verificado por el nivel 2, determinando en primer lugar si el contenido es un comando o son datos, leyendo el byte #1.

1- en el caso de que el paquete recibido sea un comando, éste será identificado y se procederá a su atención, generando la respuesta en forma inmediata si es posible, o derivándolo al procedimiento al cual está dirigido, y quedando a la espera de contestación del procedimiento invocado.

2- en el caso de que el registro leído contenga un paquete de datos se procederá en primer término a efectuar las verificaciones correspondientes:

1. se determinará si el número del canal al cual está dirigido está efectivamente activado
2. se verificará el estado del canal en las tablas de control de flujo de datos de canal, si está autorizado a recibir
3. se verificará la concordancia del número de paquete trasmitido con el número de paquete que se espera recibir

Si las verificaciones tienen un resultado positivo entonces se procederá a transferir el registro de datos desempaquetado a los registros de trasmisión del canal correspondiente de la interfaz de salida X-25. Esta transferencia se efectivizará mediante una rutina de escritura/lectura y consulta de registros que se comparte con el nivel 3 del protocolo X-25 y que consiste en transferir punteros y longitud de datos.

En caso de que alguna de las verificaciones efectuadas diera resultado negativo se procederá a ejecutar la rutina de error correspondiente.

b. Función atención tareas de trasmisión

1.Luego de cumplir con las tareas de recepción la rutina de atención y control de datos pasa a ejecutar la parte correspondiente a la trasmisión. Con tal fin, en primer término indaga si hay algún comando respuesta de alguno de los procedimientos invocados que debe ser puesto a trasmisión. Si la consulta es positiva se procede utilizando la rutina de lectura y escritura de registros ,que se comparte con el nivel 2 del Protocolo, a transferir el comando a los registros de trasmisión del nivel 2.

2. Si no hay ningún comando respuesta a trasmitir o esta tarea ya ha sido ejecutada, se procede a la atención de la función trasmisión de un paquete de datos. Para ello, de la cola de espera de atención de canales se recoge el que este de

curso y en actividad, observando si tiene algún paquete de datos a trasmitir. Si no tiene ningún paquete se pasa al próximo canal en la cola activa y así sucesivamente hasta que se encuentra uno con necesidades de trasmisión. En este caso se procede en primer término a empaquetar la información agregando el encabezamiento de nivel 3, en el que se consigna: número de canal, número de paquete enviado y número de paquete que se espera recibir. Efectuado el empaquetamiento se procede a su transferencia al nivel 2 para su trasmisión.

c. Función control de flujo de trasmisión y recepción de canales

Esta función se ejecuta con el fin de controlar el flujo de datos a través de cadà uno de los canales. Se hace efectiva llevando una tabla, en la cual se consigna el estado de cada uno de los canales en las funciones de trasmisión y recepción. Durante el proceso de trasmisión/recepción, cada canal puede estar en uno de los cuatro estados que se detallan a continuación.

DC=1 Estado de intercambio de datos completo en ambas direcciones

DC=2 Estado de receptor local no listo. Esto significa que puede trasmitir datos pero no puede recibir, dado que tiene los registros de recepción llenos.

DC=3 Estado de receptor remoto no listo. Esto implica que puede recibir datos pero no trasmitirlos.

DC=4 Estado de pausa completa, no puede recibir ni trasmitir.

Los comandos respuesta que utiliza este procedimiento son:

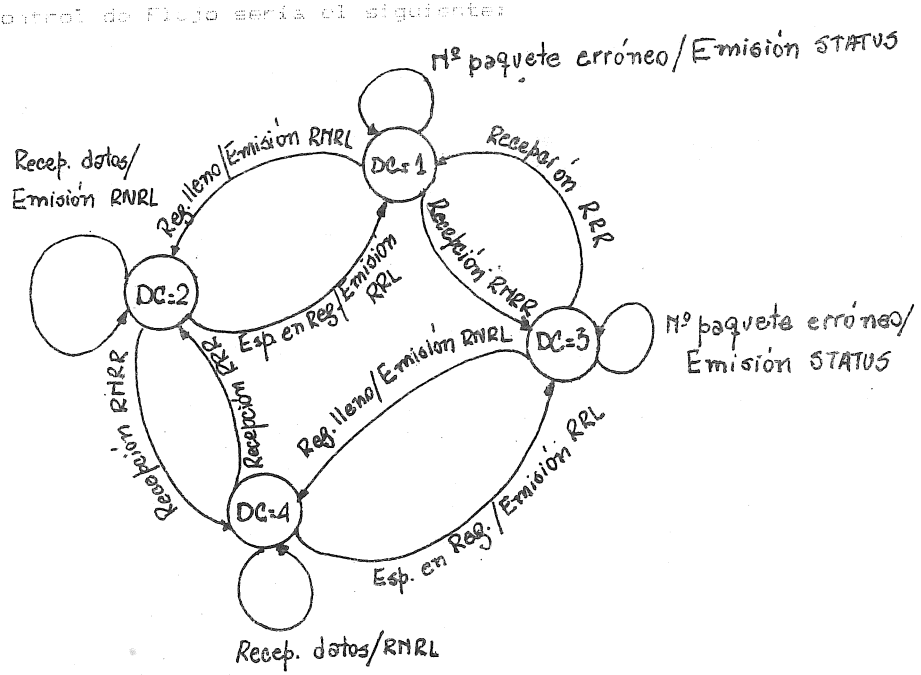
RNRL= receptor local no listo para recibir

RRL= receptor local listo para recibir

RNRR= receptor remoto no listo para recibir

RRR= receptor remoto listo para recibir

El diagrama de estados correspondiente a la [illegible]
Control de flujo sería el siguiente:



V- Conclusión

La presente es una descripción de la filosofía general utilizada para el diseño del Protocolo entre un computador principal y su controlador de comunicaciones.

Se debe tener en cuenta que esta filosofía está ligada y fuertemente influenciada por el hecho de que este Protocolo represente una parte de las comunicaciones del controlador, y que la otra parte de las comunicaciones se hace a través de una salida a la Red Pública ARPAC, cuyo protocolo de comunicación es X-25. Por lo tanto debe existir necesariamente una total correspondencia, para que esto es factible, entre el protocolo controlador de comunicaciones - host y el protocolo de salida controlador de comunicaciones - X.25.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

SERVIÇOS DE TRANSFERÊNCIA DE ARQUIVOS EM REDES LOCAIS DE AUTOMAÇÃO INDUSTRIAL

AUTOR: Tereza Cristina Melo de Brito Carvalho

ENDEREÇO: FDTE/EPUSP
Av. Prof. Luciano Gualberto, Travessa 3, no. 158
05508 - São Paulo - SP - Brasil
Fone: 815.9322 - Ramal 392

RESUMO

A utilização de sistemas computacionais em ambientes de automação industrial vem crescendo, gradativamente, à medida que têm surgido, tanto no mercado nacional como internacional, novos equipamentos desenvolvidos especialmente para este tipo de aplicação. Contudo, para se obter um sistema de automação completo, envolvendo a supervisão e o controle de vários processos de uma unidade fabril, torna-se necessário integrar tais equipamentos. Isto pode ser feito utilizando-se sistemas de arquiteturas distribuídas ou, como são normalmente denominados, Sistemas Digitais de Controle Distribuído (SDCD).

Estes sistemas são constituídos, basicamente, de sistemas computacionais interligados através de sistemas de comunicação, que normalmente são redes locais, desenvolvidas para interconectar equipamentos de tipos específicos.

Neste artigo é apresentado o SDCD empregado na automação de uma unidade fabril para supervisionar e controlar a entrada e saída de matéria-prima, e o processo de fabricação propriamente dito, no caso considerado, de chapas de alumínio.

Este SDCD é constituído de várias redes locais, interligadas entre si. Dentre elas destaca-se a rede local correspondente à sub-rede de gerenciamento, usada para interligar microcomputadores de coordenação e de gerenciamento. Estes microcomputadores trocam informações relativas à situação e à ordem produtiva, através da transferência de arquivos em tal rede local. São apresentadas as características desta rede local e aspectos de especificação dos serviços de transferência de arquivos desenvolvidos para esta rede, considerando as necessidades da aplicação em questão e as tendências de padronização de protocolos de comunicação, usados em ambientes de automação industrial.

1. INTRODUÇÃO

A utilização de sistemas computacionais na automação industrial vem crescendo, gradativamente, à medida que vem surgindo uma grande variedade de equipamentos, no mercado nacional e internacional, destinados a este tipo de aplicação. Todavia, para que seja possível automatizar completamente uma unidade fabril, faz-se necessário integrar tais equipamentos usados na automação industrial.

Com o objetivo de possibilitar tal integração, tem-se empregado sistemas digitais de arquiteturas distribuídas, ou seja, Sistemas Digitais de Controle Distribuído (SDCD), na automação industrial. Estes sistemas são constituídos, basicamente, de equipamentos computacionais interligados através de sistemas de comunicação, que normalmente são redes locais, desenvolvidas com a finalidade de permitir a interconexão de equipamentosde de tipos específicos.

Neste artigo são apresentados os serviços de transferência de arquivos implementados para uma das redes locais, que compõe o Sistema Digital de Controle Distribuído (SDCD) usado na automação de uma fábrica de alumínio. Esta rede, denominada de sub-rede de gerenciamento, é usada para interligar microcomputadores de coordenação e de gerenciamento, que trocam informações, na forma de arquivos, relativas à ordem e à situação produtiva.

Conhecendo-se as características desta rede, são descritos os serviços implementados para a mesma, discutindo-se aspectos relevantes referentes às tendências de padronização dos serviços e protocolos de comunicação desenvolvidos para redes utilizadas em ambientes de automação industrial.

2. ARQUITETURA DO SISTEMA DIGITAL DE CONTROLE DISTRIBUÍDO

Os SDCD's, normalmente empregados na automação industrial, são constituídos dos seguintes níveis hierárquivos

[12] (Figura 2.1):

a) nível de processo - responsável pelo controle do processo, propriamente dito, onde é efetuada a aquisição de dados e a geração de variáveis de controle, com o objetivo de manter as variáveis do processo dentro dos limites aceitáveis em torno do ponto de operação estabelecido;

b) nível de coordenação - responsável pela coordenação e alocação de recursos aos controladores do nível de processo, além de permitir a monitoração de variáveis de controle e o estabelecimento de pontos de operação dos processos controlados;

c) nível de gerenciamento - responsável pela determinação da estratégia da ordem de produção da unidade fabril correspondente.

No caso do SDCD usado na automação da fábrica de alumínio, os microcomputadores de coordenação (8 e 16 bits) e de gerenciamento (16 bits) são interligados através de uma rede local, correspondente à sub-rede de gerenciamento (Figura 2.2), e trocam informações entre si através da transferência de arquivos, que contêm dados relativos aos planos mensais de produção, carga diária das máquinas, posição do estoque de matéria-prima, situação das máquinas, entre outros.

As características físicas e os protocolos de comunicação desta sub-rede de gerenciamento são apresentadas no próximo item. Os serviços prestados por tais protocolos de comunicação foram utilizados na implementação dos serviços de transferência de arquivos desenvolvidos para este SDCD.

3. DESCRIÇÃO DA SUB-REDE DE GERENCIAMENTO

A rede local, correspondente à sub-rede de gerenciamento, foi adquirida no mercado nacional e possibilita a interconexão de microcomputadores de portes diferentes (8 e 16 bits) [15]. A sua topologia é do tipo via comum, implementada fisicamente através de um cabo coaxial, ao qual

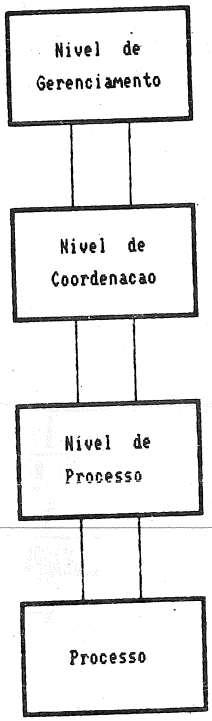


Figura 2.1 - Estrutura de um Sistema Digital de Controle Distribuído empregado na Automação Industrial.

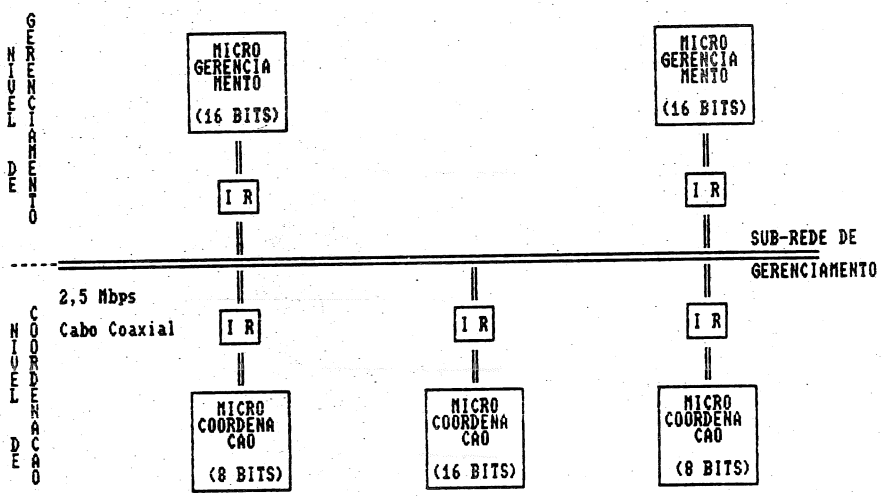


Figura 2.2 - Sub-Rede de Gerenciamento do Sistema Digital de Controle Distribuído desenvolvido para uma Planta Fabril

são ligadas as estações de trabalho através de Interfaces de Rede (IR). A taxa de transmisão desta rede é de 2,5 Mbps, e a distância máxima permitida entre duas estações é igual a 1,5 Km.

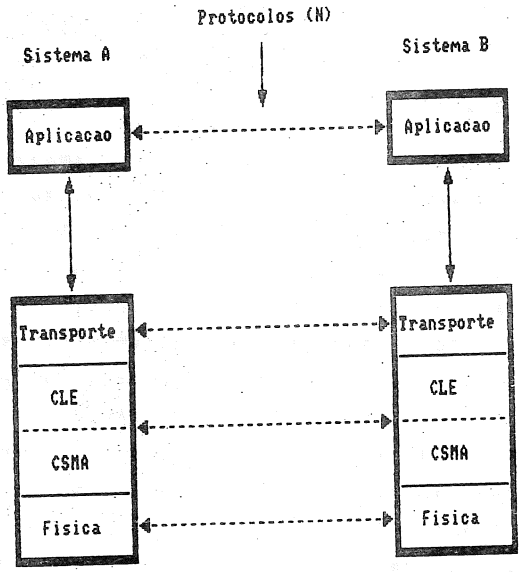
A comunicação entre tais estações de trabalho é realizada através de protocolos de comunicação, desenvolvidos de acordo com a estrutura do modelo básico de referência proposto pela ISO (International Organization for Standardization) (Figura 3.1).

Nesta estrutura, a camada de enlace é responsável pela monitoração, programação e ativação das Interfaces de Rede, quando da transmissão e/ou recepção de mensagens da linha. As camadas física e de enlace foram implementadas a partir das recomendações IEEE 802.3 e IEEE 802.2 [5], [10], embora não tenham sido desenvolvidos os mecanismos de detecção de colisão previstos para o protocolo de acesso ao meio de transmissão CSMA/CD. A camada de rede inexiste, pois não foi prevista a possibilidade de haver troca de mensagens entre redes distintas. Por último, tem-se a camada de transporte, que oferece ao usuário os serviços de transmissão de dados fim-a-fim entre duas estações da rede. Esta camada foi implementada segundo especificações particulares do próprio fabricante da rede local considerada. As camadas de sessão e apresentação inexistem.

Portanto, ao usuário desta rede são oferecidos os serviços da camada de transporte, que são utilizados pela camada de aplicação para prestar os serviços de transferência de arquivos a ela associados. No próximo item, são descritos a estrutura da camada de aplicação e os serviços por ela prestados.

4. SERVIÇOS DE TRANSFERÊNCIA DE ARQUIVOS NA SUB-REDE DE GERENCIAMENTO

Os serviços de transferência de arquivos, dentro do modelo básico de referência da ISO, são prestados pela



CLE - Controle Lógico de Enlace
CSMA - Carrier Sense Multiple Access

Figura 3.1 - Estrutura de Protocolos dos Sistemas Interligados através da Sub-rede de Gerenciamento

camada de aplicação, cuja estrutura é apresentada a seguir.

4.1. Estrutura da Camada de Aplicação

A camada de aplicação tem por objetivo oferecer meios aos processos de aplicação de acessar os recursos do ambiente de interconexão de sistemas abertos. Num sistema aberto podem residir vários processos de aplicação, que são elementos lógicos responsáveis pelo processamento de informações relativas a uma determinada aplicação. Cada processo de aplicação é representado num sistema aberto através de uma entidade de aplicação.

Cada entidade de aplicação, por sua vez, é constituída por elementos funcionalmente distintos (Figura 4.1), podendo-se identificar dois tipos de elementos: o elemento de usuário e o elemento de serviço de aplicação [1]. O elemento de usuário (User Element (UE)) corresponde à parte do processo de aplicação que usa os serviços prestados pelos elementos de serviço de aplicação com a finalidade de realizar funções que envolvam a comunicação com outros sistemas. Os elementos de serviço de aplicação, por sua vez, são classificados da seguinte maneira:

. Elementos de Serviço de Aplicação Comuns - (Common Application Service Elements (CASE's)) - oferecem serviços básicos que podem ser utilizados por diversas aplicações. Como exemplo, podem-se citar o estabelecimento e a liberação de uma conexão entre entidades de aplicação;

. Elementos de Serviço de Aplicação Específicos - (Specific Application Service Elements (SASE's)) - oferecem serviços específicos relacionados a uma dada aplicação. Como exemplo, podem-se citar: Transferência, Acesso e Gerenciamento de Arquivos (File Transfer Access and Management (FTAM)), Submissão Remota de Tarefas (Job Transfer and Manipulation (JTM)), Terminais Virtuais (Virtual Terminal (VT)), entre outros.

As normas de padronização da ISO do elemento de

serviço de aplicação SASE_FTAM [6], [7], [8], [9], responsável pela prestação dos serviços de transferência de arquivos, definem, além dos serviços e do protocolo de aplicação, um sistema de arquivos virtual, descritos no próximo item.

4.2. Elemento de Serviço de Aplicação SASE_FTAM Especificado pela ISO

O primeiro problema encontrado, quando se deseja implementar os serviços de transferência de arquivos numa rede, está relacionado com as diferenças existentes entre os diversos sistemas de arquivos associados a diferentes estações de uma rede. Estas diferenças dizem respeito ao modo de armazenamento dos dados num arquivo e ao modo de acessá-los. Com objetivo de resolver problemas decorrentes da heterogeneidade entre sistemas de arquivos associados a diferentes estações da rede, a ISO decidiu especificar um modelo de Sistema de Arquivos Virtual (SAV), descrito no próximo item.

4.2.1. Sistema de Arquivos Virtual

O modelo de Sistema de Arquivos Virtual, proposto pela ISO, define a estrutura e os atributos associados aos arquivos, e os tipos de operações que podem ser efetuadas sobre os mesmos.

A estrutura de arquivos, proposta por este modelo, é a hierárquica, que tem como vantagens a simplicidade e a possibilidade de descrever vários tipos de arquivos a partir desta estrutura, tais como: os arquivos sequenciais e indexados.

Em relação aos atributos, são definidos três subconjuntos:

a) subconjunto de atributos básicos: os atributos deste subconjunto devem ser sempre oferecidos pelos sistemas de arquivos associados às estações da rede. Como exemplos de atributos deste subconjunto, podem-se citar: o nome, o tipo

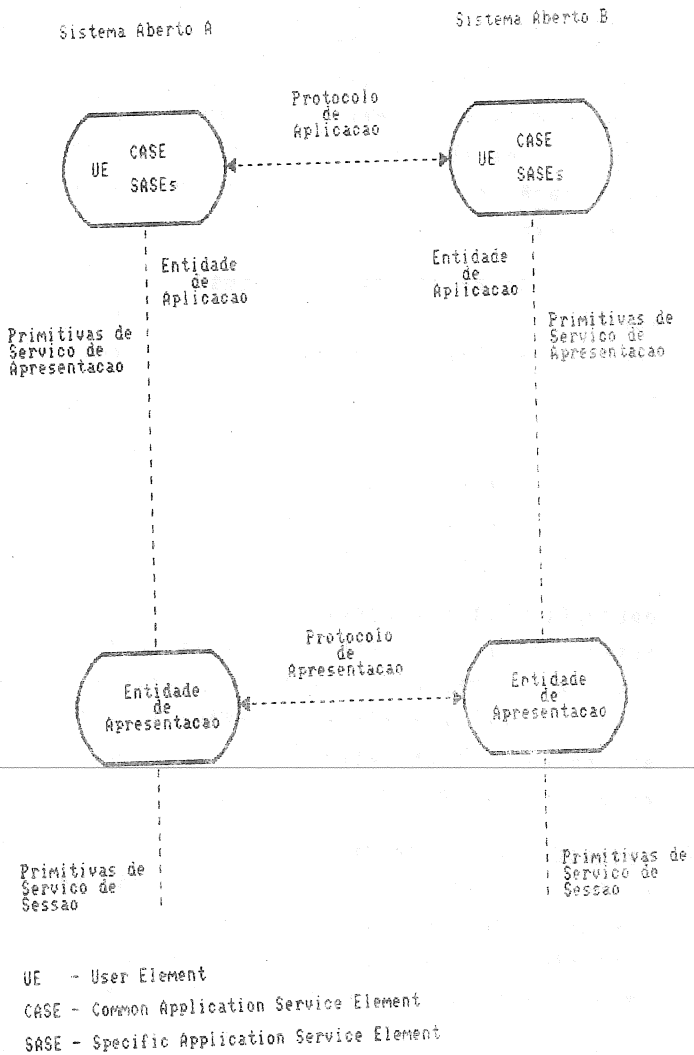


Figura 4.1 - Estrutura da Camada de Aplicação

de dados e o controle de acesso corrente a um arquivo.

b) subconjunto de atributos de armazenamento: os atributos deste subconjunto estão relacionados com as características do meio de armazenamento. Pertencem a este subconjunto, por exemplo, os seguintes atributos: tamanho, data da última atualização e controle de concorrência corrente associado a um arquivo.

c) subconjunto de atributos de segurança: os atributos deste subconjunto estão relacionados com aspectos de segurança e controle de acesso a arquivos do sistema. Como exemplos de atributos deste subconjunto, podem-se citar: senhas de acesso e identificação de algoritmos de criptografia.

Além disso, o modelo SAV da ISO define, também, os tipos de operações que podem ser efetuadas sobre arquivos. Tais operações são classificadas da seguinte maneira:

a) operações sobre o arquivo todo: nesta classe de operações estão incluídas: SELECT (verificação da existência de arquivo), CREATE (criação de arquivo), READ ATTRIBUTE (leitura de atributos), CHANGE ATTRIBUTE (alteração de atributos), OPEN (abertura de arquivo), CLOSE (fechamento de arquivo) e DELETE (remoção de arquivo);

b) operações sobre dados do arquivo: nesta classe de operações estão incluídas: LOCATE (localização de registro), READ (leitura de registro), INSERT (inserção de registro), Extensão (inclusão de registro no fim do arquivo), REPLACE (modificação de registro) e ERASE (remoção de registro).

A partir deste Sistema de Arquivos Virtual e do Sistema de Arquivos Real são definidas as funções de mapeamento, que absorvem as diferenças de representação e especificação de arquivos existentes entre os dois sistemas (Figura 4.2).

Além de definir um modelo de Sistema de Arquivos Virtual, as normas de padronização da ISO definem os serviços, que devem ser prestados pelo elemento de serviço de aplicação

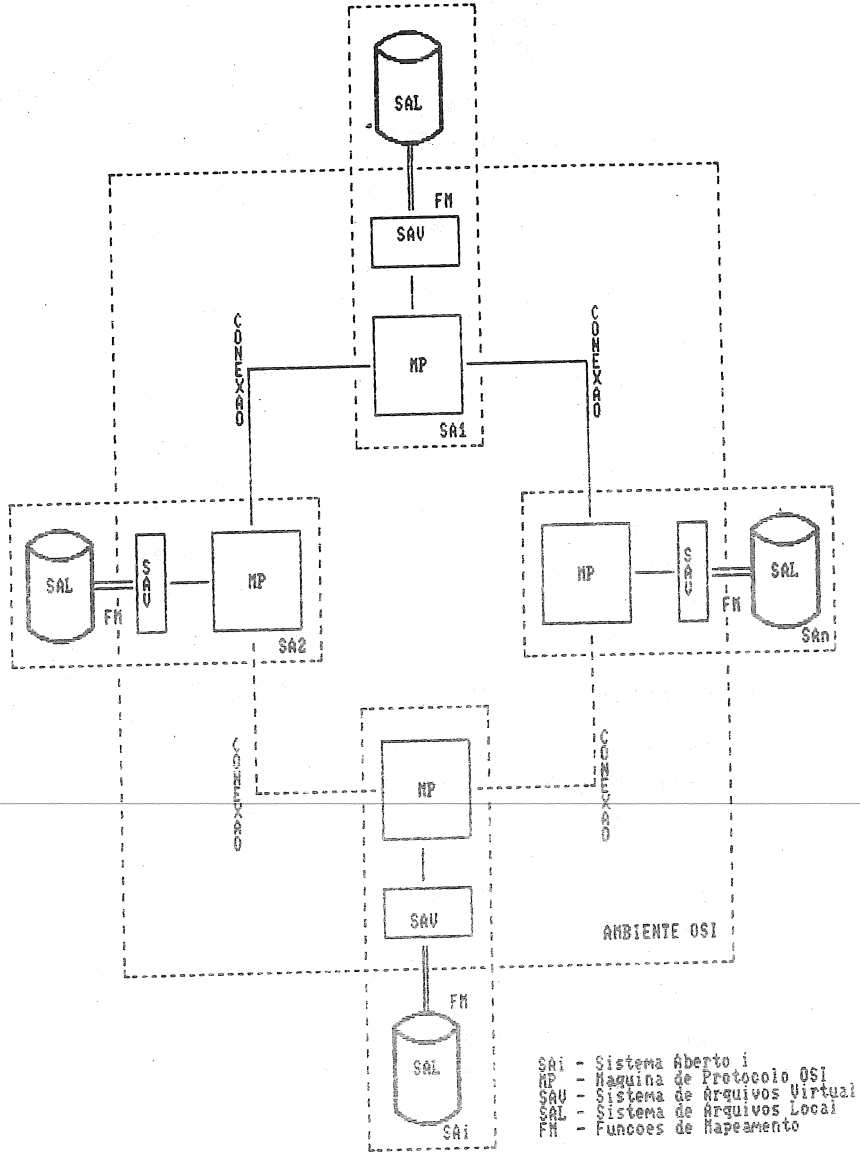


Figura 4.2 - Arquitetura de Sistemas Abertos com Serviços de Transferência, Acesso e Gerenciamento de Arquivos

SASE_FTAM, que são descritos a seguir.

4.2.2. Serviços Oferecidos pelo Elemento de Serviço de Aplicação SASE_FTAM

Os serviços, especificados pelas normas de padronização da ISO, são agrupados nos seguintes subconjuntos:

a) Subconjunto de Serviços Básicos

Neste subconjunto são especificados os serviços de estabelecimento e liberação de associação (conexão) de aplicação, além dos serviços de seleção e "desseleção" de arquivo. O serviço de seleção é usado para verificar se o arquivo especificado existe ou não no sistema de arquivos de uma estação remota. Em caso positivo, este arquivo passa a ser associado a conexão de aplicação anteriormente estabelecida. Esta associação é desfeita utilizando-se o serviço de "desseleção".

b) Subconjunto de Serviços de Transferência de Dados

Os serviços deste subconjunto têm por função prover meios para que seja possível transferir dados de um arquivo, entre dois sistemas abertos. Assim sendo, este subconjunto inclui os serviços de abertura, fechamento, leitura e gravação de dados de um arquivo residente numa estação remota.

c) Subconjunto de Serviços de Gerenciamento Limitado

Este subconjunto inclui os serviços de criação, remoção e leitura dos atributos de um arquivo residente numa estação remota.

d) Subconjunto de Serviços de Gerenciamento Extendido

Este subconjunto inclui o serviço de alteração dos atributos de um arquivo residente numa estação remota, e pressupõe a existência obrigatória do Subconjunto de Serviços de Gerenciamento Limitado.

e) Subconjunto de Serviços de Controle de Erros

Os serviços deste subconjunto têm por função prover mecanismos de recuperação de erros e de retransmissão de dados de um arquivo, que estava sendo transferido.

f) Subconjunto de Serviços de Acesso a Arquivo

Neste subconjunto são especificados os serviços de localização, remoção e movimentação de registros de um arquivo residente numa estação remota.

Naturalmente, nem todos subconjuntos destes serviços precisam ser implementados, exceto o subconjunto de serviços básicos. Isto dá, ao projetista, a flexibilidade de implementar os subconjuntos de serviços que atendam às necessidades da aplicação envolvida.

As normas de padronização definem classes e níveis de serviços (Tabelas 4.1 e 4.2), para os quais existem subconjuntos de serviços obrigatórios e outros que são opcionais.

As entidades de aplicação, que prestam tais serviços de transferência de arquivos, comunicam-se entre si através do protocolo de aplicação, também especificado pelas normas da ISO.

4.2.3. Protocolo de Transferência de Arquivos

A ISO adotou, como arquitetura do protocolo de transferência de arquivos, um modelo simplificado, constituído apenas de duas entidades de aplicação envolvidas na transferência de arquivos, que são as seguintes:

a) entidade iniciadora, que acumula as funções de controladora, produtora ou consumidora de dados;

b) entidade respondedora, que exerce as funções de consumidora ou produtora de dados.

O fluxo de informações trocadas ente estas duas

Subconjunto de Serviços / Classes de Serviço	Básicos	Transferência de Dados	Gerenciamento Limitado	Gerenciamento Extendido	Controle de Erro	Acesso a Arquivo
Classe de Transferência de Arquivo	X	X	OP	OP (*)	OP	-
Classe de Acesso a Arquivo	X	X	OP	OP (*)	OP	X
Classe de Gerenciamento de Arquivo	X	-	X	X	-	-
Classe de Transferência e Gerenciamento de Arquivo	X	X	X	X	OP	-
Classe Definida Pelo Usuário	X	OP	OP	OP (*)	OP	OP

(*) O Subconjunto de Serviços de Gerenciamento Extendido pressupõe sempre a existência do Subconjunto de Serviços de Gerenciamento Limitado.

Tabela 4.1 - Relação entre as Classes de Serviços e os Subconjuntos de Serviços Oferecidos ao Elemento de Usuário

Subconjunto de Serviços / Nível de Serviço	Básicos	Transferência de Dados	Gerenciamento Limitado	Gerenciamento Extendido	Controle de Erro	Acesso a Arquivo
Nível de Serviço Corrigível pelo usuário	X	X(*)	X(*)	OP	OP	OP
Nível de Serviço Confiável	X	X(*)	X(*)	OP	-	OP

(*) No caso deste nível de serviço, ao menos um destes dois subconjuntos de serviços precisa ser oferecido.

Tabela 4.2 - Relação entre os Níveis de Serviços e os Subconjuntos de Serviços Oferecidos ao Elemento de Usuário

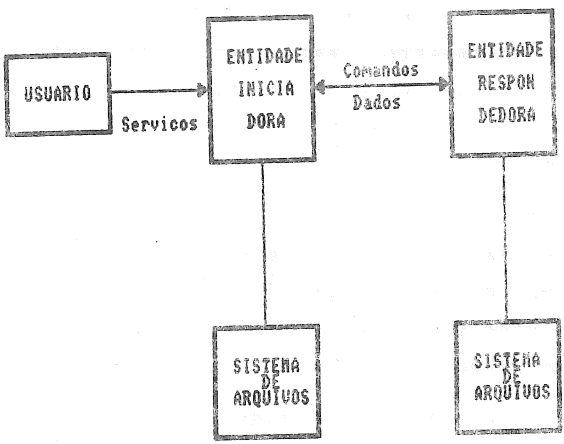


Figura 4.3 - Modelo da Arquitetura de Protocolo de Transferência de Arquivos proposta pela ISO

entidades contém, tanto comandos como dados (Figura 4.3), que constituem as unidades de dados do protocolo de transferência de arquivos.

Este protocolo de transferência de arquivos ou, mais especificamente, a máquina de protocolo que implementa o elemento de serviço de aplicação SASE_FTAM, deve garantir a execução da sequência dos serviços descritos anteriormente, verificando a consistência e tratando as primitivas de serviços emitidas pelo elemento de usuário e as unidades de dados de protocolo recebidas da entidade parceira. Como resultado deste tratamento, esta máquina de protocolo pode, quando necessário, invocar os serviços da camada de apresentação, enviar primitivas de serviço ao elemento de usuário e efetuar processamento local de dados.

Além disso, dependendo do nível de serviço oferecido (nível de serviço confiável), este protocolo deve possuir mecanismos de recuperação de erros, no caso de perda de dados do arquivo transferido e de interrupção da transferência em andamento.

Em relação às unidade de dados de protocolo, pode-se dizer que os seus tipos coincidem com os tipos de serviços anteriormente citados.

No próximo item são apresentados aspectos de especificação do elemento de serviço de aplicação SASE_FTAM implementado, que tomou por base as normas de padronização citadas [6], [7], [8] e [9].

4.3. Elemento de Serviço de Aplicação SASE_FTAM Implementado

As normas de padronização da ISO definem, em primeiro lugar, um modelo de Sistema de Arquivos Virtual, especificando a estrutura do arquivo, os atributos associados ao arquivo e os tipos de operações efetuadas sobre o mesmo. Em relação à estrutura de arquivos, nos sistemas utilizados são, inicialmente, manipulados apenas arquivos sequenciais, que são casos particulares da estrutura hierárquica especificada para

este modelo.

Dos subconjuntos de atributos definidos para este modelo, são suportados, pelo Sistema de Arquivos Real, somente os subconjuntos básico e de armazenamento, que incluem os atributos: nome do arquivo, tipo de dados, tamanho, data da última alteração e controle de acesso exclusivo (não é permitido acesso compartilhado aos dados de um arquivo).

Por último, são definidos o conjunto de operações que podem ser efetuadas sobre os arquivos. Destas operações, convém observar que as funções associadas às operações SELECT, OPEN e CREATE especificadas para o modelo SAV são realizadas por uma única operação existente no Sistema de Arquivos Real, tendo sido, portanto, necessário implementar funções de mapeamento que absorvam tais diferenças. Em relação às operações sobre dados de arquivo, as normas definem, para os arquivos sequenciais, somente as operações de busca (localização do primeiro e último registro do arquivo, e do registro seguinte e anterior ao registro seguinte), inserção e leitura. No caso do Sistema de Arquivos Real não é possível efetuar a operação de busca do registro anterior, operação que não se mostrou necessária para a classe de serviços oferecida pelo elemento de serviço de aplicação SASE_FTAM implementado.

Quanto aos subconjuntos dos serviços oferecidos pelo elemento de serviço de aplicação SASE_FTAM implementado, estes foram especificados considerando as necessidades da aplicação em questão e as características da rede local utilizada. Do ponto de vista da aplicação envolvida, o mais adequado seria a implementar a classe de transferência de arquivos com mecanismos de recuperação de erros. Isto implicaria na implementação dos subconjuntos de serviços básico e de transferência de dados, utilizando os serviços de sincronização da conexão da camada de sessão. Todavia, como a camada de sessão inexiste, no caso da rede local utilizada, foi implementada a classe de transferência de dados sem mecanismos de recuperação de erros, ou seja, é oferecido, ao elemento de usuário, o nível de serviço "corrigível pelo

usuário", conforme a definição da ISO.

Os serviços de transferência de arquivos, prestados pelo elemento de serviço de aplicação SASE_FTAM, são utilizados pelo elemento de usuário para realizar as funções a ele atribuídas, em decorrência das necessidades dos processos aplicativos definidos para o sistema implementado. No próximo item são apresentados alguns aspectos de especificação do elemento de usuário implementado para a rede local do SDCD.

4.4. Elemento de Usuário Implementado

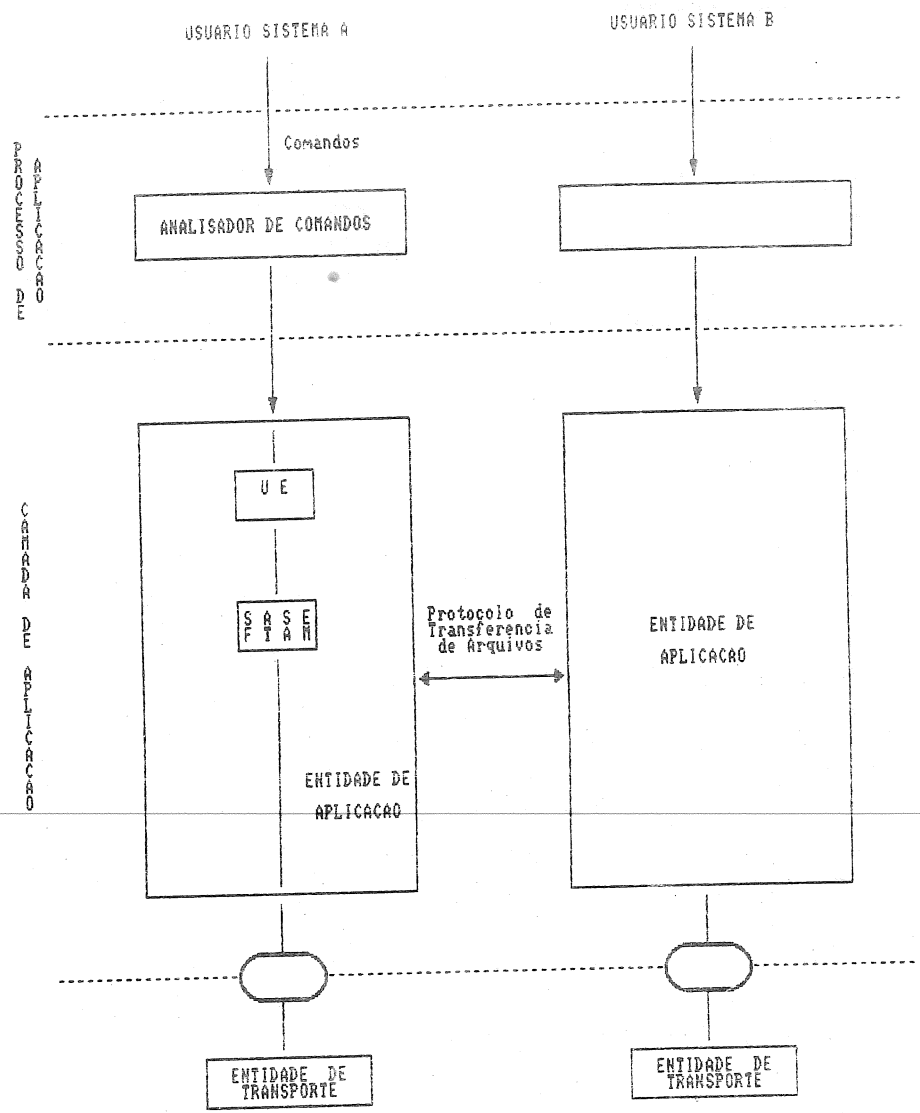
Considerando as necessidades da aplicação envolvida foram definidos, para esta rede local do SDCD, os comandos a serem implementados utilizando, para isso, os serviços de transferência de arquivos oferecidos pela mesma. Estes comandos são os seguintes:

a) UPLOAD - transferência de dados de um arquivo de uma estação remota para a estação local;

b) DOWNLOAD - transferência de dados de um arquivo da estação local para uma estação remota.

Estes comandos, antes de serem encaminhados ao elemento de usuário, são tratados através de um analisador léxico e sintático (Figura 4.4). Se for verificada a validade do comando especificado, este é encaminhado ao elemento do usuário, que se encarrega de solicitar os serviços do elemento de serviço de aplicação SASE_FTAM, de modo que tal comando seja executado.

Os comandos tratados pelo elemento de usuário são bem simples, devido as próprias necessidades da aplicação em questão. Contudo, como os serviços dos subconjuntos de gerenciamento de arquivos também estão sendo implementados, à medida que se julgar necessário, é possível implementar, para esta rede, outros comandos que utilizem tais serviços.



UE - User Element
SASE - Specific Application Service Element
FTAM - File Transfer Access and Management

Figura 4.4 - Estrutura da Camada de Aplicação Implementada

5. CONSIDERAÇÕES FINAIS

Antes de definir o conjunto de serviços a ser implementado para a rede local do SDCD considerado, foram estudadas várias especificações de serviços de transferência de arquivos. A diferença básica existente entre estas especificações e a da própria ISO estava nos tipos de serviços propostos. As normas de padronização da ISO especificam serviços bem elementares (estabelecimento de conexão, seleção, abertura e transferência de dados propriamente dita), que devem ser invocados para que seja possível realizar, por exemplo, uma operação de cópia de arquivos entre estações da rede. As outras especificações definem serviços mais poderosos, em que a cópia de arquivos pode ser invocada diretamente.

A vantagem dos tipos de serviços definidos pelas normas da ISO advém do fato de proporcionar ao projetista maior flexibilidade na especificação dos processos aplicativos, que utilizam os serviços de transferência de arquivos, para realizar as funções a eles atribuídas. Outro aspecto a ser considerado diz respeito ao Sistema de Arquivos Virtual, que resolve os problemas de heterogeneidade existentes entre sistemas de arquivos diferentes, permitindo a interconexão de estações da rede as quais estejam associados sistemas de arquivos de tipos de complexidade variável. Por último, deve-se considerar, também, o fato destas normas da ISO estarem de acordo com as tendências de padronização de protocolos de comunicação para redes locais aplicadas em ambientes de automação industrial.

A arquitetura FULL_MAP (Manufacturing Automation Protocol) [4], [11], definida para redes locais usadas em automação industrial, especifica como um dos protocolos da camada de aplicação, o protocolo de transferência da ISO ou, como é mais conhecido, o FTAM (File Transfer Access and Management).

Contudo, o FTAM devido a grande flexibilidade

oferecida e ao fato de procurar atender um grande número de aplicações, é bastante complexo. No caso do sistema implementado, o elemento de serviço de aplicação SASE_FTAM foi desenvolvido em PLM86, e ocupa uma área total de código mais dados igual a 75 Kbytes. Para implantar este programa nos microcomputadores de 8 bits, deveriam se utilizar mecanismos de "overlay", uma vez que outros processos aplicativos também deveriam ser executados nestes mesmos micros.

Esta solução mostrou-se não ser satisfatória, tendo-se optado, então, por implementar um conjunto de serviços mais simplificado, especificado pelas normas de padronização MMS (Manufacturing Message System) [4] adotadas pela arquitetura MINI_MAP. Tais serviços são na realidade um subconjunto daqueles especificados para o FTAM da ISO e foram definidos considerando aspectos reais de implementação.

Estes serviços estão sendo implementados no momento, e devem ser implantados nos microcomputadores de 8 bits, uma vez que se constatou uma redução significativa da área de código mais dados ocupada pelo programa correspondente (6,5 Kbytes). Os microcomputadores de 16 bits, por sua vez, usarão os dois protocolos, SASE_FTAM e SASE_MMS, para se comunicarem respectivamente com outros micros de 16 bits e 8 bits.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] CARVALHO, Tereza C.M.B. Um Protocolo de Transferência de Arquivos Aplicado em Redes de Automação Industrial. In: XX CONGRESSO NACIONAL DE INFORMÁTICA. Anais... São Paulo, 1987. v.2, p. 624 - 632.

[2] CARVALHO, Tereza C.M.B.; CARVALHO, Antonio J.C. Aplicação de Redes de Computadores em Automação Industrial. In: 5o. SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES. Anais... São Paulo, 1987. p.21-54.

[3] CROWDER, Robert. The MAP Specification. Control Engineering (10): 22 - 25, out. 1985.

[4] GENERAL MOTORS. MAP Specification - Version 2.1. EUA, General Motors, 1985. n.p.

[5] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. IEEE Standards for Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) - Access Methods and Physical Layer Specifications. EUA, Institute of Electrical and Electronics Engineers, Inc, 1985. 140p.

[6] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. DIS 8571/1 - Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 1: General Introduction. International Organization for Standardization, 1986. 40p.

[7] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. DIS 8571/2 - Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 2: The Virtual Filestore Definition. International Organization for Standardization, 1986. 52p.

[8] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. DIS 8571/3 - Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 3: The File Service Definition. International Organization for Standardization, 1986. 109p.

[9] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. DIS 8571/4 - Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 4: The File Protocol Specification. International Organization for Standardization, 1986. 131p.

[10] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. Information Processing Systems - Local Area Networks - Part 2: Logical Link Control. International Organization for Standardization, 1987. 82p.

[11] KAMINSKI, Michel A. Jr. Protocols for Communicating in the Factory. IEEE Spectrum (4): 56 - 62, abr. 1986.

[12] LEITE, José R. E.; MAGALHÃES, Maurício F. Protocolos de Aplicação em Redes Locais de Computadores na Automação Industrial (Protocolos MAP/TOP). In: 5o. SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES. Anais... São Paulo, 1987. p.1 - 20.

[13] MARTUCCI, Moacyr Jr.; ZERBINI, Ricardo C. Controle de Processos. In: 1o. CONGRESSO NACIONAL DE AUTOMAção INDUSTRIAL. Anais... São Paulo, 1983. p.590 - 597.

[14] MOON, Dan. Developing Standrads Smooth the Integration of Programmable Factory Floor Devices. Control Engineering (10): 49 - 52, out. 1985.

[15] RIBEIRO Filho, Henrique; NAVARRO, Luiz C. Protocolo de Comunicação da Rede Scopus. Boletim Scopus 6 (69): 9 - 20, mar. 1984.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa.
Buenos Aires, Setiembre 1988.

Diseño de una Red de Transmisión de Mensajes

Jorge A. Aguirre Norberto Ambrogetti
Marcos A. Ramos Martinez Paulina Frenkel

Grupo : E.S.L.A.I / Servicio de Procesamiento
de Datos SECYT - CONICET

1 - Introducción

Se presenta un desarrollo para la implementación de una red heterogenea de transmisión de mensajes que podrá ser utilizado en un futuro en la implementación de la Red Cientifica y Técnica Nacional.

El sistema fue diseñado para funcionar sobre una subred de baja velocidad de transferencia(ARPAC), con el objetivo de ser especialmente eficiente para la transmisión de mensajes con listas de distribución extensas.
Está organizado en dos grandes unidades funcionales diferenciadas: los nodos servidores y los nodos usuarios.

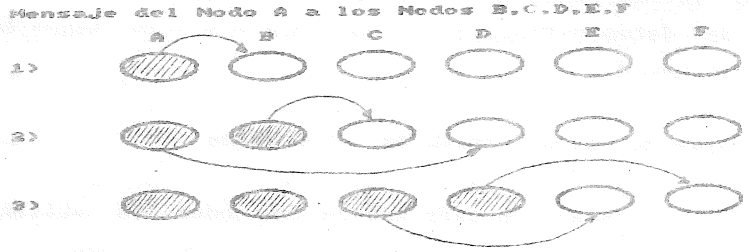
Los nodos servidores poseen la información referente a la configuración y estado de la red y a la ubicación y atributos de los usuarios. Tienen la función de autorizar, iniciar y administrar las transferencias de información, sin intervenir directamente en ellas. Este desarrollo incluye un único servidor.

Los nodos usuarios solicitan envios de mensajes y actúan bajo indicación de los servidores como emisores o receptores de información.

Esto supone, respecto de una organización donde los servidores contienen los "in_boxs" en los que se deposita y de donde se extraen los mensajes [1] la desventaja que, para que se efectivice una transferencia, sea necesario que emisor y destino

estén disponibles simultáneamente para establecer la comunicación La ventaja reside en no requerir que los mensajes deban ser transmitidos por la subred dos veces, evitando de esta forma la saturación de la vinculación servidor-subred.

Así mismo, para disminuir el tiempo de transmisión de un mensaje con una larga lista de distribución se establece una política de cooperación entre nodos usuarios. Cada nodo que haya recibido un mensaje podrá convertirse en emisor de una transmisión hacia los nodos de la lista que aún no lo hayan recibido. De esta manera un mismo mensaje puede generar varios procesos paralelos de transmisión, como lo muestra la figura siguiente:



La selección de nodos para la cooperación se realiza de acuerdo a un esquema jerárquico fijado por el administrador de la red y por la determinación de cada nodo sobre su disponibilidad.
El software desarrollado incluye las funciones de contabilización de uso de recursos que permitirán la ulterior implementación de una política de cobro de servicios.

A los efectos de la presente descripción haremos referencia al Nodo Servidor como Server y a los Nodos Usuarios como Nodos. Llamaremos Usuario a cada individuo o proceso autorizado para acceder al sistema.

2 - Descripción del server

La información referente a la configuración de la red es mantenida por el server en dos bases de datos: una de usuarios y otra de nodos.

Base - de datos de usuarios. Contiene la información primaria: nombre, nodo y privilegios de uso, e información complementaria como proyectos a los que esta vinculado, estadísticas sobre el uso de la red, etc.

Base de datos de los nodos. Contiene nombre externo del nodo, identificador interno, dirección en la subred y nivel de retransmisión.

Los procesos involucrados en el server interactúan además con estructuras de datos referentes a los nodos y a los mensajes, generadas dinámicamente.

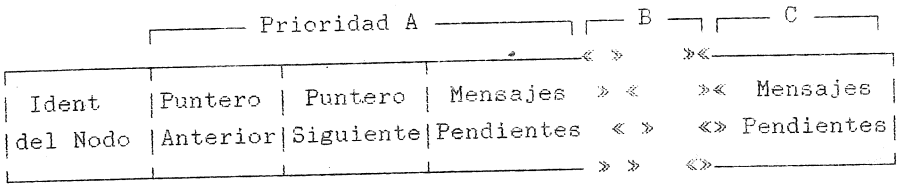
2.1 - Estructuras de datos Referentes a los nodos

Para describir el estado de cada uno de los nodos se utilizan arreglos a los cuales se accede por el identificador interno del nodo. En ellos se mantiene la información primaria extraída de la base de datos, la cantidad de canales disponibles, la cantidad de mensajes pendientes, y los campos necesarios para enlazar listas doblemente encadenadas que vinculan los nodos disponibles que son receptores de mensajes pendientes. Para cada una de las tres prioridades de transmisión que maneja el sistema existe una de estas listas.

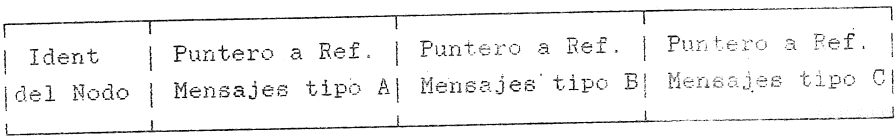
- Descripción de los Nodos

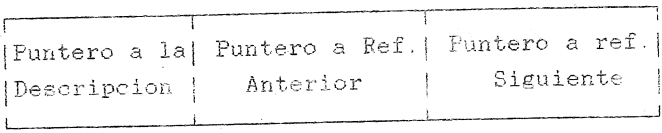
Ident del Nodo	Direccion en la subred	Estado	Cant Canales	Nivel de Retransmision	Mensajes Pendientes

- Receptores Libres



- Receptores Mensajes/Prioridades

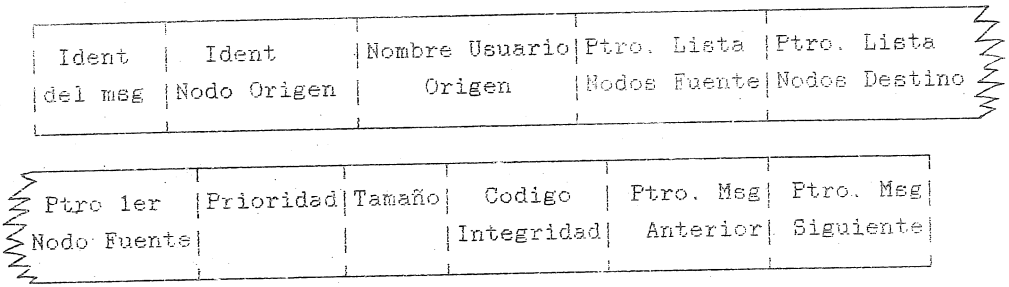




Los punteros a Referencia mensajes tipo A, B y C apuntan a las listas de referencia de mensajes correspondientes a cada prioridad.

2.2 - Estructuras de datos referentes a los mensajes .

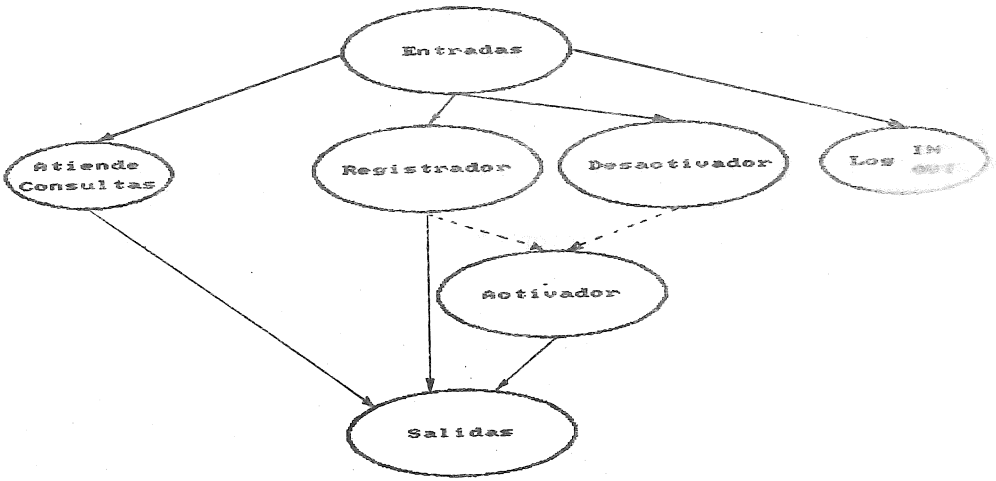
Se mantiene una estructura Descripción del mensaje que contiene información estática: origen (nodo,usuario), tamaño, código de verificación; e información dinámica que describe el estado del proceso de transmisión del mensaje:lista de nodos poseedores del mensaje -nodos fuente- ; lista de nodos destinatarios que aún no lo recibieron.



Cooperando con esta estructura se mantiene la de **Mensajes Activos** que provee un mecanismo de identificación de las transacciones asociadas a los mensajes en curso. Todos los reportes de los nodos involucrados en una de estas transacciones contienen esa identificación.
Los identificadores son generados por un algoritmo que asegura la unicidad de los mismos, prove acceso directo a la información asociada, reincorpora los identificadores en desuso para una nueva utilización y libera el espacio de la información asociada que ya no tiene utilidad.

2.3 - Interacción entre los procesos

El server esta compuesto por una serie de módulos que atienden las distintas funciones referidas al envio de mensajes.
En el siguiente diagrama simplificado se intenta mostrar la interacción de esos procesos.



- Procesos Principales

- Registrador

Verifica la validez de los parámetros de un pedido de envío, produce los eventuales rechazos y de tratarse de un requerimiento válido, lo incorpora a las estructuras anteriores y si es necesario despierta al proceso de activación.

- Activador

Este es el proceso que recorre las estructuras en busca de la información para construir una terna mensaje-fuente-destino. Esta búsqueda se inicia, alternativamente, en una de las tres listas de Receptores Libres. La cantidad de visitas consecutivas a cada una de ellas esta determinada por las prioridades que se les ha asignado.
Si se encuentra un receptor libre se pasa a ejecutar el proceso Completar Terna.

El proceso se duerme cuando ha fracasado en una recorrida completa sin que durante la misma se hayan producido modificaciones en los estados involucrados y se despierta cuando cualquier evento produce una modificación a dichos estados.
El proceso Completar Terna busca un mensaje destinado al receptor corriente e intenta asociarle un transmisor libre. Para ello se vale de las estructuras Receptores Mensajes/Prioridades y Descripción Mensajes

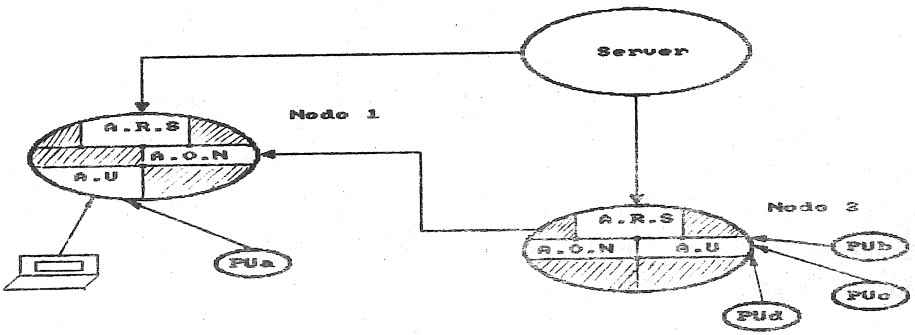
- Desactivador

Este proceso es llamado ante la recepción de un mensaje de fin de transmisión. Actualiza la cantidad de canales de los nodos implicados, si corresponde los incluye en la estructura de Receptores Libres y actualiza la estructura Descripción Mensajes en lo referente a las listas de fuentes y destinos.
Si se ha detectado la caida de uno de los nodos se lo registra como fuera de servicio.

3 - Descripción del Nodo

Sus interfaces con los usuarios, el servidor y otros nodos

El nodo recibe pedidos de atención y de acuerdo a quien los genere (un usuario, otro nodo, o el server) el sistema activa el módulo adecuado para satisfacer dicho pedido, el que sera Local si lo genera un usuario y Remoto si es generado por otro nodo o el server. Planteado esto presentamos al nodo como un conjunto de módulos que interactúan entre si.



3.1 - Función de Atención a los Usuarios (AU)

La interfase del sistema con la que los usuarios se comunican para solicitar un servicio la denominamos Frontera. El acceso a esta frontera se realiza en forma interactiva (a través de un sistema de menúes) o en forma diferida via una aplicación programada por el usuario (la que solicita a la frontera ciertos servicios). En ambas modalidades esta frontera brinda al usuario las siguientes facilidades:

a) Cambiar su Palabrar Clave de Acceso.

b) Solicitar el envío de un mensaje.

c) Consultar al Server.

a) Cambiar su Palabra Clave de Acceso

Esta facilidad permite a un usuario acceder a la base de dàtos de usuarios autorizados, a los efectos de modificar su palabra clave.

b) Solicitar el envío de un mensaje

Para brindar esta facilidad existe un módulo que recibe el Nombre del Archivo donde reside el texto del mensaje, la Lista de Usuarios destinatarios y el Nombre y Password del Usuario Solicitante.
A los efectos de privatizar el acceso existe un area propia donde se guardan aquellos mensajes que fueron aceptados para su transmisión. Este area tiene un tamaño que limita la cantidad de mensajes que el nodo puede almacenar en un determinado momento.
Dado que una lista de distribución puede contener usuarios remotos y locales, cuando estos últimos son reconocidos como válidos este módulo se encarga de efectivizarles el envío. A cada Pedido de envío aceptado le asocia un numero de identificación a través del cual se accederá a la información relacionada con dicho pedido.

c) Consultar al Server

Esta facilidad, permite a un usuario conocer información relacionada con el estado de sus Pedidos de Envío: cola en la que se encuentra, usuarios receptores del mismo, estado del envío, como asi mismo consultar las bases de datos de la red, etc.

3.2 - Función de atención a los requerimientos del Server (ARS)

Esta función preve la activación de módulos diferentes según sea el mensaje recibido del server. Dichos mensajes pueden ser:

a)Acepté un Pedido de Envío

b)Pida el envío de un mensaje a otro Nodo

c)Rechacé un pedido de envío

d)Elimine un mensaje ya procesado.

a) Acepté un pedido de envío

Este mensaje es la indicación de que el server ha verificado la validez de un Pedido de Envío. El nodo recibe del server el número original del Pedido de Envío más un número dado por el server, con el que se identificará el Pedido en la red.
El nodo se encarga, de notificar al usuario correspondiente.

b)Pida el envío de un mensaje a otro Nodo

El nodo recibe del server el aviso para que solicite a otro nodo el envío de un mensaje. En este aviso le llega la longitud del mismo en base a la cual verifica la posibilidad de almacenarlo en el espacio que tiene destinado. Si no tiene lugar suficiente le avisa al server y el proceso de recepción no se lleva a cabo. De existir lugar, desencadena un proceso de **Recepción** que se encarga de efectivizar la recepción.

Dicho proceso pide al server los datos necesarios para el tratamiento del mensaje dentro del nodo, en base a los cuales se establecerá la comunicación con el otro nodo, recibirá el cuerpo del mensaje, calculará el código de verificación de integridad, y una vez verificado su validez notificará al server, y de corresponder distribuirá el mensaje a los usuarios locales destinatarios del mismo.

c)Rechacé un Pedido de Envío

Este mensaje le indica al nodo, que un pedido de envío generado por él ha sido rechazado. El nodo recibe el número de pedido más una lista con la causa del rechazo que puede ser:

- No es un usuario conocido en la red

- No esta autorizado el envío

d)Elimine un mensaje

Esta es la manera en la que el server le notifica al nodo que ya no es necesario que retenga un mensaje porque la lista de distribución ha sido satisfecha. El proceso correspondiente, conociendo el número de mensaje en la red, accede a sus estructuras de datos, y libera el espacio del area destinada para las transmisiones o retransmisiones, eliminando finalmente todas las referencias a este mensaje.

3.3 - Función de Atención a otros Nodos (AON)

Esta función involucra la activación de un proceso de transmisión - de mensaje - a solicitud de un nodo, el que actuará como receptor del mismo.

4 - Particularidades de los procesos Transmisor y Receptor

a)Cantidad de procesos activos en un nodo

La suma de procesos transmisores y receptores activos, en un momento determinado dentro del nodo, esta limitada únicamente por los recursos del sistema. Por lo tanto es posible que un mismo nodo se encuentre relacionado en varias operaciones simultáneas, actuando indistintamente como transmisor y/o receptor.

b)Sincronismo

Una vez establecida la comunicación entre un proceso transmisor y uno receptor, la transmisión del mensaje se realiza en forma sincronizada.
El Transmisor solo puede realizar envíos sucesivos de bloques de información si entre ellos media la correspondiente recepción por parte del Receptor, y recíprocamente.
Este sincronismo permite detectar, en cualquiera de los extremos, la "caida" y generar el aviso correspondiente al server.

c)Mensajes emitidos al server

El receptor es, en situaciones normales, el encargado de avisar al server la finalización de la transmisión.

Los mensajes utilizados son los siguiente:

Mensaje recibido Normalmente, se Conserva para retransmitir

Mensaje recibido Normalmente, No se Conserva para retransmitir

En situaciones anormales los mensajes serán:

Transmisión Cancelada, Falla en el Nodo Receptor

Recepción Cancelada, Falla en el Nodo Transmisor

5 - Mantenimiento

La identificación de un individuo como un usuario de red se genera a partir del pedido presentado por este al administrador de la red, el cual le asocia un nombre único en todo el ambiente. Esta unicidad es controlada por el server. El nombre de usuario de red corresponde a uno o más usuarios del sistema operativo local quienes en definitiva son los beneficiarios de las facilidades brindadas por la red. Cada usuario de red tiene asignada una **Jerarquía**. Esta determina el conjunto de usuarios de red con los cuales puede operar. Finalmente se le asocia a cada usuario de red una palabra clave, necesaria para acceder a cualquiera de las facilidades previstas.

[1] Andrew D. Birrell
Roy Levin
Roger M. Needham
Michael D. Schroeder "Grapevine:An Exercise in Distributed Computing"
Communications of the ACM(1982)

Capítulo 8

Evaluación de Sistemas

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

MACOS: Modelo Algorítmico de Costos de Software.

L. Guerra G, S. Concha C., Depto de Informática.
Universidad Técnica Federico Santa María, Casilla 110-V Valparaíso, Chile.

RESUMEN

El presente trabajo constituye un análisis de los modelos de costos de software, recopilación de datos de proyectos de desarrollo de productos de software, análisis de tal información y finalmente la formulación y evaluación de un modelo algorítmico analítico para determinar costos del software: MACOS.

1.- ANTECEDENTES DE LA MODELACION DE COSTOS DE SOFTWARE.

A partir del estudio de: lo que ha sido la " Modelación de costos de software" hasta hoy, de la clasificación de los distintos métodos de estimación y finalmente de la recopilación y análisis de los distintos modelos de tipo algorítmo encontrados en literatura (1,2,3,4,5,6,7,8,9,10, 11,12), se concluye lo siguiente:

Durante el análisis de los distintos modelos de estimación de costos de software se detectaron las siguientes características comunes a ellos:

Todos los modelos consideran el tamaño del producto como el principal determinante de los costos y el esfuerzo del desarrollo.

La mayoría dimensiona el producto utilizando medidas relacionadas íntimamente con el código producido: Instrucciones Fuentes Entregadas, Instrucciones de Código Objeto, Número y Tipo de Operandos y Operadores utilizados.

La mayoría considera otros factores distintos del tamaño del producto de tal manera de ajustar la estimación del costo o esfuerzo obtenida como función del tamaño. Algunos integran estos factores en una sola función analítica y otros lo hacen a través de multiplicadores que modifican la estimación inicial en base al tamaño.

El problema de todos radica en que no se integran los puntos de vista del desarrollador y del usuario en la estimación. Por ejemplo, COCOMO (11) tan sólo considera factores que son perceptibles por el desarrollador; por otro lado, la técnica Puntos de Función (12) tan soló incluye aquellos perceptibles por el usuario.

Además de estas características, es importante destacar los dos hechos siguiente:

Se han hecho algunos intentos con modelos que no dimensionan el producto usando medidas relacionadas íntimamemte con el còdigo escrito. Estos usan medidas altamente subjetivas: Número de Líneas de Especificación, Número de Funciones, etc.

Un estudio de Kitchenham y Taylor (13) acerca del modelo COCOMO y la distribución del esfuerzo y tiempo en las distintas fases del desarrollo concluyó que los resultados de aquel son del mismo orden que el esfuerzo y tiempo reales y que, en promedio, los porcentajes de esfuerzo y tiempo dedicados a cada fase son muy similares a los determinados en COCOMO pero que cada proyecto individual varíe mucho respecto del promedio.
Lo anterior se tomó como base para no realizar una distribución (en términos porcentuales) de esfuerzo y tiempo en las fases de desarrollo.

Los hechos anteriores y las características mencionadas llevaron a establecer las siguientes bases para establecer un modelo.

1.- Considerar el tamaño del producto como el principal determinante de los costos de desarrollo de software. Esta consideración a priori fue comprobada posteriormente mediante una ronda de consultas con expertos.

2.- Dimensionar el producto con alguna medida independiente del código de tal forma de permitir una mejor estimación del tamaño en fases tempranas del desarrollo.

3.- Integrar en el modelo a desarrollar los puntos de vista del desarrollador y del usuario a través de otros factores distintos del tamaño que fueran perceptibles para cada uno.

4.- Considerar estimaciones globales (para el desarrollo como un todo) de esfuerzo y tiempo.

2.- FACTORES CONSIDERADOS "A PRIORI EN MACOS.

Sobre la base de las conclusiones obtenidas anteriormente se presentan los factores que se consideran en MACOS como los principales determinantes de los

costos y el esfuerzo de desarrollo. El conjunto de factores considerados es el resultado del análisis de todos aquellos incluídos en los modelos existentes, de la realización de una encuesta (que resulta ser muy interesante de analizar a fondo) a personas que han estado mucho tiempo relacionada con el desarrollo de software en Chile y de la realización de un experimento que permitió determinar la medida de dimensionamiento del tamaño del producto de software (considerado el principal determinante de los cotos y el esfuerzo).
Se definieron algunas características que debiera tener una medida del tamaño del producto:

Significativa al usuario no técnico.
Independiente de la tecnología de desarrollo de software.
Servir como estimador.
Objetiva.
Disponible antes de la "fabricación" del producto".
Contabilizada automáticamente.

Analizando estas características en los siguientes candidatos:

Código objeto.
Código fuente.
Instrucciones categorizadas.
Puntos de Función.
Métricas de Halstead.

se concluyó que la medida "Puntos de Función" cumplía cabalmente con las características 1, 2 y 5. Respecto de la característica 6, los "Puntos de Función" no se pueden contabilizar automáticamente.

En lo relacionado con la característica 3 se estableció:

Los "Puntos de Función" pueden ser estimados razonablemente bien en fases tempranas del desarrollo.
Existe correlación entre la medida y el esfuerzo (en HH, por ejemplo) dedicado al desarrollo.

Albretch (12) definió los puntos de función considerando, además del conteo, una complejidad asociada a cada Entrada, Salida, Archivo Lógico Interno, Interfase y Consulta que componen el producto de software. La razón de asociar esta complejidad es cuantificar el procesamiento asociado con cada función (y por ende el código asociado a ellas).

En el modelo MACOS se optó por considerar aparte los aspectos relacionados con la complejidad de procesamiento, es decir, los puntos de función sólo se consideran como una medida del tamaño del producto y no una forma de cuantificar el procesamiento asociado con ellos. Se han eliminado además las ponderaciones asociadas con cada nivel de complejidad (Albretch explica estas ponderaciones como la importancia que tiene la función para el usuario y son determinadas en base a prueba y error).

Esta decisión hace que la contabilización de los Puntos de Función sea mucho más sencilla y requiera menos tiempo. Por lo que se soluciona la principal desventaja de la medida: la dificultad de su contabilización.

Por otro, lado, se disminuye la subjetividad de la medida al no tener que asociar a cada función un grado de complejidad. Respecto de esta última característica (la número 4.-) quedan dudas aún y es necesario estudiar más acerca de ello para disminuir esta subjetividad lo más posible; es probable que a nivel de instalación sea factible el definir más acabadamente que se entiende por "Puntos de Función" (definiendo estándares) y de esta manera disminuir su subjetividad.
Se definieron otros factores que inciden en el esfuerzo dedicado al desarrollo de software, a saber:

- INTUSU :Interfase con el Usuario.
- ESPREQ :Especificación de requerimientos.
- RECONT :Plan de Recursos y control del proyecto.
- PLAPRO :Plan Detallado del Proyecto.
- TESTVV :Plan Detallado de Testing y Verificación y Validación
- AMBTRA :Ambiente de Trabajo.
- DOCEXI :Documentación Existente.
- CONPER :Continuidad del Personal.
- CALADM :Calidad de la administración
- DATARC :Características de Captura, Validación Actualización y Almacenamiento de Datos.

- FAFLCA :Facilidad de Cambio y Flexibilidad.
- CONPRO :Confiabilidad del Producto.
- RESTPO :Restricciones en el Tiempo de Desarrollo.
- MODDOP :Modo de Desarrollo y Operación del producto.
- EXUSUA :Experiencia del Usuario.
- CAPAN :Capacidad de los Analistas.
- EXPANA :Experiencia de los Analistas.
- CAPPRO :Capacidad de los programadores.
- EXPLEN :Experiencia en el lenguaje.
- RAPCPU :Restricciones de A.P. y CPU.
- TRESPC :Tiempo Respuesta Promedio del Computador.
- HERRAD :Herramientas de Ayuda al Desarrollo.
- PMOPRO :Prácticas Modernas de Programación.
- COMPLP :Complejidad del Producto.
- FAICMA :Facilidad de Instalación,Conversión y Mantención.

3.- RECOPILACION DE DATOS.

Para poder establecer el modelo MACOS se recopiló la información necesaria para ello: acerca de los factores determinados y de otras características de un proyecto de desarrollo: costos, esfuerzo, horas de computador, nivel de documentación, etc.

Durante la recopilación de datos se detecto que, en general, las distintas empresas del área no llevan un registro formal del proceso de desarrollo de software. Esto llevó a tener que trabajar con datos que provenían del "recuerdo" de las personas por lo que, desde el punto de vista de su fidelidad,es posible que existan algunas dudas.

La principal conclusión de la recopilación de datos es, entonces,la necesidad de contar con un registro formal del desarrollo de proyectos de software.
Lo anterior significa que el registro debe ser fácil de llevar a cabo pues de otra manera significaría distraer el esfuerzo en la tarea principal que es el desarrollo.

4.- ANALISIS DE LA INFORMACION RECOPILADA.

Además de la recopilación de datos, se hizo necesario realizar una serie de análisis de ella para obtener algunas conclusiones acerca de las relaciones existentes, por ejemplo entre: esfuerzo y tamaño del producto, horas de

computador y tiempo de desarrollo, etc. Todas estas relaciones se integran en el modelo MACOS.
Durante el análisis de los datos, considerando la cantidad de información recopilada y la fidelidad de información, se determino que.

Existe correlación entre PF y esfuerzo.

Por otro lado, es interesante el aspecto de la productividad del desarrollo. Es destacable el hecho de que ésta permanezca como una constante no dependiente del lenguaje ni del tamaño del producto.

Se determinó que los montos de documentación y de tiempos de computador varían mucho entre los distintos sistemas, por lo que se confirma lo planteado por Boehm (11) respecto de estas medidas.

En relación a la distribución del esfuerzo y tiempo en las distintas fases del desarrollo y a los factores distintos del tamaño del producto, no fue posible analizar sus comportamientos debido a la falta de información acerca de ellos.

5.- EL MODELO MACOS.

De acuerdo a las conclusiones de las actividades precedentes se presenta el modelo MACOS con sus definiciones y suposiciones, la calibración del modelo a una realidad particular y otros aspectos adicionales.

1.- MACOS es un modelo de estimación de tiempos de costos de software de tipo algorítmico. Considera la siguiente ecuación principal:

$$ENE = 3.55 * PF \hat{} 1.35 \qquad (1)$$

La ecuación (1) entrega una estimación inicial o nominal del esfuerzo (ENE) medido en Horas Hombre (H-H).

2.- Una Hora - Hombre se define como :
"El trabajo de una persona durante una hora considerando overhead.

Se elige H-H como medida del esfuerzo del trabajo, a cada H-H se le asocia un Costo Promedio que permite determinar el Costo Total de desarrollo del proyecto.

3.- El principal determinante de los costos es el tamaño del producto, medido en Puntos de Función (PF). Los Puntos de Función dimensionan un producto de software contabilizando el número de :

- Entradas del usuario externo.
- Salidas al usuario externo.
- Consultas del usuario externo.
- Archivos maestros desde el punto de vista lógico del usuario.
- Intefases con otros programas de aplicación.

La suma de las cantidades de cada uno de los ítems anteriores es el número de PF.

4.- Las fases cubiertas por la estimación son:

Conocimiento del problema.
Definición de requerimientos.
Análisis (Diseño Lógico, Confección de Modelo de Datos).
Diseño físico (Construcción de la estructura computacional, definición del modelo físico de datos).
Programación (Implementación y Pruebas, Construcción).
Test, Integración, Puesta en marcha, Documentación final.

El tiempo del desarrollo (TDES) se mide en Horas. La principal razón para ello es evitar al máximo los problemas que provocan las diferencias entre las horas trabajadas por mes (al usar la medida del esfuerzo Hombres-Mes). Estas diferencias se deben principalmente al trabajo en horas extraordinarias y a la dedicación parcial de alguna de las personas trabajando en el proyecto.

La estimación inicial o nominal de esfuerzo (EN) dada por la ecuación (1) es ajustada por un Factor de Ajuste (FA) definido por:

$$FA = 1 + X/100 + y/100$$

X es la suma de todos los Grados de Influencia (GI) positivos; aumento de los costos.

y es la suma de todos los Grados de influencia (GI) negativos; disminución de los costos.

Donde los Grados de Influencia (GI) definen la incidencia de los factores ya mencionados.

Los GI son :

DISMINUCION DE LOS COSTOS.

Influencia muy importante	--->	- 5
Influencia significativa	--->	- 4
Influencia promedio	--->	- 3
Influencia moderada	--->	- 2
Influencia incidental o muy baja	--->	- 1

SIN EFECTO

No presente o sin influencia	--->	0

AUMENTO DE LOS COSTOS

Influencia incidental o muy baja	--->	1
Influencia moderada	--->	2
Influencia promedio	--->	3
Influencia significativa	--->	4
Influencia muy importante	--->	5

El producto del EN por el FA determina el Esfuerzo Ajustado (en H-H):

EAE = ENE * FA

El EAE representa el esfuerzo que se estima, debe dedicarse al desarrollo del producto considerado.

Con el EAE se utiliza

TDESE = EAE * NTPC

donde:

NTPC es el Número Promedio de Personas de Tiempo Completo trabajando en el proyecto.
TDESE es el Tiempo de Desarrollo Estimado.

Para estimar el Tiempo de Desarrollo (TDESE) requerido.

El EAE multiplicado por una constante que indica el costo por hora-hombre, entrega la estimación del costo para el desarrollo del proyecto.

No se hace distribución del esfuerzo y el tiempo sobre las fases de desarrollo debido a las razones dadas anteriormente.

6.- CONCLUSIONES.

Es necesario revisar varios aspectos del modelo planteado : relación esfuerzo con PF, definición de los PF, relación EAE/tiempo de desarrollo, integración de los factores distintos del tamaño al modelo, definición del FA , etc.
A pesar de lo anterior, el Error Promedio de Estimación de COCOMO es muchisimo más alto que el de MACOS.
Lo anterior significa que COCOMO no se ajusta a la realidad del desarrollo de software en Chile, por lo que definir un modelo como MACOS permite hacer estimaciones realistas en nuestro ambiente.

Es importante destacar que MACOS, debido a los proyectos considerados, está orientado a sistemas administrativos de tamaño pequeño a mediano (basándose en el número de líneas de código fuente de los proyectos considerados). Sera interesante estudiar su aplicabilidad en otro tipo de sistemas.

La concepción de MACOS pretende suplir las fallas encontradas en otros modelos. Muchas de sus características son una primera aproximación a la solución de tales problemas . Lo anterior significa que aún queda muchisimo por hacer.

- Falta una definición más acabada de la medida del dimensionamiento. En este tema sería interesante buscar alternativas aunque Puntos de Función es un buen candidato.

- Es necesario definir mejor la integración de los factores distintos del tamaño. Para ello se recomienda "dimensionar" la influencia de estos factores de manera multiplicativa y además definir un rango en que el "ajuste" sea válido pues, fuera de ese rango es probable que ese rango no sea válido

- Debe definirse otra relación para estimar el Tiempo de Desarrollo.

- Falta una definición completa de las Fases de Desarrollo (y sus actividades) cubiertas por la estimación.

- Si es posible, debiera obtenerse los datos a medida que el proyecto avanza. Esto se hace muy difícil debido a los problemas de restricción en el tiempo de desarrollo a que se enfrentan en general los proyectos de software.

- Lo anterior significa que es necesario definir procedimientos automáticos para recopilación de datos pues, de otra manera, es prácticamente un hecho que se dejará de lado el proceso de recopilación.

7.- REFERENCIAS.

1.-Quantitative analysis of programming cost factors: a progress report. L. Farr , H. J. Zagorski, A.B.Frielnik, 1965.

2.-Management handbook for estimation of computer programming cost. E. A. Nelson, Sistems Development Co.,1967.

3.-Estimating resources for large programming systems J. D Aron, NATO Science Comittee, Rome, 1969.

4.-The cost of developing large-scale software. R. W. Wolverton, IEEE Trans. Computers, 1974

5.-A macro-estimating methodology for software development. L. H. Putnam.Proceedings IEEE. COMPCON 76, 1976.

6.-Prediction of software effort and project duration : four new formulas. V. Schneider. ACM SIGPLAN notices, 1978.

7.-Software cost measuring and reporting, Boening Co. , U. S. Air Force, ASD Document, 1979.

8.-System life cycle estimation (SLICE): A new approach to estimating resources for aplication program development. A. L. Kustanowitz, IEEE COMPSAC, 1977.

9.-A method of programming measurement and estimation. C. E. Walston, C. P. Felix, IBM Systems Journal, 1977.

10.-Estimation of computer sequirements and software development cost. Taback and Ditmore General Research Co. , 1974.

11.-Software Engineering Economics. Barry W. Boehm Prentice Hall, 1981.

12.-Measuring Application development productivity. A. J. Albrecht. SHARE-GUIDE, 1979.

13.-Software Proyect development cost estimation. B. A. Kitchenham, N. R. Taylor. The Journal of Systems and Software , 5, 1985.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

MODELO DE DESEMPEÑO DE UN SISTEMA COMPUTACIONAL UNIVERSITARIO

Francisco Ovalle
Francisco Aurtenechea
Ignacio Casas

Pontificia Universidad Católica de Chile
Escuela de Ingeniería - Depto. Ciencia de la Computación(143)
Casilla 6177 , Santiago, Chile
Teléfono: 56-2-555-0058 @4440
Telex: 240395 PUCYA CL Fax: 56-2-5510046
e-mail: jfovalle@puc.cl ó ...uunet!uchdcc!juncal!jfovalle

Resumen

En este trabajo se presenta un estudio de modelación y evaluación de un sistema computacional centralizado en operación en un ambiente universitario. El modelo implementado está basado en la tecnología de modelos analíticos de redes de espera.

Un sistema computacional universitario tiene varias características que lo hacen distinto a otros sistemas en operación en empresas. Esto se manifiesta principalmente en lo heterogéneo de las horas de su uso, y en la variedad de los tipos de procesos que se ejecutan. En contraste, en una empresa generalmente existen uno o dos lenguajes de desarrollo y unos pocos procesos que acumulan en sí la mayor parte de la carga. En un ambiente universitario, la carga en el sistema está formada mayoritariamente por ejecuciones de programas personales de cada alumno. Estos procesos son difíciles de monitorear separadamente. Además, en el ambiente universitario se utilizan múltiples lenguajes de desarrollo y programación.

En este artículo se presenta una metodología para realizar la modelación de sistemas computacionales en ambientes universitarios, de tal forma de hacer más expedita la recopilación y análisis de la información, a través de su automatización y obtener los patrones de demanda de las aplicaciones más típicas como editores y compiladores.

Se muestran además las mediciones realizadas al sistema y la construcción de diversos modelos analíticos capaces de responder preguntas planteadas tanto ante cambios en el hardware del equipo como en la carga de trabajo. Se describe también el software utilizado para la implementación de los modelos y su resolución analítica.

1. INTRODUCCION

En los últimos años se ha hecho cada vez más necesario el encontrar herramientas y técnicas que permitan comprender el comportamiento de los sistemas computacionales y predecir la evolución de la carga a que se ven sometidos, para poder responder a las preguntas de costo y desempeño que aparecen a través de la vida del sistema. El gran número de decisiones que deben ser tomadas a lo largo de las distintas etapas de análisis, desarrollo y puesta en marcha del sistema afectarán directamente su desempeño, pero el impacto de esas decisiones es difícil de predecir, debido a la complejidad de las alternativas (trade-offs) de dimensionamiento, diseño e implementación. De igual forma, es difícil planificar la forma en que debe crecer la capacidad de procesamiento del sistema, en función de la evolución de la demanda de servicio (nuevas aplicaciones, aumento de la tasa de llegada de transacciones, etc.).

Todas las preguntas que aparecen al realizar estas planificaciones son de gran importancia para las organizaciones afectadas y de las cuales se pueden derivar serias repercusiones si las decisiones son incorrectas. Desafortunadamente estas preguntas son complejas y las respuestas correctas son difíciles de obtener.

Actualmente se están desarrollando nuevas técnicas de modelación, basadas en modelos analíticos de redes de espera, que pueden ser utilizadas para ayudar en la evaluación y predicción del desempeño de sistemas computacionales ([Grah78], [Saue81], [Lazo84], [Casa86]). El desempeño de un sistema se evalúa, en estos modelos, en base al caudal o número de procesos por unidad de tiempo, tiempos de respuesta para transacciones típicas, utilización de centros de servicio (CPU, discos, canales, etc.) y largo de colas en los distintos centros.

Esta metodología permite predecir el impacto en el desempeño de realizar modificaciones tanto en el hardware del equipo como en la carga de trabajo. Entre los primeros podemos nombrar el cambio de CPU por otra de mayor velocidad, o bien colocar una segunda CPU, agregar o quitar discos, etc.; entre los segundos tenemos la redistribución de los archivos en los discos, incrementos en la tasa de llegada de transacciones, etc. De igual forma, nos lleva a la identificación de los tipos de proceso que consumen más recursos, identificación del "cuello de botella" (unidad más cargada del sistema) y la utilización de las unidades.

En este artículo se presenta un estudio de evaluación y predicción del desempeño, por medio de técnicas basadas en modelos de redes de espera, para un sistema computacional en operación en la Pontificia Universidad Católica de Chile, el cual, por estar inmerso en un ambiente universitario, tiene características muy especiales de carga de trabajo (mayores detalles pueden verse en [Oval88]).

El sistema evaluado consta de un computador VAX/VMS 8600 con 20 MBytes de memoria principal, 2 impresoras y 116 terminales; y un computador VAX/VMS 750 con 8 MBytes de memoria principal y 24 terminales. Estos dos computadores están unidos en "cluster", compartiendo de este modo 7 discos del tipo DEC RA81.

En la actualidad este sistema da muestras de encontrarse cerca de la saturación, en especial durante días y horas peak, como son dos o tres días antes de la entrega de una tarea de algún curso, lo que ocurre muy frecuentemente.

Un sistema computacional universitario tiene varias características que lo hacen distinto a otros sistemas en operación en empresas. Esto se manifiesta principalmente en lo heterogéneo de las horas de su uso (prácticamente no se usa durante la noche, a no ser en trabajos batch), y en la variedad de los tipos de jobs que se ejecutan. En una empresa generalmente existen

uno o dos lenguajes de desarrollo y unos pocos procesos que acumulan en sí la mayor parte de la carga.

En cambio, en un ambiente universitario, la carga en el sistema está formada mayoritariamente por ejecuciones de programas personales de cada alumno. Estos procesos son difíciles de monitorear separadamente, debido a qué los archivos de monitoreo crecerían en una forma abismante y prohibitiva. Además, en el ambiente universitario se utilizan múltiples lenguajes de desarrollo y programación, lo cual hace que la carga de trabajo en el sistema sea aún más heterogénea.

En la sección 2 se presenta una breve introducción a los modelos de desempeño basados en redes de espera y su utilización para la predicción del desempeño. En la sección 3 se presentan estadísticas de utilización de CPU y discos, tiempo de CPU de cada tipo de proceso, número de I/O a cada disco, etc. y se muestran los modelos analíticos usados.

En la sección 4 se muestra la caracterización de la carga de trabajo y la validación de los modelos y en la sección 5 se presenta el análisis de las modificaciones introducidas, donde se utiliza el modelo para predecir el desempeño del sistema frente a cambios tanto de hardware como de carga de trabajo.

2. MODELOS DE DESEMPEÑO

En esta sección se describen los modelos de desempeño basados en redes de espera, se resume la evolución que ha experimentado esta técnica, sus componentes básicos y su relación con los componentes de un sistema computacional. Se explica como se obtienen los distintos parámetros del modelo a partir de información de monitoreo obtenida durante un período de observación del sistema y finalmente se indica como se utiliza el modelo para la predicción del desempeño frente a eventuales variaciones de la carga de trabajo y mejoras en las componentes de hardware del sistema.

2.1 ¿Qué es un Modelo de Red de Espera?

Un Modelo de Red de Espera es una forma particular de enfrentar la modelación de un sistema computacional, donde el sistema es representado como una red de colas [Lazo84], la cual es evaluada en forma analítica.

La modelación de redes de espera puede distinguirse de las ramas más tradicionales de la teoría de colas por el hecho de que un sistema es representado no ya como un punto de congestión único, sino más bien como un grupo de servidores (y sus colas) interconectados, los cuales corresponden a los recursos individuales del sistema por donde circulan clientes que requieren servicio.

2.2 Componentes y definición de un Modelo de Red de Espera

Un modelo de red de espera está formado por : **centros de servicio** que representan los recursos de sistema, tales como procesadores (CPU's) y dispositivos en general (discos, canales, controladores), en los que se produce congestión; **clases de procesos** que representan las tareas que poseen características sismilares de consumo de recursos y **dominios** que indican competencia por recursos (como la memoria principal) de las tareas dentro de una clase y entre distintas clases. ([Casa87], [Lazo84], [Oval88]).

Para definir un sistema computacional dado se deben identificar todos los centros de servicio, las clases de proceso y sos respectivos dominios y obtener los valores de los parámetros

asociados a cada componente, los cuales incluyen las características de servicio de los centros y la carga de trabajo impuesta por las clases de procesos en cada centro de servicio.

2.3 Construcción y uso de un modelo de desempeño

El estudio de modelación de un sistema computacional en operación a través del empleo de un software de evaluación de redes de espera (por ejemplo: MAP [Lazo84]) comprende tres etapas:

(1) **Creación de un modelo base:** el modelo base debe representar fielmente al sistema actual, con una carga de trabajo promedio para los distintos intervalos de observación considerados para las mediciones del sistema. Para la obtención de este modelo se debe realizar una medición del sistema computacional, mediante un monitor de sistema, durante períodos de carga significativa; definir las componentes del modelo (centros de servicio, clases de procesos y dominio de memoria), identificando aquellos procesos que tienen un mayor consumo de recursos y finalmente utilizar las mediciones entregadas por el monitor del sistema para dar valores a los parámetros de los componentes del modelo.

(2) **Validación del modelo base:** una vez completado el proceso anterior, la definición de componentes y sus parámetros son ingresados al software de evaluación de redes de espera para obtener estimaciones de desempeño previamente descritas. Enseguida, las estimaciones del modelo son comparadas con las mediciones recogidas por el monitor del sistema durante el período de observación. El modelo base se considera validado una vez que las estimaciones proporcionadas por el modelo coinciden (con un margen de error poco significativo) con las mediciones reales observadas directamente en el sistema.

(3) **Análisis de modificaciones o proyecciones:** Una vez que se obtiene un modelo base validado, se procede a efectuar cambios en la definición de componentes del modelo y sus parámetros y obtener del software de modelación las estimaciones del futuro desempeño correspondientes (¿ qué pasa si agrego una segunda CPU o si agrego n terminales o si aumenta la tasa de llegada de un cierto tipo de proceso?). Las modificaciones al modelo utilizando el software de modelación son fáciles de realizar y las mediciones estimadas son entregadas en cuestión de segundos.

3. MEDICIONES REALIZADAS Y CONSTRUCCION DE MODELOS ANALITICOS

En esta sección se describe la forma en que se obtuvieron las mediciones de los distintos computadores y se presentan los modelos desarrollados.

3.1 Mediciones en el sistema

Para realizar las mediciones necesarias en el estudio de modelación, se recurrió a monitores existentes en los equipos: el SPM (Software Performance Monitor) que es un set de herramientas que permiten reunir información sobre la carga de trabajo y realizar reportes de las estadísticas de desempeño de sistemas VAX/VMS; y el Accounting Utility que entrega información de los procesos individuales, para fines de facturación.

En primer instancia se procedió a realizar monitoreos para determinar períodos y horas peak (ver gráfico 3.1), ya que las mediciones para la obtención del modelo deben hacerse en un período significativo que incluya una buena muestra de todos los tipos de proceso. Típicamente se considera un período de 1 o 2 horas de duración, cuando la carga alcanza valores peak

(generalmente interesa observar el sistema bajo condiciones de fuerte carga y no cuando el sistema está siendo poco utilizado [Casa87]). Por tanto, se decidió realizar el estudio durante un período de 6600 seg. (aproximadamente 2 horas) de un día típico.

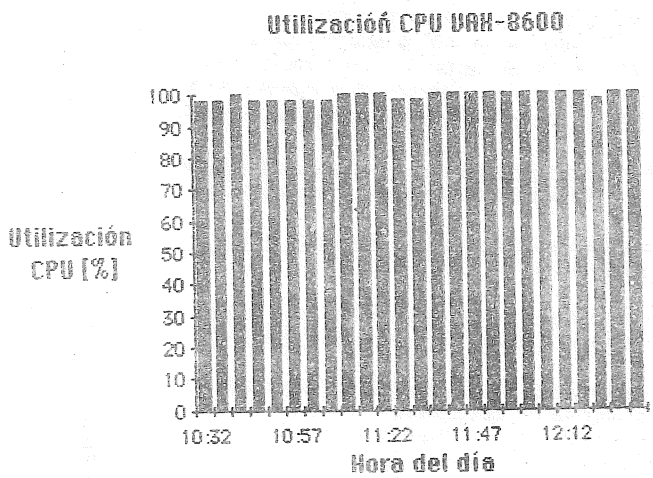


Grafico 3.1 - Utilización CPU Computador VAX-8600

Durante dicho período se realizaron monitoreos con las dos herramientas disponibles (SPM y Account). Esto permitió realizar tanto un monitoreo por tipos de usuario, como por proceso, además de información general del estado del computador. (Cuando se hable de tipos de proceso nos referimos a compilaciones, comandos DCL, editores, etc. ; mientras que los tipos de usuario se refieren, por ejemplo, a Ingeniería, usuarios externos, Medicina, etc.).

Las mediciones por tipo de usuario en el computador VAX-8600 pueden resumirse en la tabla 3.1 del anexo (las mediciones en el computador VAX-750 son análogas) donde se pueden apreciar el tiempo de CPU (en segundos), el número de transacciones, utilización total de cada disco durante el período de observación, intervalo de muestreo y número de I/O de cada clase en cada disco.

La obtención del número de I/O de cada clase en cada disco merece una explicación detallada debido a que el monitor no entrega esta información directamente. El monitor sólo indica el número total de I/O que cada usuario o cada proceso, realiza en el subsistema de discos. Por otro lado, el monitor indica el número total de I/O's realizados en cada disco, sin discriminación de usuarios o procesos. Por esta razón, la distribución de accesos por usuario en los discos se hizo averiguando en que disco se encontraba residente cada tipo de usuario.

Debido a que no es posible discriminar el porcentaje de recursos de cada tipo de usuario a cada disco, se consideró que cada grupo realizaba un acceso al disco del sistema un 40% de las veces, al disco donde se encontraba residente el 50% y al disco de apoyo (SCRATCH o APOLO según tipo de usuario) un 10%.

Se realizaron alteraciones en los porcentajes de algunas de las clases debido a que se conocía el comportamiento de algunos de los usuarios.

Otro punto interesante está en la distribución de los accesos físicos a disco: por un lado se conocía el total de accesos a cada disco (sin discriminación de usuarios) y por otro lado se conocía la suma total de accesos al subsistema de discos de cada una de las clases. Esta última suma era mayor que la primera, por lo cual se concluyó que los "direct I/O" (accesos a disco) que arrojaba el monitor account por usuario no eran accesos físicos como lo advierte la literatura [Lazo84]. Por tanto se rebajó la cantidad de I/O de cada clase de acuerdo al factor Gc (definido en [Lazo84] como la razón entre los I/O's físicos y lógicos de la clase "c")

$$Gc = \frac{\text{I/O físicos de la clase c}}{\text{I/O lógicos de la clase c}} = \frac{\text{\# total de accesos a disco}}{\Sigma \text{ accesos clase c}}$$

(el supuesto de que la razón Gc depende de la clase pero no del dispositivo es realista en la mayoría de los sistemas).

De la misma forma, en la tabla 3.2 del anexo se pueden ver las mediciones por tipo de proceso en el computador VAX-8600 (las mediciones del VAX-750 son análogas).

La obtención de los accesos a disco por cada tipo de proceso, difiere un tanto del método expuesto en el punto anterior, ya que ahora se carece de toda información que pueda orientar sobre la forma en que se distribuye la utilización de cada disco por parte de cada tipo de proceso. Para estimar el número de I/O de cada clase de proceso en cada disco se recurrió a las siguientes estadísticas de monitoreo: el número de I/O totales en cada disco y el número de I/O totales de cada clase. Dados estos datos, se asumió que el número de accesos de cada clase a cada disco era una proporción del número de accesos totales a cada disco [Lazo84]. Los resultados para el VAX-8600 se muestran en la tabla 3.3 del anexo.

3.2 Construcción de modelos

A continuación se describen los modelos de desempeño implementados utilizando el software MAP.

Se realizaron 3 modelos para el cluster de computadores VAX-8600/VAX-750 : dos modelos abiertos (transaccionales) que no incluyen explícitamente el manejo de terminales (figura 3.1), llamados MODELO 1 y MODELO 2 y un modelo mixto donde se cuenta con una clase interactiva (figura 3.2), llamado MODELO 3. En el modelo 1 la carga se representa por clases de usuarios, mientras que en los modelos 2 y 3 la carga de trabajo se representa por clases de procesos.

En todos estos modelos se representaron explícitamente, como componentes de hardware, las CPU (VAX-8600 y VAX-750) y los 7 discos RA81.

La memoria principal impone dos tipos de restricciones al procesamiento de trabajos. El primer tipo de restricción está relacionado con el número máximo de procesos que pueden ser ejecutados simultáneamente, es decir, el nivel máximo de multiprogramación. La memoria principal puede ser compartida en un instante dado, por un número limitado de procesos. Aquellos procesos que no puede ser atendidos, deben esperar (estado "wait") a que se libere alguna partición de memoria para entrar en ejecución. El segundo tipo de restricción de la memoria se refiere a las actividades de swapping y paginación, las cuales interfieren cuando la memoria es inadecuada en el procesamiento normal de un trabajo, aumentando los tiempos de respuesta.

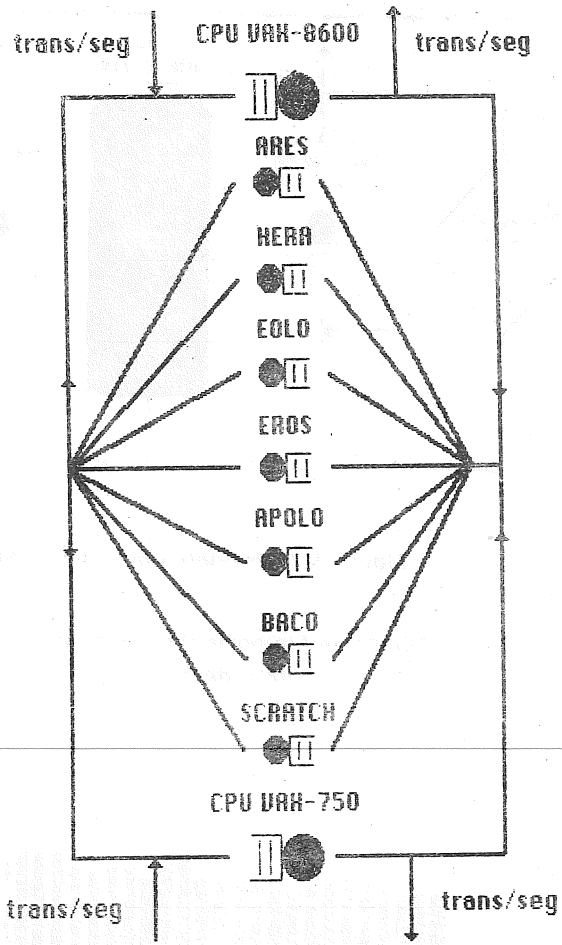


FIGURA 3.1 - Modelo abierto sistema VAX-8600/VAX-750

De las mediciones de monitoreo (ver Gráfico 3.2) se observó que la carga del sistema no posee restricciones de nivel de multiprogramación, ya que la memoria principal se ocupa en un bajo porcentaje (alrededor del 60%). Es decir, no se forman colas de procesos en espera de particiones de memoria. Por esta razón, en los modelos no se incluyen colas de memoria ni tampoco dominios de memoria para las clases de procesos.

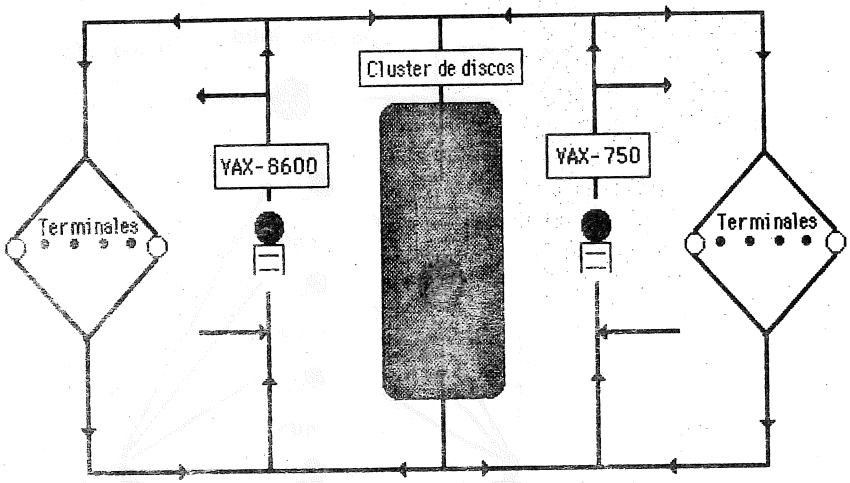


Figura 3.2 - Modelo mixto VAX-8600/VAX-750

**Utilización Memoria (Porcentaje) vs.
Hora del día**

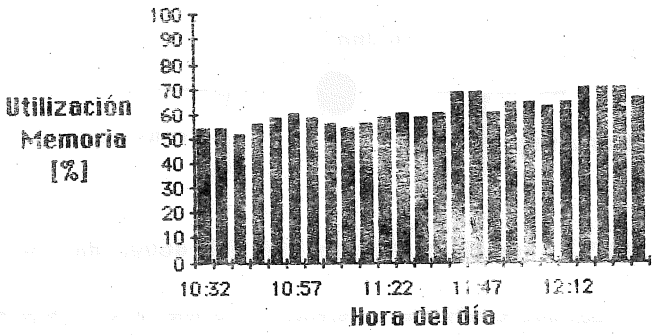


Gráfico 3.2 - Utilización Memoria Computador VAX-8600

Por otra parte, aunque el swapping durante las horas peak es prácticamente inexistente, las actividades de paginación y swapping se representan implícitamente en el modelo al incluir el overhead debido al sistema operativo.

4.CARACTERIZACION DE LA CARGA DE TRABAJO Y VALIDACION

A partir de las mediciones realizadas descritas en la sección anterior, se describe la metodología de obtención de los parámetros que caracterizan la carga de trabajo del sistema computacional.

Estos parámetros son los valores de las tasas de llegada, consumo de CPU y demandas de servicio de discos para las distintas clases de procesos.

4.1 Porcentaje de uso de CPU

Si se compara la suma del uso de CPU por cada uno de los procesos con el uso de CPU global del sistema, se puede notar que existe una diferencia, la que está dada por el overhead que no es asignable a procesos en particular. De esta forma, se tendrán básicamente dos opciones: considerar esta carga como debida a una clase especial, (la clase overhead) o bien, distribuir este overhead porcentualmente de acuerdo al uso de CPU en cada una de las clases de procesos consideradas en la modelación. En esta modelación se procedió a considerar la segunda opción, ya que aparte de ser más representativa de la realidad, facilita el proceso involucrado durante la fase de análisis de modificaciones en el modelo (sección 5).

4.2 Tasa de llegada de transacciones

La tasa de llegada es una medida de la cantidad de requerimientos (transacciones) que llegan al sistema por unidad de tiempo. Para estimar este parámetro se midió el total de transacciones de cada clase de procesos, en el período de observación.

La tasa de llegada promedio para una clase, en transacciones por segundo, está dada por el total de transacciones de esa clase dividido por el largo del intervalo de monitoreo.

Los valores de la tasa de llegada para cada clase en cada uno de los modelos fueron obtenidos directamente del monitor y se incorporan como datos de entrada al modelo correspondiente.

4.3 Demanda de servicio en los discos

Una vez que se obtuvo el número de I/O de cada clase en cada disco, de acuerdo a las metodologías expuestas con anterioridad, se procedió a calcular la tasa de acceso de cada clase en cada disco por el simple expediente de obtener el porcentaje que representa el total de direct I/O de la clase en el disco, del total de direct I/O realizados a dicho disco.

Luego se calculó el tiempo total de uso de cada disco por cada clase como el producto entre el valor de la tasa de acceso de cada clase en cada disco, recién calculado, la utilización del disco y el largo del intervalo de muestreo.

Finalmente, la demanda de servicio en cada disco impuesta por una instancia de una clase dada, se calcula dividiendo lo anterior por el total de transacciones realizadas por esa claseen el período de observación. Por tanto,

$$Dc,d = (d * Ud * T) / trc,$$

donde :

Dc,d : Demanda en disco d de la clase c [seg/transacción]

d : Tasa de acceso de cada clase en cada disco(entregada por monitor)

Ud : Utilización del disco d (entregada por monitor)
T : Largo del intervalo de muestreo [seg]
trc : Número total de transacciones de la clase c [transacciones]

4.4 Demanda de servicio de CPU

La demanda de CPU por cada clase está dada por el cuociente entre el tiempo de procesador ocupado por la clase y el número de transacciones de la misma.

La suma de los tiempos de procesador de todas las clases dividido por el intervalo de muestreo entrega la utilización global de la CPU, que es un dato entregado por el monitor.

4.5 Parámetros de los modelos

En las tablas 4.1 y 4.2 del anexo, se muestran los parámetros calculados para los modelos transaccionales por tipo de usuario y por tipo de proceso de acuerdo a las diferentes clases de proceso.

Para el modelo mixto, se consideró interactivo todo aquello que no fuese compilaciones, ediciones, debugging, procesos submit, algunos programas como el minitab y aquellos programas desarrollados por los usuarios que en general, salvo contadas excepciones, carecen de la característica de ser interactivos.

Para el cálculo del tiempo de pensamiento, Z, de las terminales, cuyo número promedio era de 14 para el VAX750 y de 20 para el VAX8600, se procedió a ocupar un modelo transaccional con las mismas clases que tendría el modelo mixto, introduciendo este modelo a MAP, se obtiene como resultado, entre otros datos, el tiempo de respuesta para las clases interactivas.

Teniendo el tiempo de respuesta, se procedió a calcular los tiempos de pensamiento por medio de la Ley de tiempo de respuesta [Lazo84] :

$$Z = (N / X) - R \quad \text{donde}$$

Z : Tiempo de pensamiento (entre transacciones)
N : Número promedio de terminales activas
X : Caudal (trans/seg)
R : Tiempo de respuesta

De esta forma, se obtuvo un tiempo de pensamiento promedio para el VAX8600 de 35.05 seg y de 127.49 seg para el VAX750, lo que es bastante razonable debido a las características de usuarios de ambos computadores (el VAX750 está destinado fundamentalmente a alumnos que durante el desarrollo de programas tienen un tiempo de pensamiento mayor, mientras que el VAX8600 está destinado a trabajos externos, docentes e investigaciones).

Los parámetros calculados para el modelo mixto, pueden verse en la tabla 4.3 del anexo.

4.6 Validación de los modelos

En este punto se presentan resúmenes de los informes de desempeño entregados por el software MAP al evaluar los modelos descritos anteriormente.

4.6.1 Modelo transaccional por tipo de usuario

La tabla 4.4 muestra el porcentaje de utilización de las CPU's y discos, como asimismo el largo promedio (número de transacciones) de las colas de espera en cada centro de servicio. La tabla muestra utilizaciones totales de la CPU de 99.4% y 67.55% para el VAX8600 y VAX750 respectivamente. Estas estimaciones pueden compararse con las mediciones entregadas por el monitor que indican utilizaciones del 98% y 67.5%

Centro		Utilización [%]		Largo Cola
		MAP	MONITOR	
(CPU)	VAX8600	98.01	99.40	49.17
(CPU)	VAX750	67.55	67.50	2.08
	ARES	73.50	73.50	2.77
	HERA	11.90	11.90	0.14
	EOLO	4.40	4.40	0.05
	EROS	3.90	3.90	0.04
	APOLO	4.60	4.60	0.05
	BACO	3.50	3.50	0.04
	SCRATCH	16.40	16.40	0.20

TABLA 4.4 - Resumen Resultados (MAP) por Centro de Servicio (modelo 1, tipo de usuario)

Las utilizaciones de CPU muestran que la CPU del VAX8600 está cercana a la saturación, para el período de observación.

Se puede apreciar que la utilización de los discos, exceptuando el ARES (disco del sistema) con un 73% de utilización, es bastante baja con respecto a la utilización de la CPU. Definitivamente, el "cuello de botella" del sistema VAX8600/VAX750 es la CPU del VAX8600.

Cualquier redistribución de archivos en los discos, con el fin de uniformar la carga en el subsistema de I/O deberá tender a disminuir la carga del disco ARES, aunque su efecto será poco notorio debido a la gran contención producida en la CPU del VAX8600.

En sección 5 se analizarán los resultados de cambiar esta CPU por otra más rápida y trasladar parte de la carga de este disco a otro con poca carga.

Con respecto al largo de las colas en los diferentes centros de servicio se aprecia, como era de esperarse, que la cola más larga se produce en la CPU del VAX8600. Por otro lado, en la tabla 4.5 del anexo, columna "residencia CPU", se muestra el tiempo promedio de residencia (en segundos) para cada tipo de transacción en las CPU's. El tiempo de residencia en un centro de servicio incluye tanto el tiempo de servicio como el de espera en la cola. Los tiempos de respuesta, según puede verse en la tabla 4.5 (anexo) son extremadamente altos como corresponde a un equipo saturado.

Se puede apreciar en la tabla 4.5 (columna "Tiempo resp.") que el mayor tiempo de respuesta es, en el caso del VAX8600 para las transacciones FIS e IEE y en el VAX750 para SCC e

IIC (el tiempo de respuesta incluye tanto el tiempo requerido en la CPU como en los otros centros de servicio y los tiempos de contención).

4.6.2 Modelo transaccional por tipo de proceso

En las tablas 4.6 y 4.7 (anexo) se puede ver el resumen de resultados de la evaluación del modelo, a través de MAP, por centro de servicio y por clase de proceso.

	Centro	Utilización [%] MAP	MONITOR	Largo Cola
(CPU)	VAX8600	98.00	99.40	48.94
(CPU)	VAX750	67.55	67.50	2.08
	ARES	73.50	73.50	2.77
	HERA	11.90	11.90	0.14
	EOLO	4.40	4.40	0.05
	EROS	3.90	3.90	0.04
	APOLO	4.60	4.60	0.05
	BACO	3.50	3.50	0.04
	SCRATCH	16.40	16.40	0.20

Tabla 4.6 - Resumen Resultados (MAP) por Centro de Servicio (modelo 2, tipo de proceso)

Se puede ver, como era de esperar que los resultados obtenidos son muy similares al modelo anterior, encontrando que el "cuello de botella" del sistema es la CPU del VAX8600 y que el disco más utilizado es el ARES, tal y como debía ocurrir. Al igual que en 4.6.1 se puede observar que los tiempos de respuesta evidencian un equipo saturado, el cual no es capaz de soportar mayor carga.

Dejando de lado las clases "OTROS" procesadas por ambas CPU's, donde están incluidos especialmente los programas desarrollados por los usuarios, se puede apreciar que el mayor tiempo de respuesta es, en el caso del VAX8600, para el paquete estadística MINITAB, COMPILACIONES y EDICIONES. En el caso del VAX750 este mayor tiempo se produce para el uso del comando HELP (lo que no es de extrañar, debido a las características propias de los tipos de usuario de cada computador, EDICIONES y MINITAB (en ese orden).

4.6.3 Modelo mixto

En la tabla 4.8 se aprecia que debido a la introducción de la parte interactiva, los porcentajes de utilización de las CPU's VAX8600 y VAX750 subieron a 99.51% y 69.35% lo que es bastante cercano a la realidad de 99.4 % y 67.5%. Todo lo dicho sobre los discos en el punto 4.6.1 es también válido aquí.

Se puede apreciar que tal como se esperaba el largo de cola en la CPU del VAX8600 es bastante mayor que en la del VAX750.

Es interesante señalar como, debido a la casi saturación de la CPU en el VAX8600, el tiempo de respuesta interactivo (tabla 4.9 del anexo) es de algo más de 1 minuto, lo que ha sido constatado en las horas peak.

	Centro	Utilización [%] MAP	MONITOR	Largo Cola
(CPU)	VAX8600	99.51	99.40	60.44
(CPU)	VAX750	69.35	67.50	2.20
	ARES	75.04	73.50	2.96
	HERA	12.20	11.90	0.14
	EOLO	4.46	4.40	0.05
	EROS	3.96	3.90	0.04
	APOLO	4.67	4.60	0.05
	BACO	3.55	3.50	0.04
	SCRATCH	16.67	16.40	0.20
	TERMINALS			17.79

Tabla 4.8 - Resumen Resultados (MAP) por Centro de Servicio

5. ANALISIS DE PROYECCIONES

En esta sección se presentan resúmenes de los informes de desempeño entregados por el software MAP para algunas modificaciones al modelo mixto.

Se decidió trabajar con el modelo mixto ya que contiene una parte interactiva importante y permite plantear soluciones a la saturación existente analizada en la sección anterior.

5.1 Cambio de CPU del VAX8600 a una un 50% más rápida

En esta sección se observa el desempeño que tendría el sistema si se cambia la CPU actual del equipo VAX8600, que se encuentra saturada, por una un 50% más rápida. Todos los demás componentes del sistema permanecen sin variación, al igual que la carga de trabajo del sistema.

Las tablas 5.1 y 5.2 muestran un resumen del informe entregado por MAP con esta modificación: ahora la CPU del VAX-8600 es de un modelo definido con 6 MIPS en vez de 4 MIPS. Debido a ésto, las demandas de servicio en la CPU, para cada clase de transacción, se reducen en un cuarto.

Se observa que la utilización de las CPU's disminuyen de un 99.51% a un 92.36% en el caso del VAX8600 y de un 69.35% a un 68.59% en el caso del VAX750, sin embargo, la utilización del disco ARES aumenta peligrosamente de un 75.04% a un 88.58%. La pequeña disminución en la utilización del VAX8600 se explica por la existencia de la clase interactiva, ya que al existir mayor capacidad de proceso, el modelo responde aumentando el caudal de esta clase, lo que se puede apreciar comparando la TABLA 5.2 con la 4.9 donde aumenta el caudal de clases interactivas del VAX8600 de 0.1914 a 0.4939 transacciones/segundo. La disminución en la utilización del VAX750 podría explicarse por variaciones en la utilización del subsistema de I/O (recordemos que existe un cluster de discos entre los computadores VAX8600 y VAX750).

Centro		Utilización [%] Proyectada	Actual	Largo Cola Proyectada	Actual
(CPU)	VAX8600	92.36	99.51	9.60	60.44
(CPU)	VAX750	68.59	69.35	2.12	2.20
	ARES	88.58	75.04	7.22	2.96
	HERA	13.65	12.20	0.16	0.14
	EOLO	5.76	4.46	0.06	0.05
	EROS	5.06	3.96	0.05	0.04
	APOLO	5.90	4.67	0.06	0.05
	BACO	4.53	3.55	0.05	0.04
	SCRATCH	20.88	16.67	0.26	0.20
	TERMINALS			25.92	17.79

Tabla 5.1 - Resumen Resultados (MAP) por Centro de Servicio CPU de 6.0 MIPS en VAX8600

Clase	Tiempo resp. [seg]	Caudal [trans/seg]	Util. CPU [%]	Residencia CPU [seg]	Cola CPU
TAC_COMPIL	39.16	0.0082	2.85	36.90	0.30
TAC_LINK	19.23	0.0064	0.63	10.53	0.07
TAC_MINIT	59.86	0.0009	0.49	57.17	0.05
TAC_OTROS	258.64	0.0311	0.46	156.64	4.87
ANT_COMPIL	22.70	0.0014	0.59	13.64	0.02
ANT_LINK	29.33	0.0011	0.34	9.92	0.01
ANT_MINIT	50.98	0.0014	0.73	16.68	0.02
ANT_SUBMIT	3.19	0.0058	0.31	1.68	0.01
ANT_OTROS	231.16	0.0127	45.71	112.19	1.43
TAC_INTERA	12.84	0.4939	42.49	8.73	4.31
ANT_INTERA	15.32	0.1063	20.92	5.97	0.64

Tabla 5.2 - Resumen Resultados (MAP) por Clase de Proceso CPU de 6.0 MIPS en VAX8600

Con respecto a los tiempos de respuesta, se observa una reducción del orden de un 84% en el VAX8600 mientras que hay un aumento de un 26% en el VAX750. La reducción en el VAX8600 era completamente predecible ya que se cuenta con mayor capacidad de proceso. Del aumento en el VAX750, es responsable la mayor cantidad de transacciones interactivas del VAX8600 y las variaciones en la utilización del subsistema de I/O.

Es interesante notar que los tiempos de respuesta en terminales, sin ser buenos han llegado a valores aceptables.

5.2 Aumento de 10% en carga interactiva de terminales

El resumen de resultados de MAP del sistema con una CPU de 6.0 MIPS en el VAX8600 y un 10% de aumento en la carga interactiva se muestra en las tablas 5.3 y 5.4.

Centro		Utilización [%] Proyectada	Actual	Largo Cola Proyectada	Actual
(CPU)	VAX8600	93.95	99.51	11.78	60.44
(CPU)	VAX750	70.35	69.35	2.30	2.20
	ARES	90.41	75.04	8.68	2.96
	HERA	13.98	12.20	0.16	0.14
	EOLO	5.85	4.46	0.06	0.05
	EROS	5.14	3.96	0.05	0.04
	APOLO	6.00	4.67	0.06	0.05
	BACO	4.60	3.55	0.05	0.04
	SCRATCH	21.24	16.67	0.27	0.20
	TERMINALS			27.44	17.79

Tabla 5.3 - Resumen Resultados (MAP) por Centro de Servicio CPU de 6.0 MIPS en VAX8600 y aumento de 10% en carga interactiva de terminales

Clase	Tiempo resp. [seg]	Caudal [trans/seg]	Util. CPU [%]	Residencia CPU [seg]	Cola CPU
TAC_COMPIL	47.11	0.0082	2.85	44.48	0.36
TAC_LINK	22.79	0.0064	0.63	12.69	0.08
TAC_MINIT	72.04	0.0009	0.49	68.91	0.06
TAC_OTROS	307.19	0.0311	45.90	188.83	5.87
ANT_COMPIL	24.99	0.0014	0.59	14.42	0.02
ANT_LINK	33.15	0.0011	0.34	10.48	0.01
ANT_MINIT	57.67	0.0014	0.73	17.62	0.02
ANT_SUBMIT	3.53	0.0058	0.31	1.77	0.01
ANT_OTROS	257.45	0.0127	45.71	118.54	1.51
TAC_INTERA	15.32	0.5124	44.08	10.55	5.41
ANT_INTERA	17.23	0.1152	22.67	6.31	0.73

Tabla 5.4 - Resumen Resultados (MAP) por Clase de Proceso CPU de 6.0 MIPS en VAX8600 y aumento de 10% en carga interactiva de terminales

Se aprecia que la utilización de la CPU del VAX8600 ha aumentado al 93.95% y la del VAX750 al 70.35%, lo cual deja la CPU del VAX750 más cargada que en la actualidad.

La utilización del disco ARES llega al 90.41% debido al mayor overhead por manejo de terminales lo que amerita un traslado de parte de la carga de este disco a otro de menor utilización.

Los tiempos de respuesta han aumentado en un 17% en el VAX8600 y en un 11% en el VAX750.

5.3 Traslado de 50% de carga de ARES a APOLO

Debido a la saturación manifiesta del disco ARES, se ha procedido a trasladar un 50% de su carga al disco APOLO, mientras que la carga transaccional se mantiene con un 10% de aumento y la CPU del VAX8600 a 6.0 MIPS.

Con este traslado (ver tabla 5.5) la CPU del VAX8600 se encuentra utilizada al 95.6% y la del VAX750 al 74.29% (81.1% y 74.26% para CPU de 8.0 MIPS en VAX8600 respectivamente). En tanto, el disco ARES ha disminuido su utilización al 46.78% y APOLO la ha aumentado de 6.1% a 52.97% (50.14% y de 6.39% a 56.92% respectivamente para CPU de 8.0 MIPS en VAX8600). Las variaciones del tiempo de respuesta pueden verse en la tabla 5.6.

Centro		Utilización [%] Proyectada	Actual	Largo Cola Proyectada	Actual
(CPU)	VAX8600	95.60	99.51	15.11	60.44
(CPU)	VAX750	74.29	69.35	2.78	2.20
	ARES	46.78	75.04	0.88	2.96
	HERA	14.50	12.20	0.17	0.14
	EOLO	6.03	4.46	0.06	0.05
	EROS	5.30	3.96	0.06	0.04
	APOLO	52.97	4.67	0.01	0.05
	BACO	4.75	3.55	0.05	0.04
	SCRATCH	21.92	16.67	0.28	0.20
	TERMINALS			25.87	17.79

Tabla 5.5 - Resumen Resultados (MAP) por Centro de Servicio CPU de 6.0 MIPS en VAX8600, CARGA TRANSACCIONAL CON AUMENTO DEL 10% Y TRASLADO DE 50% DE LA CARGA DEL DISCO ARES A APOLO

Clase	Tiempo resp. [seg]	Caudal [trans/seg]	Util. CPU [%]	Residencia CPU [seg]	Cola CPU
TAC_COMPIL	56.83	0.0090	3.13	56.09	0.50
TAC_LINK	18.84	0.0070	0.69	16.00	0.11
TAC_MINIT	87.78	0.0010	0.54	86.91	0.09
TAC_OTROS	271.38	0.0342	50.49	238.13	8.14
ANT_COMPIL	19.12	0.0015	0.65	16.52	0.02
ANT_LINK	17.58	0.0012	0.37	12.01	0.01
ANT_MINIT	30.04	0.0015	0.80	20.19	0.03
ANT_SUBMIT	2.46	0.0063	0.34	2.03	0.01
ANT_OTROS	170.01	0.0140	50.28	135.86	1.90
TAC_INTERA	14.61	0.4735	40.74	13.25	6.27
ANT_INTERA	9.92	0.1010	21.65	7.20	0.80

Tabla 5.6 - Resumen Resultados (MAP) por Clase de Proceso CPU de 6.0 MIPS en VAX8600, CARGA TRANSACCIONAL CON AUMENTO DEL 10% Y TRASLADO DE 50% DE LA CARGA DEL DISCO ARES A APOLO

6. CONCLUSIONES

En este trabajo se ha presentado una aplicación de la tecnología de modelos de redes de espera a un sistema computacional en operación en un ámbito universitario. El sistema en estudio fué monitoreado durante períodos de alta utilización de recursos. Diversos modelos analíticos fueron implementados para responder preguntas planteadas tanto ante cambios en el hardware del equipo como en la carga de trabajo.

Para lograr lo anterior, se estableció una metodología que permitirá a futuro automatizar la recopilación y el análisis de la información de tal forma de hacerla más expedita. El objetivo entonces, de futuros trabajos es el de elaborar herramientas de software que permitan automatizar el proceso de interpretación de estadísticas de monitoreo y su transformación a parámetros de un modelo de redes de espera.

Con este trabajo, se logró conocer el patrón de demanda de las aplicaciones más típicas como editores y compiladores, de tal forma que es posible alterar los patrones de demanda de estas aplicaciones por separado, para los estudios de proyecciones.

A través del trabajo de proyección de los modelos se ha visto que se hace necesario cambiar el procesador del VAX8600 por uno más rápido (o una red de procesadores) para permitir un mejor desempeño del sistema, como asimismo una redistribución de la carga ejercida sobre el disco ARES en alguno de los discos poco utilizados como el APOLO.

Referencias

[Casa86] Casas I.
"Prophet: A Layered Analytical Model for Performance Prediction of Database Systems", Computer Systems Research Institute, University of Toronto, Tech. Rep. CSRI-180, Toronto, Canada, 1986.

[Casa87] Casas I., Aurtenechea F.
"Evaluación y Predicción del Desempeño de Sistemas Computacionales", Actas de la XIII Conferencia Latinoamericana de Informática, Bogotá, Colombia, Nov. 1987

[Grah78] Graham G.S. (ed.)
"Queueing Network Models of Computer System Performance", ACM Comp. Surveys Special Issue #10.3, Sept. 1978.

[Lazo84] Lazowska E.D., Zahorjan J., Graham G.S., Sevcik K.C.
"Quantitative System Performance - Computer System Analysis Using Queueing Network Models", Prentice Hall Inc., New Jersey, 1984.

[Oval88] Ovalle F.
"Modelación y Evaluación del Desempeño de un Sistema Computacional Centralizado", Memoria de Título, Escuela de Ingeniería, Pontificia Universidad Católica de Chile, Abril 1988.

[Saue82] Sauer C.H., et al.
"The Research Queueing Package - Version 2: CMS Users Guide", IBM Research Report RA 139 (#41127), T.J. Watson Research Center, Yorktown Heights, New York, Dic. 1982.

Período de muestreo: 6600 (seg)

Tipo de usuario	Transacciones	T. CPU
SCC	206	767.77
IIC	4	14.6
ICM	13	67.23
ICS	31	24.53
IEE	46	1721.60
AGR	25	26.65
EAE	21	26.95
EXT	206	131.36
ICT	10	1.21
MED	70	132.03
FIS	45	2673.72
ICH	198	149.23
BIO	48	182.96
MAT	3	0.61
ISD	2	18.47
ICE	22	27.74
OTROS	9	2.18
START	56	499.29

Disco	Utilización
ARES	0.440
HERA	0.049
EOLO	0.041
EROS	0.035
APOLO	0.039
BACO	0.031
SCRATCH	0.134

Número total de Direct I/O de cada clase en cada disco:

Tipo Usuario	ARES	HERA	EOLO	EROS	APOLO	BACO	SCRATCH	TOTAL
SCC	40942.46				4094.25	6823.74	16376.98	68237.44
IIC	415.66	519.57					103.91	1039.15
ICM	6650.55	3325.28					6650.55	16626.38
ICS	138.55	173.19					34.64	346.38
IEE	2216.85	1108.43					2216.85	5542.13
AGR	415.66		519.57				103.91	1039.15
EAE	277.11		346.38				69.28	692.77
EXT	1662.64			2078.30	207.83		207.83	4156.60
ICT	138.55	173.19					34.64	346.38
MED	1870.47						207.83	2078.30
FIS	6927.70		4329.79	4529.79			1731.91	17319.14
ICH	2493.96	1246.98					2493.96	6234.89
BIO	1524.09		1905.11				381.02	3810.21
MAT								0.00
ISD	138.55	173.19					34.64	346.38
ICE	2493.96	1246.98					2493.96	6234.89
OTROS								0.00
START	12469.78							12469.78
TOTAL	80776.51	7966.81	7100.85	6408.09	4302.08	6823.74	33141.92	146520.00

Tabla 3.1 - Mediciones por tipo usuario computador VAX-8600

Período de muestreo: 6600 [seg]				
Tipo de proceso	Transacciones	T. CPU	Disco	Utilización
EDICIONES	152	566.63	ARES	0.440
COMPILACIONES	54	281.96	HERA	0.049
LINK	42	62.57	EOLO	0.041
DIRECTORY	254	768.38	EROS	0.035
LOGINOUT	239	80.16	APOLO	0.039
MINITAB	6	48.54	BACO	0.031
DELETE	150	34.32	SCRATCH	0.134
MAIL	27	20.76		
TYPE	100	20.76		
ADEMAS	264	39.59		
OTROS	205	4544.34		

Tabla 3.2 - Mediciones por tipo de proceso computador YAX-8600

Número total de Direct I/O de cada clase en cada disco:

Tipo Usuario	ARES	HERA	EOLO	EROS	APOLO	BACO	SCRATCH	TOTAL
EDICION.	4715.48	502.47	386.52	386.52	270.56	386.52	1932.58	8580.63
COMPILA.	373.83	39.83	30.64	30.64	21.45	30.64	153.21	680.25
LINK	1115.69	118.88	91.45	91.45	64.01	91.45	457.25	2030.19
DIRECTORY	7791.33	830.22	638.63	638.63	447.04	638.63	3193.17	14177.67
LOGINOUT	1127.29	120.12	92.40	92.40	64.68	92.40	462.01	2051.30
MINITAB	49.31	5.25	4.04	4.04	2.83	4.04	20.21	89.73
DELETE	739.68	78.82	60.63	60.63	42.44	60.63	303.15	1345.98
MAIL	305.30	32.53	25.02	25.02	17.52	25.02	125.12	555.55
TYPE	92.10	9.81	7.55	7.55	5.28	7.55	37.75	167.59
ADEMAS	413.35	44.05	33.88	33.88	23.72	33.88	169.41	752.17
OTROS	63796.62	6798.00	5229.23	5229.23	3660.46	5229.23	26146.15	116088.94
TOTAL	80520.00	8580.00	6600.00	6600.00	4620.00	6600.00	33000.00	146520.00

Tabla 3.3 - Número de I/O computador YAX-8600

Clase	Demandas CPU (seg/tran)		Demandas disco (seg/tran)						
	VAX8600	VAX750	ARES	HERA	EOLO	EROS	APOLO	BACO	SCRATCH
ANTSCC		70.038	41.242			1.015	1.777	1.015	1.904
ANTMATALU		3.674	1.484	1.004					0.291
ANTIIC		49.310	7.421	5.022					1.456
ANTICEALU		4.784	1.590	1.076					0.312
ANTIICALU		10.528	1.237	0.837					0.243
ANTIEEALU		3.790	1.531	1.036					0.300
ANTICSALU		3.333	1.601	1.083					0.314
ANTPSIALU		2.742	2.573		0.374				0.275
ANTSTART		8.064	5.060						
TACSCC	3.727		7.146				1.189	0.993	2.121
TACIIC	3.650		3.736	5.273					0.693
TACICM	5.172		18.392	10.383					13.652
TACICS	0.791		0.161	0.227					0.030
TACIEE	37.426		1.733	0.978					1.286
TACAGR	1.066		0.598		0.792				0.111
TACEAE	1.283		0.474		0.629				0.088
TACEXT	0.638		0.290			0.364	0.060		0.027
TACICT	0.121		0.498	0.703					0.092
TACMED	1.886		0.961						0.079
TACFIS	59.416		5.535		3.667	3.468			1.027
TACICH	0.754		0.453	0.256					0.336
TACBIO	3.812		1.142		1.513				0.212
TACMAT	0.203								
TACISD	9.235		2.491	3.515					0.462
TACICE	1.261		4.075	2.301					3.025
TACOTROS	0.242								
TACSTART	8.916		8.005						

Tabla 4.1 - Parámetros modelo por tipos de usuario (modelo 1)

Clase	Demandas CPU (seg/tran)		Demandas disco (seg/tran)						
	VAX8600	VAX750	ARES	HERA	EOLO	EROS	APOLO	BACO	SCRATCH
ANTEDICIO.		9.978	2.059	0.489	0.021	0.028	0.049	0.028	0.209
ANTCOMP.		4.367	1.042	0.247	0.011	0.014	0.025	0.014	0.106
ANTLINK		3.174	2.234	0.530	0.023	0.030	0.053	0.030	0.227
ANTDIRECT.		0.663	0.630	0.150	0.006	0.009	0.015	0.009	0.064
ANTLOGIN.		1.567	2.692	0.639	0.027	0.036	0.064	0.036	0.274
ANTMINIT.		5.337	3.948	0.937	0.040	0.054	0.094	0.054	0.401
ANTDELETE		0.441	0.276	0.065	0.003	0.004	0.007	0.004	0.028
ANTSUBMIT		0.536	0.174	0.041	0.002	0.002	0.004	0.002	0.018
ANTTYPE		0.645	0.197	0.047	0.002	0.003	0.005	0.003	0.020
ANTHELP		10.285	3.144	0.746	0.032	0.043	0.075	0.043	0.320
ANTADEMA.		0.400	0.270	0.064	0.003	0.004	0.006	0.004	0.027
ANTOTROS		35.906	13.689	3.248	0.139	0.186	0.325	0.186	1.392
TACEDICIO.	3.726		1.119	0.125	0.104	0.139	0.099	0.079	0.341
TACCOMP.	5.221		0.250	0.028	0.023	0.029	0.022	0.018	0.076
TACLINK	1.490		0.958	0.107	0.089	0.076	0.085	0.067	0.292
TACDIRECT.	3.025		1.106	0.123	0.103	0.088	0.098	0.078	0.337
TACLOGIN.	0.335		0.170	0.019	0.016	0.014	0.015	0.012	0.052
TACMINIT.	8.090		0.296	0.033	0.028	0.024	0.026	0.021	0.090
TACDELETE	0.229		0.178	0.020	0.017	0.014	0.016	0.013	0.054
TACMAIL	0.769		0.408	0.045	0.038	0.032	0.036	0.029	0.124
TACTYPE	0.208		0.033	0.004	0.003	0.003	0.003	0.002	0.010
TACADEMAS	0.150		0.056	0.006	0.005	0.004	0.005	0.004	0.017
TACOTROS	22.168		11.224	1.250	1.046	0.893	0.995	0.791	3.418

Tabla 4.2 - Parámetros modelo por tipos de proceso (modelo 2)

Clase	Demandas CPU (seg/tran)		Demandas disco (seg/tran)						
	VAX8600	VAX750	ARES	HERA	EOLO	EROS	APOLO	BACO	SCRATCH
ANTCOMP.		4.367	1.042	0.247	0.011	0.014	0.025	0.014	0.106
ANTLINK		3.174	2.234	0.530	0.023	0.030	0.053	0.030	0.227
ANTMINIT.		5.337	3.948	0.937	0.040	0.054	0.094	0.054	0.401
ANTSUBMIT		0.536	0.174	0.041	0.002	0.002	0.004	0.002	0.018
ANTOTROS		35.906	13.689	3.248	0.139	0.186	0.325	0.186	1.392
ANTINTERA.		1.968	1.094	0.260	0.011	0.015	0.026	0.015	0.111
TACCOMP.	5.221		0.250	0.028	0.023	0.020	0.022	0.018	0.076
TACLINK	1.490		0.958	0.107	0.089	0.076	0.085	0.067	0.292
TACMINIT.	8.090		0.296	0.033	0.028	0.024	0.026	0.021	0.090
TACOTROS	22.168		11.224	1.250	1.046	0.893	0.995	0.791	3.418
TACINTERA.	1.291		0.462	0.051	0.043	0.037	0.041	0.033	0.141

Tabla 4.3 - Parámetros modelo mixto (modelo 3)

Clase	Tiempo resp. [seg]	Caudal [trans/seg]	Util. CPU [%]	Residencia CPU [seg]	Cola CPU
ANTSCC	377.68	0.0039	27.59	215.82	0.85
MATALU	18.41	0.0265	9.74	11.32	0.30
ANTIIC	187.39	0.0003	1.48	151.95	0.05
ICEALU	22.34	0.0021	1.01	14.74	0.03
IICALU	38.35	0.0136	14.36	32.44	0.44
IEEALU	18.99	0.0191	7.24	11.68	0.22
ICSALU	17.91	0.0077	2.58	10.27	0.08
PSIALU	18.88	0.0080	2.20	8.45	0.07
ANTSTAR	43.94	0.0017	1.35	24.85	0.04
TACSCC	218.75	0.0312	11.63	186.98	5.84
TACIIC	204.03	0.0006	0.22	183.12	0.11
TACICM	356.96	0.0020	1.02	259.45	0.51
TACICS	40.60	0.0047	0.37	39.70	0.19
TACIEE	1886.80	0.0070	26.09	1877.61	13.09
TACAGR	56.70	0.0038	0.40	53.48	0.20
TACEAE	66.94	0.0032	0.41	64.38	0.20
TACEXT	33.56	0.0312	1.99	31.99	1.00
TACICT	8.86	0.0015	0.02	6.07	0.01
TACMED	98.34	0.0106	2.00	94.62	1.00
TACFIS	3010.37	0.0068	40.52	2980.81	20.33
TACICH	40.21	0.0300	2.26	37.81	1.13
TACBIO	197.37	0.0073	2.77	191.23	1.39
TACMAT	10.20	0.0004	0.01	10.20	0.00
TACISD	477.25	0.0003	0.28	463.31	0.14
TACICE	84.87	0.0033	0.42	63.26	0.21
TACOTRO	12.15	0.0014	0.03	12.15	0.02
TACSTAR	477.50	0.0085	7.56	447.30	3.79

Tabla 4.5 - Resumen Resultados (MAP) por Clase de Usuario

Clase	Tiempo resp. [seg]	Caudal [trans/seg]	Util. CPU [%]	Residencia CPU [seg]	Cola CPU
TAC_EDICION	191.34	0.0230	8.59	186.18	4.29
TAC_COMPIL	261.93	0.0082	4.27	260.78	2.13
TAC_LINK	78.92	0.0064	0.95	74.40	0.47
TAC_DIREC	156.19	0.0385	11.64	151.09	5.81
TAC_LOGIN	17.54	0.0362	1.21	16.75	0.61
TAC_MINIT	405.41	0.0009	0.74	404.04	0.37
TAC_DELETE	12.25	0.0227	0.52	11.43	0.26
TAC_MAIL	40.28	0.0041	0.31	38.40	0.16
TAC_TYPE	10.52	0.0152	0.31	10.37	0.16
TAC_ADEMA	7.05	0.0400	0.60	7.49	0.30
TAC_OTROS	1168.87	0.0311	68.85	1107.12	34.39
ANT_EDICION	39.46	0.0124	12.40	30.75	0.38
ANT_COMPIL	17.86	0.0014	0.60	13.46	0.02
ANT_LINK	19.23	0.0011	0.34	9.78	0.01
ANT_DIREC	4.71	0.0286	1.90	2.04	0.06
ANT_LOGIN	16.21	0.0211	3.30	4.83	0.10
ANT_MINIT	33.13	0.0014	0.73	16.45	0.02
ANT_DELETE	2.53	0.0114	0.50	1.36	0.02
ANT_SUBMIT	2.39	0.0058	0.31	1.65	0.01
ANT_TYPE	2.82	0.0161	1.04	1.99	0.03
ANT_HELP	44.99	0.0003	0.31	31.70	0.01
ANT_ADEMA	2.37	0.0112	0.45	1.23	0.01
ANT_OTROS	168.54	0.0127	45.70	110.66	1.41

Tabla 4.7 - Resumen Resultados (MAP) por Clase de Proceso

Clase	Tiempo resp. [seg]	Caudal [trans/seg]	Util. CPU [%]	Residencia CPU [seg]	Cola CPU
TAC_COMPIL	322.01	0.0082	4.27	320.81	2.62
TAC_LINK	92.13	0.0064	0.95	91.33	0.58
TAC_MINIT	498.48	0.0009	0.74	497.06	0.45
TAC_OTROS	1415.86	0.0311	68.85	1361.99	42.30
ANT_COMPIL	19.56	0.0014	0.60	15.95	0.02
ANT_LINK	20.01	0.0011	0.34	10.14	0.01
ANT_MINIT	34.19	0.0014	0.73	17.05	0.02
ANT_SUBMIT	2.48	0.0058	0.31	1.71	0.01
ANT_OTROS	175.20	0.0127	45.70	114.74	1.46
TAC_INTERA	77.83	0.1914	24.71	75.63	14.48
ANT_INTERA	10.86	0.1102	21.68	6.10	0.67

Tabla 4.9 - Resumen Resultados (MAP) por Clase de Proceso

Capítulo 9

Inteligencia Artificial

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

APREND - UM SISTEMA PARA AQUISIÇÃO AUTOMÁTICA DE CONHECIMENTO A PARTIR DE EXEMPLOS.

FERNANDO A. de Carvalho GOMES
Universidade Estadual do Ceará - UECe

GIUSEPPE MONGIOVI
Universidade Federal da Paraíba - UFPb

HÉLIO de Menezes SILVA
Universidade Federal da Paraíba - UFPb

Grupo de Inteligência Artificial
Departamento de Sistemas e Computação
Universidade Federal da Paraíba - UFPb
Caixa Postal 10071
Campina Grande - Pb
58100
Brasil

Telefone : (083) 321.7222 r.510
Telex : 0832211

Sumário - Este trabalho descreve o APREND, um sistema para aquisição automática de conhecimento a partir de exemplos. O método utilizado associa dois processos, o ID3 e o D-S. O ID3 é usado para gerar uma árvore de decisão que é a representação de todo o conhecimento adquirido e que pode ser transformado para a forma de regras. O D-S é um algoritmo baseado na teoria das evidências de Dempster-Shafer e é usado para a determinação dos fatores de certeza das regras geradas pelo ID3.

1.0 INTRODUÇÃO

A aquisição de conhecimento é o processo de transferência de um certo domínio do conhecimento de um perito nesse domínio para o sistema, envolvendo também a transformação deste conhecimento numa forma que possa ser entendida.

Embora a tecnologia de desenvolvimento de sistemas especialistas tenha experimentado avanços consideráveis, a aquisição de conhecimento continua sendo o gargalo ("bottleneck") na construção de um sistema especialista (Feigenbaum 1981 e Buchanan 1985), isto deve-se principalmente à problemas de comunicação entre o engenheiro do conhecimento e o perito no domínio. Estes problemas de comunicação advêm da linguagem específica que cada domínio do conhecimento possui e, de acordo com (Waterman 1986): "o perito combina as partes do conhecimento básico de uma forma tão rápida, que torna-se difícil para ele descrever o processo de raciocínio inerente a uma conclusão".

Existem dois métodos de aquisição de conhecimento: cognitivo e automatizado (Boy 1988). Os métodos cognitivos são fundamentados essencialmente no engenheiro do conhecimento, que funciona como uma espécie de "filtro" entre a informação fornecida pelo perito e a representação do conhecimento na máquina. Os métodos automatizados são aqueles que são processados com o auxílio do computador e, portanto, sem a influência direta do engenheiro do conhecimento. Entre os métodos de aquisição automática de conhecimento destacam-se: indução por meio de árvores de decisão, entrevista automática com o perito e métodos gráficos de apresentação das estruturas do conhecimento.

A conscientização de que o cruciante problema da

aquisição de conhecimento deva ter soluções "rápidas e impessoais", veio estimular a investigação de métodos automatizados que aumentem a eficiência e confiabilidade do processo.

O processo de indução através de árvores de decisão, é um método simples, que não necessita de nenhum conhecimento prévio do domínio. Este método funciona a partir de exemplos, que são descritos por atributos e diagnósticos ou classes e registrados no cotidiano do perito no domínio. A indução se utiliza de teorias probabilísticas e como resultado fornece o conhecimento representado na forma de árvore de decisão, de onde pode-se obter regras do tipo SE...ENTÃO. À estas regras associa-se um fator de certeza ou fator de atenuação (FA), que representa a característica de incerteza do conhecimento sendo adquirido. Por exemplo, um FA de 0,8 indica que existe uma forte evidência (0,8 em 1,0) de que a regra se verifica.

O objetivo deste trabalho é descrever um sistema de aquisição automática de conhecimento, baseado na indução em árvores de decisão (Quinlan 1986a) e em um processo estatístico para o cálculo do FA das regras geradas. Esta associação fornece regras que constituem um conhecimento confiável a ser aplicado em sistemas especialistas baseados em regras.

Na seção 2.0 será feita uma descrição do método de aquisição automática de conhecimento utilizado pelo sistema. Na seção 3.0 será mostrado uma visão geral do sistema APREND e um exemplo completo da obtenção de regras para sistemas especialistas.

2.0 O METODO UTILIZADO

O método proposto para obtenção de regras a partir de exemplos é uma combinação do algoritmo ID3 (Quinlan 1986a) e um processo para o cálculo dos FA's das regras geradas, baseado na teoria das evidências de Dempster-Shafer (Shafer 1976).

O ID3 gera regras de modo eficiente, através de indução em árvores de decisão, porém não calcula os FA's das regras geradas. A determinação informal dos FA's é um processo subjetivo e normalmente leva a resultados discrepantes. Portanto se faz necessário um processo formal de geração dos FA's. O processo de Dempster-Shafer (D-S) poderia ser usado para gerar as regras com os seus respectivos FA's, porém com uma complexidade explosiva e com alguns incovenientes na fase de formação das regras. Então a solução encontrada foi combinar as vantagens dos dois processos: o ID3 para gerar as regras e o D-S apenas para calcular os FA's das regras já formadas, porém com uma complexidade muito reduzida.

2.1. Indução em Árvores de Decisão:

O algoritmo ID3 é um membro da família TIDT ("Top-Down Induction of Decision Trees") e é originário do CLS, um algoritmo desenvolvido por Hunt nos anos cinquenta e apresentado em (Hunt 1966).

O ID3 é o algoritmo que mais tem apresentado resultados práticos, por exemplo, o sistema de diagnóstico de falhas de placas de circuitos impressos da ITT Europa, o sistema para separação gas-óleo da British Petroleum e o controle de processos para usinas nucleares da Westinghouse (Carter 1987). A grande vantagem do ID3 com relação a outros métodos é a escolha do

universo de exemplos que serão trabalhados. Nesse aspecto, o ID3 utiliza apenas alguns exemplos para chegar a uma árvore que classifique todos os exemplos observados . Este pequeno conjunto de exemplos chama-se janela.

A tarefa do ID3 é formar uma árvore de decisão que classifique corretamente todos os exemplos observados.

No primeiro passo, o algoritmo irá fornecer uma árvore de decisão que, não só classifica corretamente todos os elementos do conjunto janela, como também grande parte dos exemplos observados. Em seguida irá pesquisar na totalidade dos exemplos, todos aqueles que não são classificados pela árvore fornecida. Então, uma seleção destes elementos será realizada de forma a incluí-los no conjunto janela. A partir da nova janela, o algoritmo formará uma nova árvore de decisão e o processo de classificação se repetirá até que todos os exemplos observados sejam classificados. Ao final do processo, cada caminho da árvore de decisão resultante,representa uma regra. O conjunto destas regras constitui o conhecimento correspondente aos exemplos observados.

2.2 O **Processo D-S:**

A característica principal do D-S é a de considerar a totalidade dos exemplos observados não como um universo absoluto, mas como parte de um universo maior. Por exemplo, cem casos anotados por um médico especialista, não podem ser considerados como o universo de todos os casos possíveis. Já outros métodos estatísticos consideram os cem exemplos como um universo absoluto. Esta característica do D-S produz FA's com valores que estão mais próximos da realidade.

O processo D-S apresenta uma desvantagem. Para se calcular o FA de um atributo com relação a uma classe, necessita-se percorrer a totalidade dos exemplos anotados. Entretanto esta ineficiência praticamente desaparece quando as regras já estão formadas. É o caso de o processo ser usado apenas para o cálculo dos FA's.

3.0 DESCRIÇÃO DO SISTEMA APREND

3.1 Visão Geral

O sistema é dividido em três fases distintas: a edição da tabela de exemplos, a geração de regras e a determinação dos FA's.

A fig. 3.1 mostra um gráfico de estrutura correspondente ao problema a ser resolvido, apresentando dois níveis de detalhamento.

O módulo "EDITAR EXEMPLOS" edita os elementos da tabela de exemplos, que dizem respeito ao domínio do conhecimento que se quer implantar no sistema especialista. O objetivo principal deste módulo é obter exemplos que se passaram no cotidiano do perito no domínio. O módulo "GERAR REGRAS" corresponde ao algoritmo ID3. Ele chama o módulo "INCL. JANELA" nas fases de formação da janela e inclusão de elementos não mapeados.

O módulo "DETERMIN. FA", corresponde ao D-S.

Tanto a nível conceitual, como a nível de implementação, foram usados alguns recursos importantes para diminuir a complexidade dos algoritmos envolvidos.

3.2 Adquirindo Conhecimento

O conhecimento é adquirido a partir de um conjunto de

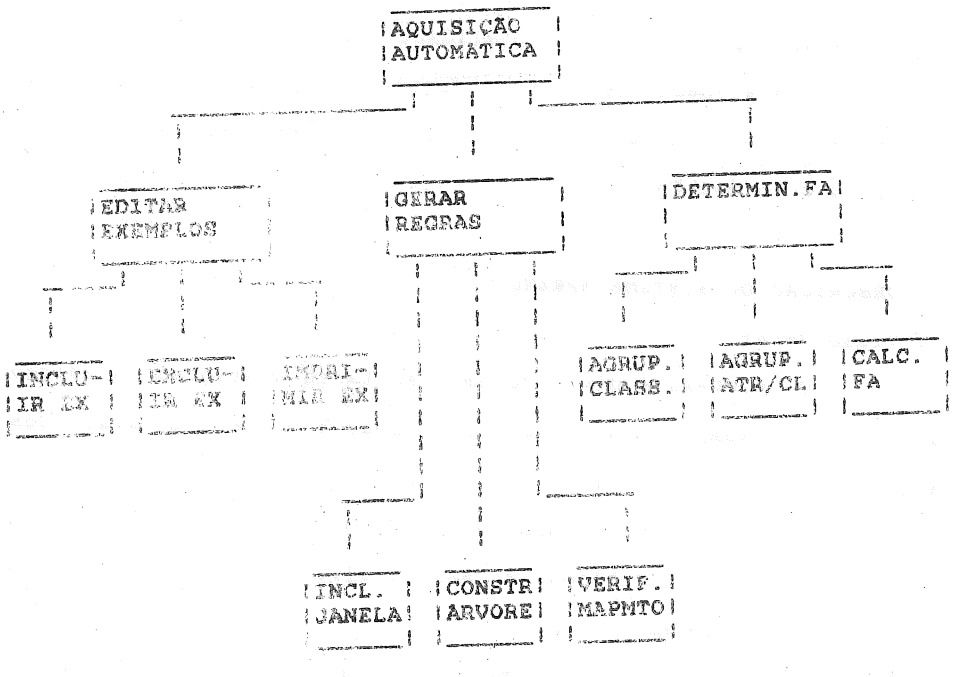


Figura 3.1 Visão Geral do Sistema

casos. Cada caso, também chamado de objeto ou exemplo, é descrito como uma coleção de atributos. Cada atributo indica alguma característica importante de um objeto e será limitado a um conjunto de valores discretos, mutuamente exclusivos. A classe, é o diagnóstico do caso.

A seguir descreve-se informalmente o sistema APREND, através da apresentação de um exemplo ilustrativo.

O exemplo ilustrativo consiste na observação por vários anos, das condições meteorológicas das manhãs de sábado, durante um longo período de observações.

Consideram-se os seguintes atributos, adaptados de (Quinlan 1986a):

ATRIBUTO VALORES

Tempo.....[ensolarado, nublado, chuvoso]

Temperatura.....[frio, agradável, quente]

Umidade.....[alta, normal]

Vento.....[forte, fraco]

Quando juntos, pode-se caracterizar muito bem as condições meteorológicas de uma manhã de sábado, por exemplo:

Tempo : ensolarado;

Temperatura : agradável;

Umidade : normal;

Vento : fraco.

Para formar um objeto, resta definir a classe deste objeto, que possui um conjunto de valores observados de acordo com o domínio do conhecimento que se quer adquirir. Esta observação poderia ser, no caso dos atributos já definidos, a decisão de ficar em casa ou sair para praticar um esporte. Assim sendo, a classe terá os valores [Casa, Futebol, Basquete].

Objetos que parecem ser atípicos são particularmente interessantes, pois indicam uma situação que, mesmo difícil de acontecer, vem influenciar na construção da árvore de decisão, como veremos adiante.

Utilização do ID3

Para a escolha dos elementos do conjunto janela, leva-se em conta a proporção da ocorrência de cada classe em todos os exemplos observados.

Suponha que foram observados um total de 1.000 casos. Para o conjunto janela,inicialmente, foram utilizados 11

elementos, conforme a fig. 3.2.

No.	Tempo	Temperatura	Umidade	Vento	Classe
1	ensolarado	agradável	normal	forte	basquete
2	nublado	frio	alta	fraco	casa
3	nublado	agradável	alta	fraco	basquete
4	chuvoso	frio	alta	forte	casa
5	nublado	quente	normal	fraco	futebol
6	ensolarado	quente	normal	fraco	futebol
7	nublado	agradável	alta	forte	basquete
8	chuvoso	frio	alta	fraco	casa
9	nublado	agradável	normal	forte	futebol
10	ensolarado	quente	alta	fraco	casa
11	chuvoso	quente	alta	forte	basquete

Figura 3.2 Conjunto Janela

A raiz da árvore de decisão corresponde ao atributo de menor entropia, isto é, o que apresenta menor incerteza com relação às classes. O conceito de entropia é originário da teoria da informação e sua fundamentação pode ser encontrada em (Aczél 1975).

A entropia de um atributo A é a média ponderada das entropias dos valores de A. Matematicamente é dada por:

$$H(C|A) = \sum_{i=1}^{N} p(v_i) \cdot H(C|v_i) \qquad (3.1)$$

onde:

N - número total de valores do atributo A;

$p(v_i)$ - é a probabilidade de o valor v_i aparecer no conjunto janela;

$H(C|v_i)$ - é a entropia do valor v_i de A, representada pela expressão:

$$H(C|v_i) = -\sum_{j=1}^{M} p(c_j|v_i) \cdot \log p(c_j|v_i), \qquad (3.2)$$

onde:

M - número total de classes.

$p(c_j|v_i)$ - é a probabilidade de se ter um exemplo com o valor v_i concluindo c_j.

No conjunto janela da fig. 3.2, calcula-se a entropia de cada atributo. Por exemplo, a entropia do atributo Tempo, é obtida como segue:

$$H(\text{Classe=casa}|\text{Tempo=nublado}) = -\frac{1}{5}\log\frac{1}{5} = 0,46,$$

$$H(\text{Classe=basquete}|\text{Tempo=nublado}) = -\frac{2}{5}\log\frac{2}{5} = 0,53,$$

$$H(\text{Classe=futebol}|\text{Tempo=nublado}) = -\frac{2}{5}\log\frac{2}{5} = 0,53,$$

então, pela expressão (3.2) tem-se:

H(C|Tempo=nublado) = 0,46 + 0,53 + 0,53 = 1,52.

Calculando-se a entropia para os outros valores do atributo Tempo, encontra-se:

H(C|Tempo=ensolarado) = 1,59 e H(C|Tempo=chuvoso) = 0,92.

Desta forma, pela expressão (3.1), a entropia do atributo Tempo será:

$$H(C|\text{Tempo}) = \frac{5}{11}1,52 + \frac{3}{11}1,59 + \frac{3}{11}0,92 = 1,37.$$

Calculando-se a entropia dos outros atributos, tem-se:

H(C|Temperatura) = 0,85,

H(C|Umidade) = 0,92,

H(C|Vento) = 1,41.

O atributo de menor entropia é Temperatura, que será, conforme a fig. 3.3, a raiz da árvore de decisão.

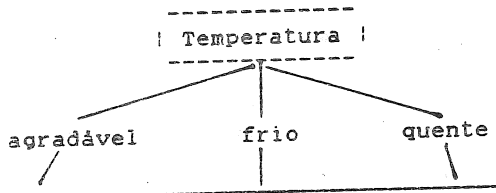


Figura 3.3 Raiz da árvore de decisão

Para se obter as sub-árvores correspondentes aos valores do atributo raiz, procede-se de forma análoga à anterior. Nesse caso, usa-se uma sub-janela que é formada apenas pelos elementos da janela anterior que contêm o valor do atributo raiz correspondente à sub-árvore específica.

Por exemplo, para o valor **agradável** da fig. 3.3, a sub-janela é a da fig. 3.4.

No.	Tempo	Temperatura	Umidade	Vento	Classe
1	ensolarado	agradável	normal	forte	basquete
3	nublado	agradável	alta	fraco	basquete
7	nublado	agradável	alta	forte	basquete
9	nublado	agradável	normal	forte	futebol

Figura 3.4 Sub-janela para Tempo=agradável

A fig. 3.5 mostra a árvore de decisão que classifica corretamente cada elemento do conjunto janela da fig. 3.1. Supondo que essa árvore é a árvore de decisão final, cada caminho dessa árvore (raiz-folha) representa uma regra. Por exemplo, o caminho mais a esquerda, na árvore da fig. 3.5, fornece a regra:

SE Temperatura é agradável &

Umidade è alta

ENTÃO Jogue Basquete

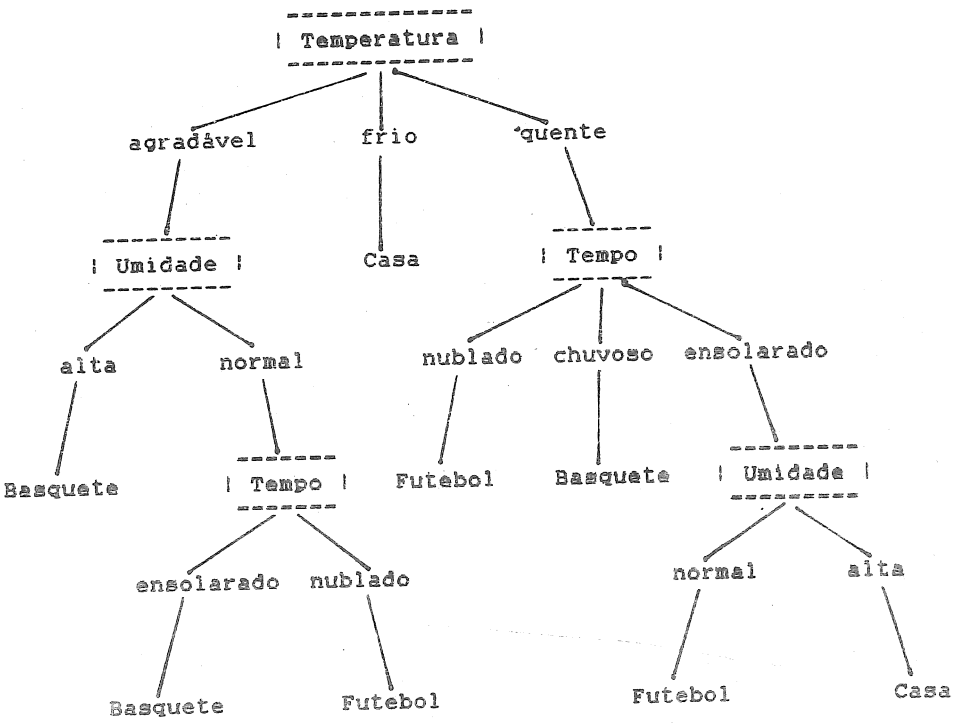


Figura 3.5 Árvore de Decisão

Se dois objetos no conjunto janela possuem os mesmos valores para todos os atributos, mas com valores diferentes para a classe, deve-se rotular o nó relativo a estes dois exemplos como Nulo, isto constitui o que se chama de "ruído" e um processo de eliminação do ruído é proposto em (Quinlan 1986b).

Utilização do D-S

De posse das regras geradas pelo ID3, calcula-se o FA de cada regra, através do D-S.

Inicialmente deve-se conhecer alguns conceitos da Teoria de Dempster-Shafer. A evidência que uma condição da regra (atributo = valor), contribui para que a conclusão (classe)

ocorra, chama-se de "medida de crença do valor do atributo com relação a uma classe", ou C(v,c), onde "v" é o valor do atributo e "c" é o valor da classe. A "medida de crença do valor do atributo com relação a uma classe" permite calcular o FA deste atributo com relação a uma regra que conclua a mesma classe. Além da "medida de crença", necessita-se calcular também a "medida de crença-contrária", ou C(v,-c), que é a evidência de que a condição da regra contribui para que não ocorra a conclusão. Estes conceitos podem ser vistos em (Shafer 1976).

A expressão que fornece o fator de certeza (FA) de um atributo com relação a uma classe, é:

$$FA(v,c) = \frac{C(v,c).(1 - C(v,-c))}{1 - C(v,c).C(v,-c)}$$

Para se encontrar a medida de crença de um atributo com relação a uma classe utiliza-se a expressão:

$$C(v,c) = pl \ . \ (1 - P)$$

onde:

pl - é a probabilidade de ocorrer o valor v e a classe c, ao mesmo tempo, no conjunto dos exemplos;

p - é a área sob a curva de Gauss que dá a probabilidade de que a proporção de ocorrer a classe c, não mudará, ao separarmos todos os exemplos que possuem o valor do atributo escolhido.

Para se encontrar o FA final de uma regra, basta combinar, dois a dois, os fatores de certeza dos atributos que formam esta regra, usando a seguinte expressão:

$$FA = FA1 + FA2 - FA1 \ . \ FA2.$$

Para exemplificar a utilização do método, calcular-se-á, com dados simulados do exemplo anterior, o FA para a regra:

SE Temperatura é agradável &

Umidade é alta

ENTÃO Jogue Basquete.

A "medida de crença" em que Temperatura = agradável influencia Classe = basquete (utiliza-se agora a totalidade dos exemplos, ou seja: 1.000), é:

C(Temperatura=agradável,basquete) = 0,78 . (1 - 0,38) = 0,48,

a seguir calcula-se a "medida de crença-contrária" em que Temperatura ≠ agradável influencia Classe=basquete, acha-se:

C(Temperatura=agradável, -basquete) = 0,22 . (1 - 0,38) = 0,14,

com estes valores, tem-se:

$$FA(\text{Temperatura=agradável, basquete}) = \frac{0,48 \;.\; (1 - 0,14)}{1 - 0,48 \;.\; 0,14} = 0,44.$$

Agora encontra-se o FA do outro atributo da regra (Umidade = alta), que é:

FA(Umidade=alta,basquete) = 0,53.

Combina-se os dois FA's. Então o FA, resultará:

FA = 0,44 + 0,53 - 0,44 . 0,53 = 0,74,

portanto a regra será:

SE Temperatura é agradável &

Umidade é alta

ENTÃO Jogue Basquete

FA = 0,74

esta é uma regra que é aplicável em sistemas especialistas.

4.0 Conclusões

Neste trabalho foi abordado o problema geral da aquisição de conhecimento. Em particular, salientando-se a importância dos métodos de indução para gerar árvores de decisão a partir de exemplos. Foi mostrado que, entre os métodos de indução, o algoritmo ID3 é o que tem apresentado melhores resultados práticos. Para lidar com a natureza incerta do conhecimento, as regras geradas pelo ID3 foram complementadas com fatores de certeza, calculados por meio de um algoritmo baseado na teoria das evidências de Dempster-Shafer.

As idéias apresentadas foram utilizadas no projeto do sistema APREND. Para melhor ilustrar essas idéias, o sistema foi descrito informalmente através do acompanhamento de uma exemplo ilustrativo.

Referências

Aczél, J & Daróczy, Z, On Measures of Information and Their Characterizations, Academic Press, New York, 1975.

Boy, G., Faller, B. & Sallantin, J., Acquisition et Ratification de Connaissances; Crim, Montpellier, 1988.

Buchanan, B.G. ,Lenat, E. & Waterman, D.A., Building Expert Systems, Addison Wesley, Reading, Mass., 1985.

Carter, C. & Catlett J., Assessing Credit Card Applications Using Machine Learning, in IEEE Expert (Fall), 1987.

Feigenbaum, E.A., Expert Systems in 1980s, in State of the Art Report on Machine Intelligence, A. Bond, Ed., Maidenhead: Pergamon-Infotech, 1981.

Hunt, E.B., Marin, J. & Stone,P., Experiments in Induction,

Academic Press, New York, 1966.

Michalski, R.S. & Chilauski,R.L., Knowledge Acquisition by encoding expert rules versus computer induction from examples: a case study involving soybean pathology, in Fuzzy Reazoning and its Applications, Gaines, B.R., London, 1981.

Quinlan, J.R., Induction of Decision Trees,In Machine Learning, Kluver Academic Publishers, Boston,MA, 1986a.

Quinlan, J.R., The Efect of Noise on Concept Learning, in Machine Learning: An Artificial Intelligence Approach, V.2, Michalski, R et al. (Eds.), Morgan Kaufmann, Palo Alto, CA,1986b.

Shafer, G., A Mathematical Theory of Evidence, Princeton University Press, Princeton, NJ, 1976.

Waterman, D.A., A Guide to Expert Systems, Addison Wesley, 1986.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa.
Buenos Aires, Setiembre 1988.

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
PROGRAMA DE ENGENHARIA DE PRODUCAO
PROGRAMA DE ENGENHARIA EM SISTEMAS E COMPUTACAO
NUCLEO INTERDISCIPLINAR DE ENGENHARIA DE CONHECIMENTO
COPPE/UFRJ/1988

UMA ABORDAGEM GERAL SOBRE CRENCA E CONHECIMENTO NO CONTEXTO SOCIAL

Juan C. Mangione

Caixa Postal 68511-CEP: 21945-Cidade Universitária-Rio de Janeiro
TEl: (021) - 280-8632 Ramal 419 / (021)- 280-9322 Ramal 128-
TELEX: 33817 UFCOBR BRASIL

I.- Resumo: O escopo e suporte do trabalho:

Utilizamos neste trabalho uma linguagem formal denominada lógica modal para M agentes. Esta lógica apresenta características que não agridem as colocações de filósofos, psicólogos e teólogos. O nosso trabalho dirige-se a formulações lógico-matemáticas e não às expressões trascendentais do ser humano, ainda que alguns cientistas tenham conseguido esquematizar matematicamente esses fenômenos. A nossa preocupação, é só refletir sobre alguns pontos em relação à Inteligência Artificial e à codificação de fenômenos da área da Engenharia do Conhecimento.

II.- Desenvolvimento do trabalho:

A Teoria do Conhecimento tem se convertido nos últimos anos num dos pilares da Inteligencia Artificial, a Economia e a Linguística, entre outras ciências. Seguindo uma ordem oportuna para o entendimento deste trabalho, tem-se organizado os temas de maneira a lograr uma introdução gradativa e coerênte dentro da visão global da comunicação. Porem, começaremos com o tema da

"crença", para posteriormente, estudar alguns tópicos sobre "consciência" e, finalmente, nos ocuparemos do "conhecimento", propriamente dito.

IV.- Lógica de Levesque KL,Sel para Crença explícita e implícita e Consciência

Quando alguem se refer ao problema de conhecer um determinado fato real ou formal, surge a pergunta se o sujeito tem consciência do fato p. Halpern [H,86] questiona e se pergunta quais as interpretações de fatos reais ou formais introduzidos dentro de culturas onde estes fatos são atípicos. Obviamente, estariam faltando conceitos fundamentais em relação ao entorno do problema. Não há dúvidas de que tudo sujeito pertence a uma tribu em particular sendo que nos referimos com este termo a um grupo social, localizado geográfica e culturalmente, evoluido e contemporáneo e de modo algum, ténta-se conferir conotações pejorativas. As normas do convívio mudam de uma tribu para outra e cada grupo social se encarrega de ministrar os conceitos necessários para asegurar a difusão dos fatos básicos na vida social e o reconhecimento e assimilação dos símbolos que os representam.

Do mesmo modo que se tenta conscientizar sobre certos fatos, reais ou formais, existem em formas variadas, as vezes, contrastantes, outras complementares, uma ampla faixa de crenças num grupo social determinado. As crenças, em particular, têm a potencialidade suficiente para fazer agir um determinado sujeito, ainda de maneira absolutamente arbitrária.

IV.1.- Crença Explícita e Implícita:

Vamos utilizar os símbolos utilizados por Halpern [H,86] para designar as crenças, segundo os quais B, representa a crença

explícita e L, a implícita. As crenças implícitas de um agente ou sujeito, incluim todas as consequências lógicas das suas crenças explícitas.

Um modelo para estas crenças seria:

$$M = (S, \mathfrak{b}, T, F) \qquad \text{(IV.1)}$$

onde:

- S, é um conjunto de situações primitivas;
- $\mathfrak{b}$, é um subconjunto de S (aquelas que poderiam ser certas segundo no que se acredite).
- T e T(p), são todas as situações que suportam a certeza de p.
- F e F(p), são todas as situações que suportam a falsedade de p.
- T e F, são partes da função primitiva.

Resulta importante aqui, esclarecer que se apresentam situações de incoerência pois, existem elementos de T(p) os que simultaneamente, pertencem a F(p).

$$T(p) \cap F(p) \qquad \text{(IV.2)}$$

É por isso que uma situação de incoerência suporta tanto a certeza como a falsedade de uma proposição primitiva.

Definição 1:

Denominasse "situação completa" àquela que suporta à certeza ou falsedade de toda proposição primitiva e que não é incoerênte. O seja, uma situação é completa quando é exatamente membro de T(p) ou de F(p).

Definição 2:

Uma situação s é compatível com uma outra situação s' quando ambas concordam para tudo dominio que s seja definida. O seja, se

e do mesmo modo pode se provar

para F(p).

Agora, definimos as relações de suporte:

$$\models T \quad e \quad \models F,$$

relações de suporte entre situações e fórmulas. Podemos escrever, então que:

$$M, s_i \models_T \phi, \qquad \text{<IV.3>}$$

quando a situação s no modelo M, suporta a validade ou certeza da função e do mesmo modo para a falsedade. Também podemos escrever que:

$$M, s_i \models_T p,$$

com p proposição primitiva se:

$$\text{se } T(p)$$

Escrevemos:

$$M, s_i \models_T \neg\phi \text{ se } M, s_i \not\models_T \phi$$

$$M, s_i \models_F \neg\phi \text{ se } M, s_i \models_T \phi$$

$$M, s_i \models_T \phi_1 \wedge \phi_2 \text{ se}$$

$$M, s_i \models_T \phi_1 \text{ e } M, s_i \models_T \phi_2$$

$$M, s_i \models_T B\phi \text{ se } M, t_i \models_T \phi \ \forall\ (t \in b) \qquad \text{<IV.4>}$$

$$M, s_i \models_F B\phi \text{ se } M, s_i \not\models_T B\phi$$

$$M, s_i \models_T L\phi \text{ se } M, t_i \models_T \phi \ \forall\ (t \in b_x) \qquad \text{<IV.5>}$$

$$M, s_i \models_F L\phi \text{ se } M, s_i \not\models_T L\phi$$

Levesque [L,84] , define uma fórmula como válida, se

$$\phi$$

é verdadeira na situação s ∀ modelo do tipo:

$$M = (S, b, T, F)$$

e para toda situação completa

$$(se S)$$

Tal como Levesque [L,84] expõe, a crença explícita, implica na crença implícita, então:

$$(B\phi \dashrightarrow L\phi) \qquad \text{<IV.6>}$$

Seria muito simples comprovar que a crença implícita tem fechamento sob implicação e que se acredita implícitamente em todas as fórmuls válidas. Então, se:

$$\models \phi;\ \phi \text{ com } \phi$$

proposicional, então,

$$\models L\phi \qquad e,$$

(IV.7)

$$\models (L\phi \wedge L(\phi \rightarrow \Psi)) \rightarrow L\Psi$$

IV.2.- Consciência:

Vamos introduzir o conceito de "consciência" o que será denotado pela letra A, para manter a terminolgia original do inglês, procedente do vocábulo "awareness". Seja uma fórmula proposicional e sejam,

$$P_1, P_2, P_3, \ldots, P_k$$

as proposições primitivas que nela aparecem. Temos assim,

$$A(P_1) \wedge A(P_2) \wedge \ldots \wedge A(P_k) \rightarrow B(\phi)$$

(IV.8)

A expressão anterior diz que um agente acredita numa fórmula válida se é consciente de todas as proposições primitivas que aparecem nessa fórmula. Uma crença, com consciência ou não dos fatos envolvidos, estará sempre sujeita à uma relação que:

* pode se acreditar em alguma coisa falsa, em tanto
* que não se pode conhecer um fato falso.

Daqui em diante qualquer uma outra explicação seria redundante. Lembre-se que ainda tendo o conjunto de situações possíveis e as suas compatíveis ainda estamos dentro de um sistema baseado em crenças e não no conhecimento.

Em concordância com as fórmulas apresentadas até este ponto, podemos dizer que no nosso caso, é uma função (fórmula válida) e que só se pode acreditar nela se o agente fosse consciente de

todas as proposições primitivas que nela aparecem. Supondo que trabalhamos numa função "Devagar, escola", na que se incluim as primitivas p1, p2, p3, sendo respectivamente:

p_1: fator humano;

p_2: fator ambiente;

p_3: fator técnico.

Quantos integrantes da nossa tribu reconhecem (obedecem), por exemplo àquele sinal de Devagar, escola? Quantos deles, associam o sinal com seus conteúdos, humanos, ambientais e técnicos? Qual o peso relativo do sinal segundo cada tribu?

Em quanto consideramos inútil para os nossos propósitos às formas de proceder proprias das crenças, resgatamos a forma explícita pela sua associação à consciência. Podemos formalizar a relação crencas - consciência através dos seguintes formalismos: (IV.9)

$$B_i(pVq)=[(A_{ip} \wedge L_{ip})V(A_{iq} \wedge L_{iq})V(A_{ip} \wedge A_{iq} \wedge L_i(pVq))]$$

Também, podemos escrever:

$$B_iBJp=(A_{ip} \wedge L_i(A_{jp} \wedge L_{jp})), \quad (IV.10),$$

devendo se lembrar que:

$$A_{ip}=B_i(pV\neg p), \quad (IV.11),$$

em termos gerais para a nossa fórmula

$$B_i\phi=L_i\phi \wedge A_i\phi, \quad (IV.12),$$

do que surge que não é possível crença explícita si não estiver associada à consciência.

Das fórmulas (IV.9) - (IV.12), se deduiz que existe uma conjunção permanente entre L e A para implicar em B mas, ainda não é um ponto de apoio para os nossos propósitos. Por exemplo,

pense-se no caso da expressão (IV.10), ajudada pela (IV.11). É evidente que o fato de que o agente i acredita que o agente j acredita em p, equivale a dizer que:

1.- i, é sciente de p e mantem uma crença implícita (intuitiva) de que j é sciente de p além de ter uma crença implícita de p.

Ou substituindo (IV.11) em (IV.10):

2.- i, acretida expressamente em p ou -p e também tem uma intuição de que j acredita expressamente em p ou -p, além de ter uma certa intuição de p).

Estes modelos das crenças são os espaços comuns onde quase tudo mundo se encontra englobado no comportamento individual e social. O nosso sistema de crenças, é bem maior do que a nossa base de conhecimento pois, dominamos um pequeno quando não diminuto espaço cognitivo. Para o homem que exercita uma vida social dentro de uma "tribu", como é que cada integrante pode compatibilizar o seu sistema de crenças com os dos restantes integrantes da tribu? Existe alguma possibilidade de ante-por as proprias crenças ante as alheias? Como se garante que o comportamento social daria certo se se basear num sistema de crenças?

O que não apresenta pontos escuros ou de dúvida é o fato de que no nosso caso, as diferentes formas cognitivas num grupo social e suas implicações no entorno, dependem fortemente da base de conhecimento que cada um dos integrantes detem e do sistema de crença enraizado na comunidade. O maior ou menor peso de um sobre o outro, depende de uma disciplina cognitiva e de uma separação permanente entre "fatos propriamente ditos" (conhecimento) e aquelas "regras" de comportamento baseadas num sistema de crenças.

No nosso diario agir, supomos, achamos, acreditamos que os nossos interlocutores supõem, acham, acreditam coisas ou fenômenos iguais ou compatíveis com os nossos mas, poucas vezes,

estamos certos de que "conheçamos" e de que eles conheçam a realidade de tais fatos?

Infelizmente, na ordem social, a grande maioria orienta-se pelas crenças e muito pouco pelo conhecimento. Acontece ainda hoje, que existe toda uma tradição de crenças conformando um quase-folklore do comportamento social e que, irrevogavelmente, pertence a cada tribu o grupo. É parte da nossa comunicação diaria, tácita, subjacente e assumida como certa, pois, tudo aquele que conseguisse se entroçar num nivel de comunicação superior, é pelo fato de ter superado (assimilado) e aprofundado aquelas premisas mínimas que pertencem ao ambiente das crenças mas nem sempre se englobam na nossa base de conhecimentos.

V.- Conhecimento, conduta e comportamento social:

As nossas ações cotidianas estão relacionadas pela conjunção de crenças e suposições. É assim que, integrantes de um mesmo grupo social ou tribu, diferem nas suas crenças políticas, religiosas, morais e algumas outras de tipo mais casuístico. Não há nada de ruim nesse tipo de aspectos do comportamento humano se se toma em conta o fato que é certamente, proprio do livre arbítrio do homem e se constitui no seu privilégio. Tudo isto pode obrar tanto a favor ou em contra do grupo social entanto que não representa perigo na órdem individual sendo que tudo mundo tem direito para obrar segundo o que acredita como "conveniente". Se a vida social dependesse das crenças, deveriamos preocuparnos por achar "o comum denominador de todas elas" de maneira que fosse justa a forma de conducirnos socialmente. É aqui que aparece com toda a sua potência o conceito e valor do "conhecimento", em termos de comportamento social. Nenguem age segundo o que "não conhece" porem, a necessidade de nos basear em fatos para garantir que a comunicação social e os resultados

esperados disfrutem do suporte de fatos, concretos e conhecidos.

V.1.- Modelo clássico do conhecimento:

Vamos utilizar o modelo clássico para conhecimento e crença, denominado, "modelo dos mundos possíveis". Em termos gerais (e de maneira intuitiva), a ideia é que além do estado real das coisas, existe um número de estados possíveis delas. Para expressar isto de um modo diferente, pode-se falar que um agente conhece um fato

ϕ se ϕ

é verdadeira em todos os mundos que o agente pensa como possíveis. Por exemplo, a figura abaixo dá uma idéia de como funciona o modelo clássico em termos de dois agentes que têm mundos possíveis. Assim, para o agente A, pode ser certo que o Carnaval se celebra no verão ou no inverno e que para um outro agente B, no entanto, esse evento acontece induvidavelmente no verão. Sendo necessário formalizar estes conceitos, vamos utilizar a lógica modal para m agentes, fazendo uso de proposições primitivas p, q, r,... e incluiremos, conjunção, negação e os operadores modais:

$$K_1, K_2, \ldots, K_m.$$

Assim, se

ϕ e Ψ

são fórmulas, também são fórmulas:

$\neg\phi$, $\phi \rightarrow \Psi$ e $K_1\phi$,

sendo a última fórmula lida como, "o agente 1, conhece o fato" ϕ

V.2.- Estruturas de Kripke

Quando necessitamos um símbolo semántico para a nossa logica, utilizamos o símbolo $\models$. Além disso, Kripke define otros

elementos para conferir à logica aspectos semanticos. Una estrutura de Kripke tem a forma:

(S, Π, p1,...pm), onde:

S: conjunto de estados ou mundos possíveis;

Π: é uma assignação de valor verdadeiro à proposição primitiva para cada estado

$$(s \in S)$$

de maneira que:

$$\Pi(s,p) \in (\text{verdadeiro}, \text{falso})$$

para cada estado s e proposição primitiva p.

pi: relação de equivalência sobre o conjunto S, para i=1,...m.

pi, é uma relação de possibilidade do agente i. Intuitivamente

$$(s,t) \in Pi$$

quando o agente i, não pode distinguir o estado s do t, o seja, que se s é o verdadeiro estado do mundo, i, pode conssiderar t como possível. Como um simples exemplo da notação,

$M, s_j \models \phi$, ϕ satisfaz o estado s do modelo M;

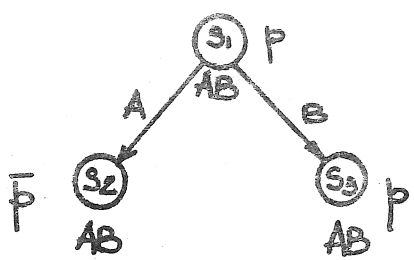
$M, s_j \models p$, p satisfaz o estado s do modelo M se $\Pi(p,s)$ = verdadeiro.

$M, s_j \models \neg\phi$ se $M, s_j \not\models \phi$

$M, s_j \models \phi \wedge \psi$ se $M, s_j \models \phi$ e $M, s_j \models \psi$

$M, s_j \models K_i\phi$ se $M, t_j \models \phi$ $\forall$ $t / (s,t) \in Pi$

FIGURA # 1



No exemplo da fig. 1, tomando p como única proposição primitiva, simbolizando, digamos, o Carnaval se celebra no verão. No estado,

S_1

p é verdadeiro, mas o agente A não sabe disso pois, considera possível tanto a certeza como a falsedade de p, o seja,

S_1 ou S_2,

são possíveis. Por um outro lado, A sabe que B sabe se p é ou não verdadeira e, finalmente, B sabe o estado certo de p mas, não sabe que A não conhece esse fato porque ele considera

S_3

possível e nesse estado, A conhece p. Formalizando:

$$M, s_1 \models p \wedge \neg K_{Ap} \wedge K_A(K_{Bp} \vee K_{B\neg p}) \wedge K_{Bp} \wedge \neg K_B \neg K_{Ap}$$

Deve se observar que ainda no estado S1 como no estado S3 a proposdão primitiva p, toma o mesmo valor verdadeiro e poderia se pensar que são iguais sendo eliminável um deles, mas isso não é certo porque um estado não é caracterizado completamente pelos valores verdadeiros que a primitiva assume mas também pela relação de possibilidade. Isto é, por exemplo em S1, A não conhece p mas se o conhece no estado S3.

A noção de conhecimento apresentada, pode ser completamente caracterizado por um sistema de axiomas de Hintika [Hi,62]:

1). **Todas as instâncias de tautologias proposicionais;**

2). $Ki\phi \wedge Ki(\phi \rightarrow \psi) \rightarrow Ki\psi$

3). $Ki\phi \rightarrow \phi$

4). $Ki\phi \rightarrow KiKi\phi$

5). $\neg Ki\phi \rightarrow Ki\neg Ki\phi$

6). $\dfrac{\phi, \phi \rightarrow \psi}{\psi}$ **modus ponens**

7). $\dfrac{\phi}{Ki\,\phi}$

1 e 6, são remanentes da lógica proposicional:

2, diz que, o conhecimento de um agente, tem fecho sob implicação;

3, diz que um agente só conhece cosas certas;

4 e 5 são axiomas de introspeção pois, sendo que se um agente pode observar a sua base de conhecimento, porem, saber o que ele sabe.

Neste ponto é necessário esclarecer que no trabalho de Halpern [H,86], consideram-se agentes conhecedores ideais o que implica em que: todos eles conhecem as fórmulas e as consequências lógicas desse conhecimento. Comprová-se-á que esse modelo não se adapta à realidade humana, pois implicaria numa "onisciência"lógica. Halpern [H,86] diz que segundo os problemas poderia se aplicar as formas já expostas, nem sempre sendo as adequadas.

Um outro modelo do conhecimento humano, baseia-se num tratamento "sintático" que permete que aquilo que um agente conhece, seja representável por um conjunto de fórmulas. Este conjunto de fórmulas, não deveria estar restrito a fechar sob consequência lógica ou conter todos os axiomas de um determinado esquema de axiomas. Este tratamento, não sofre o problema da onisciência lógica nas aparece como dificílimo para ser analisado. Se nos propomos representar o conhecimento por um conjunto arbitrário de fórmulas, não temos princípios para guiar uma análise baseada no conhecimento.

Recentemente, Levesque [L,84] tem contribuido para solucionar estes problemas através de uma aproximação semântica. O seu trabalho diferencia entre conhecimento implícito e explícito. Este último está formado pelos fatos que o agente registra a nível consciênte. Por uma outra via é exatamente o conhecimento implícito o que satisfaz os axiomas discutidos previamente e resulta interessante colocar aquí que um agente

pode não ter ideia do seu proprio conhecimento implícito. Levesque [L,84] conssidera um modelo de mundos possíveis, onde, num estado determinado, uma proposicção p, pode ser tanto, verdadera ou falsa, todas as duas ou nehuma, sendo assim, um estado inconsistente.

Segundo os conceitos apresentados por Levesque [L,84] , acontece que ainda falta definir alguns outros conceitos como conhecimento explícito e conhecimento comum.

Existe conhecimento explícito num grupo quando "todos sabem da existência de p". Denotamos por Ep (everybody knows p) e utilizamos uma estrutura de Kripke. Assim:

$$M, s_i' = Ep \text{ se } M, t_i' = p \ \forall\ t \in p1 \cup p2 \cup p3 \cup \ldots \cup pm$$

de maneira que:

$$M, s_i' = E_p \ <---> \ M, s_i' = K_{1p} \wedge K_{2p} \wedge \ldots \wedge K_{mp}$$

do que pode-se falar que a conjunção $\wedge$ exige que todos e cada um dos integrantes saibam da existência do fato p para que exista Ep.

Introduzimos agora o conceito de Conhecimento Comum. Existe Cp num grupo qualquer sobre um fato real ou formal p quando, não soamente todos os integrantes do grupo conhecem dito fato mas, quando além disso, todos sabem que todos sabem...que todos sabem... que todos conhecem p. Segundo as estruturas de Kripke já utilizadas, podemos escrever que se:

$$M, s_i' = Ep <---> K_{1p} \wedge K_{2p} \cdots \wedge K_{mp}$$

então, o Cp está caracterizado por:

$$M, s_i' = Cp ---> M, s_i' = Ep; \text{ com } K = 1, 2, 3, \ldots, m$$

O conhecimento comum é requisito indispensável para produzir qualquer forma de acordo e alguns autores afirmam que existem, básicamente, duas formas de Cp. Por exemplo, opinam que existe Cp pela participação num mesmo grupo e certamente justificam-lo com

o exemplo de que um sinal vermelho é equivalente para todos os motoristas com licença de conduzir. A segunda forma de adquirir Cp de um fato qualquer p, é pela co-presencia quando acontece dito fato.

Para fazer uma análise completa, é necessário caracterizar o estado do conhecimento do sujeito num determinado instante de tempo. Tomaremos só um fato proposicional p (interpretado no seu estado verdadeiro) e tomaremos também como verdadeiro que o agente pode raciocinar em forma proposicional perfeitamente, de maneira que possa conhecer todas as consequências lógicas de p (embora fique isto bem longe do que o agente realmente conhece). Agreguemos agora, um outro fato proposicional q. Por introspeção, o agente pode saber que não conhece p e por instrospeção posterior, pode também saber que não conhece q. Apesar do beneficio introspectivo, o fato de não conhecer q, não é consequência lógica de não conhecer p.

Em relação aos fatos tratados até este ponto, este é o caso de um agente com perfeitas possibilidades de computar, o seja, raciocinar e com um perfeito sentido introspectivo. Este agente sabe perfeitamente o que conhece e aquilo que não conhece; vive num mundo proposicional real e o seu mundo conceitual está referenciado ao mundo real e ao seu conhecimento. O conjunto de fórmulas da lógica Proposicional do Conhecimento com o que o agente raciocina, está definido por:

L1.- Todas as primitivas p,q,r,...,z, são fórmulas.

L2.- Se p e q são fórmulas, também serão -p,p ^q e p ---> q

L3.- Se p é fórmula, kip, também é uma fórmula.

L4.- As únicas fórmulas do conjunto são as requeridas por L1, L2 e L3.

Tem-se tratado de explicar que o agente sofre do problema da

"omnisciência lógica" e que isso não se adequa muito ao modelo de raciocinio humano. Isto, devesse ao fato que temos apresentado uma aproximação sintática que é menos elegante se fosse comparada com o enfoque semántico de Levesque [L,84] . Este último, dá lugar as crenças e ao conhecimento, sendo desse modo, mais completo e flexível.

V.3.- Hierarquia do Conhecimento

As diferença fundamental entre crença e conhecimento é que não se conhecen fatos que não são certos ou verdadeiros, entanto que pode se acreditar em fatos falsos. O seja, temos falado de conhecimento sendo que se se satisfaz algum estado

(s ∈ S)

necessáriamente, existe uma proposição p,q,...,z que satisfaz o estado t (por exemplo) de um outro estado do mesmo modelo M. As crenças tem uma característica pela qual o agente pode acreditar hoje num determinado fato, e amanhã, mudar totalmente a sua crença, coisa poco provável na estrutura do conhecimento, pois, só se conhece o que é. Vamos, então, dar uma hierarquização dos diferentes níveis do conhecimento e os seus valores relativos.

V.3.1.- Conhecimento Implícito

Se diz que um grupo tem conhecimento implícito de um fato p, $I_G p$ se e só se, existe um agente (hipotético ou virtual) que conheça exatamento todo o que cada integrante do grupo conhece. Digamos, um agente conhece q e um outro agente conhece q --> p, então, se diz que o grupo conhece p. Formalmente:

$$K_1 q \wedge K_2 (q \rightarrow p) \longrightarrow I_G p \qquad \text{<V.3.1.>}$$

Apresenta-se um caso de excepcional valor nesta forma de conhecimento e não pelo que providencia mas pelo que tem se enraizado no nosso modelo de comportamento. Em primeira instância, _ devese aceitar o fato de que o Direito, como

instituição de longa data e acerbo, está baseado numa filosofia de tipo comportamentalista. Esperam-se "y" respostas concordantes com os "x" estímulos dados e para aqueles que não se submetessem as leis vigentes, aplicar-se-ão as formas punitivas correspondentes até se assegurar que o seu comportamento garanta possibilidades de inserção social. Acontece que a lei, para o cidadão comum, apresenta um aspecto altamente repressivo pois tudo o que consta escrito é certamente proibitivo e é isso que justamente provoca a rejeição por parte de cidadaõ comum. Além disso, e como fator mais importante, surge a pergunta, onde está aquele agente que sabe tudo o que cada um de nos conhece? Supõe-se que sendo que a lei é publicada e alguns membros da comuidade se ocupam de conhecer o seus conteúdos, então esses agentes sabem a lei e suas consequências mas, o resto da comunidade, como normalmente acontece, não conhece a lei nem as suas consequências. Esta fica espléndidamente explicitado pela fórmula (V.3.1.). Ainda não seja obvio, não é possível esperar um comportamento elevado ou superior de parte de uma comunidade se ela não foi treinada o mais exatamente, educada para produzir esse tipo de condutas. Sendo que se educa para a Matemática, conhecem-se as respostas e falhas típicas de que não consegue incorporar esta disciplina. Mas, quando se trata da vida e comportamento social, então surge a necessidade de entender o dominio da letra da lei para descobrir o espíruto dela, como forma de regulamentação da armonia das normas da convivência e de co-participação voluntária, o que alguns autores denominam Sinergia [Mas,80]. Para isto acontecer, é indispensável, sempre em termos de continuar com o mesmo tipo de sociedade e contexto, facilitar o "treinamento" ao nível sistemático que a educação institucional detem no entorno social específico, pois não é

possível suportar o modelo de (V.3.1) como paradigma de conhecimento para o comportamento social.

O cujo dito agente que conhece o que todos conhecem, conclui então que todos devem-se comportar segundo a lei e se sabe que tal sujeito não é um sujeito real e a sua presência no grupo é ficticia. Só se trata de uma "figura" hipotética e externa ao grupo que se usa para apresentar esta forma de conhecimento mas, de qualquer maneira, as respostas esperadas da população, baseiam-se neste nível de conhecimento. Nada mais errado pois, são uma minoria absoluta os que conhecem (no mais profundo sentido da palavra) a letra, o espíritu da lei e as suas consequências. Todos os restantes membros da comunidade, só são levados por uma conduta inercial, pelas crenças suportadas individualmente e por um sistema de eleição social que, sem os benefícios de educação sistêmica, podem se produzir rápidamente agentes nocivos dentro da sociedade e transmitir em forma errada a premisas básicas das normas do convívio. Definitivamente, só se pode atingir um comportamento esperado quando se ensaiarem as mecânicas desses modelos e isto, certamente, não pode ser feito só no ambiente doméstico nem ocasionalmente em alguma escola isolada do verdadeiro contexto social da tribu. Sempre até aqui o termo tribu como identificador de grupo étnico ou social e não como primitivo. As respostas que podem oferecer benefícios na comunidade, vão se encontrar na educação sistematizada sobre os tópicos e as áreas mais urgentes.

V.3.2.- Conhecimento explícito para alguns membros do grupo

Se diz que existe conhecimento explícito para alguns integrantes do grupo G quando uma parte qualquer dele, conhecem expressamente o fato em questão. Formalmente:

$$S_{Gp} = \bigvee K_{ip} \quad (V.3.2)$$

Isto implica numa disjunção onde qualquer um que conhecesse expressamente o fato p, seria em relação à forma-de conhecimento anteriormente apresentada, o agente que induziria a pensar em uma forma de conhecimento implícito pois, são eles os que geram o efeito intuitivo nos que justamente, não se ocuparam de conhecer ou estudar o dito fato. Podemos expandir a expressão <V.3.2> da seguinte maneira:

$$S_{Gp}=K_{1p} \vee K_{2p} \vee \ldots \vee K_{np} \quad \text{<V.3.3>}$$

Tomando esta amostragem do grupo, ainda não transmite confianza em que seja o modelo apropriado para esperar um comportamento adequado na sociedade. Estão faltando um número não determinado e imprevisívelmente perigoso de agentes que conheçam expressamente o dito fato. Surge assim, a forma completa de conhecimento explícito.

V.3.3.- <u>Conhecimento explícito expandido a todos os integrantes do grupo</u>

Se diz que todos num grupo conhecem p se o só se todos os integrantes do grupo G sem exceição conhecem-lo. Denotamos por:

$$E_{Gp}$$

podendo formalizar de mesmo modo que já fizemos para o conhecimento explícito de alguns dos membros do grupo como:

$$E_{Gp}= \wedge K_{1p}$$

Expandindo esta fórmula, podemos escrever, em virtude das disjunções implicadas que

$$E_{Gp}=K_{1p} \wedge K_{2p} \wedge \ldots \wedge K_{np} \quad \text{<V.3.4>}$$

Também podemos escrever:

$$E^{K}_{Gp} \quad \text{<V.3.5>}$$

com k > 2 e que se interpreta como,

$$p \quad e \quad E^{K}$$

conhecido em G.

Podemos acrescentar dizendo que:

$$E^{1}_{Gp} = E_{Gp} \quad \langle V.3.6 \rangle$$

E também escrevemos que:

$$E^{k+1}_{Gp} = E_{Gp}\, E^{k}_{Gp} \quad \langle V.3.7 \rangle$$

para k >= que 1 e se diz que E é k-conhecida quando todos sabem que todos sabem... que todos sabem que p é verdadeira.

$$E^{*}_{Gp} = \bigwedge_{(i,j \in G) \text{ e } (i<j<k)} K_{i1}\, K_{i2} \ldots K_{ijp}$$

Esta é, certamente, a forma em que se pode esperar que se conheçam os fatos que regem a vida de uma comunidade e esse é o terreno onde a Educação tem as suas grandes possibilidades. Esta forma de conhecimento, longe de ser a melhor ou, a mais elaborada, é, simplesmente a mais fatível pois, dependendo sempre da natureza humana e dos diversos comportamentos dos grupos humanos em questão, as formas de conhecimento mais elaborados apresentam-se como quase impossíveis. Lamentavelmente, os sistemas de eleições sociais se confrontam com as diversas contribuições dos integrantes numa forma destrutiva em relação ao comportamento esperado e as contribuições construtivas ficam sem produzir maiores benefícios. Não se intenta desde esta comunicação favorecer a ótica de que pretende uma forma de comportamento comum. Trata-se, simplesmente de que, a partir de uma educação coerênte e reflexiva, os integrantes de uma certa comunidade, tribu o grupo social, aprimorem o sentido sinergético o de associação voluntária, em prol de melhoras sociais. A ideia, em forma alguma, se relaciona com sometimento ou imposições na ordem social. Encontramo-nos, pelo contrário, propiciando uma forma de educação dissuasiva, com inspiração no progresso da comunidade, com aspirações a produzir um sujeito crítico, e um cidadão respeituoso da vida, como o bem supremo.

V.3.4.- Conhecimento comum

V.3.4.1.- Conhecimento comum por descobrimento de fatos:

Tenta-se aqui, mudar o estado do conhecimento implícito para quaisquer forma de conhecimento explícito. Exemplo disso seria o estado de equilibrio de uma balança onde, a partir do conhecimento intuitivo, precisa-se de um algoritmo de deteção que determine o ponto de equilibrio.

V.3.4.2.- Conhecimento comum por co-presencia com o fato p:

Surge da interação de todos os integrantes de um grupo quando analisam e determinam que todos sabem do fato p, além de comprovar que todos sabem, que todos sabem,..., o fato p.

V.3.4.3.- Conhecimento comum por membros de uma comunidade:

Segundo Halpern [H,86], esta forma de conhecimento existe pelo simples fato de pertencer a uma comunidade determinada, até aquela formada pelos integrantes de uma comunidade de motoristas com licença. Diz Halpern [H,86] que isso é garantia que todos sabem,..., que todos sabem do fato p e de que p --) q. Desde este ponto de vista Halpern [H,86] aparece como correto só que não tem contemplado que ainda disfrutando do status de, por exemplo, eleitor, nem eles conhecem as leis e o espíru da constituição.

Simplesmente, todos os cidadãos eleitores do país não têm conhecimento comum da constituição brasileira nem todos os motoristas com licença, têm conhecimento das normas e sinais do trânsito, entre outros vários exemplos válidos.

Existe conhecimento comum de um fato p num grupo G quando, p é verdadeiro e todos sabem que todos sabem que todos sabem,...,que todos sabem p. Formalmente:

$$C_{Gp}=p \wedge E_{Gp} \wedge E^2_{Gp} \wedge \ldots . E^{n}_{Gp}$$

que implica em todas as fórmulas:

$$k_{11}, k_{12}, \ldots, k_{1mp},$$

sendo ij os membros de G para quaisquer número finito n e isso é

equivalente à conjunção infinita de todas essas fórmulas. Dentro do marco referencial formado por as varias formas de conhecimento desenvolvidas até aquí, resulta evidente a hierarquia que existe no conhecimento.

$$C_{G} \rightarrow E^{K+1}_{G}, \ldots, \rightarrow E_{G} \rightarrow S_{G} \rightarrow I_{G} \rightarrow p$$

VI.- Análise conclusiva:

Desde o ponto de vista estrito da Teoria do Conhecimento, quando se incorpora o fator tempo, o Conhecimento Comum é inviável pois, duas pessoas percebem em tempos diferentes e oferecem respostas até infinitesimalmente diversas. Isso implica numa impossibilidad de distribuir e confirmar fatos em forma "comum" pois existe uma infinita gama de agentes e cada um, com uma recepção e percepção diferente. Paralelamente, o tempo de transmissão da informação, faz impossível no terreno do real, o Conhecimento Comum. Evidentes perdas de tempo encontram-se nestes processos. Certamente, esta forma de conhecimento, implica numa certeza de que todos e cada um dos agentes conhecem um fato p e de que todos sabem que todos sabem... que todos sabem tal fato p. Daí se comprende que não temos mecanismos automáticos para garantir tal fenômeno sem dilações. Adquiri-se Cp apois de "investido" um tempo de transmissão-recepção.

No nosso escopo as demandas por melhoras sociais, implicam num desejo de atingir o nivel de Conhecimento Explícito por quanto, os vagos conhecimentos que a população detem, baseiam-se no suporte do Conhecimento Implícito. Um ponto viável para atingir seria as primeiras formas de conhecimento explícito por quanto estes conceitos, alentam para atingir o ponto chave onde todos e cada um dos agentes conheça os fatos. Daí em diante, pode se esperar uma conduta mais evoluida, uma toma de consciência do funcionamento social e individual e uma melhora substancial nos

benefícios intelectuais propriamente ditos.----------

BIBLIOGRAFIA:

[H,86] Halpern, J; MOses, Y; Knowledge and common knowledge in a distributed enviroment,' Relatório Técnico, Stanford University, 1986.

Halpern, J; Reasoning about knowledge: an overview. Relatório técnico, IBM, Almaden, California, 1986.

Halpern, J; Moses, Y; A Guide to the Modal Logics of Knowledge and belief: Preliminary draft. Relatório técnico. Stanford University, 1986.

Halpern, J; MOses, Y; Towards a theory of knowledge and ignorance: preliminary draft. Relatório técnico. Stanford University, 1986.

Halpern, J; Belief, Awareness and limited reasoning: preliminary report. Relatório técnico. IBM, Almaden., California, 1986

[Hi,62] Hintika 62]; Knowledge and belief, Cornell University, 1962.

[L.84] Levesque, H; A logic of implicit and explicit belief. Proceedings of the National conference on Artificial Intelligence, 1984.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

Computing Certainty Factors in Expert Systems Using Dihypergraphs Flows

Julián Aráoz D.
Departamento de Procesos y Sistemas
Universidad Simón Bolívar

Edgardo Broner
Departamento de Matemáticas y Ciencias de la Computación
Universidad Simón Bolívar

Alberto Torres F.
Departamento de Matemáticas y Ciencias de la Computación
Universidad Simón Bolívar

Abstract: We use a Directed Hypergraph model that allows the reduction of derivation in Horn Systems to the calculation of a Linear Programming problem feasible solution associated to a Unit Flow searching. We explore the application of this tool in the calculation of Certainty Factors. Since several Expert Systems Knowledge Bases are expressed, or can be expressed, with Horn clauses associated with Certainty Factors, the results of this work can be applied to Knowledge Searching in such bases. These results show the importance of the systematic application of Combinatorial Optimization models to Expert Systems.

Resumen: Se utiliza un modelo de Hipergrafos Dirigidos que permite reducir una derivación en un sistema de Horn al cálculo de una solución factible en un problema de Programación Lineal asociado a la búsqueda de flujos unitarios. Se explora la aplicación de esta herramienta en el cálculo de Factores de Certeza. Como las Bases de Conocimiento de muchos Sistemas Expertos estan expresadas, o se pueden expresar, en cláusulas de Horn con Factores de Certeza asociados, el resultado de este trabajo se puede aplicar directamente en la búsqueda en tales Bases de Conocimiento. Estos resultados muestran la importancia de la aplicación sistemática de los modelos de Optimización Combinatoria al área de Sistemas Expertos

1.- Introduction:

In last few years, Expert Systems [2,5] have become one of the most important areas in Artificial Intelligence. This technology has been developed for a wide variety of practical problems resolution, and has been established as a new programming methodology paradigm. In this way, the knowledge used building a system transcends program limits and become part of the data. That is why Expert Systems are also called Knowledge-based Systems.

Various schemes are used to represent a given Knowledge Base. Such a Knowledge Base is usually modifiable by user interaction and is handled by the system in order to produce new knowledge or to check if a given fact is a consequence of the represented knowledge. Generally, these operations have been seen as a particular case of logical deduction and have been treated with logical methods.

From other point of view, in [11], it was presented a directed hypergraph based model for Horn clauses systems, which leads to the equivalence between logical deduction in such systems and a linear programming problem corresponding to a generalization of flow to directed hypergraphs. With this approach, we consider in this work the inclusion of Certainty Factors in the model.

In section 2, we present the definitions and results of Directed Hypergraphs, Flows and Horn Systems that we use in this work. In section 3, we consider various scheemes for Certainty Factors and how can be implemented in this Linear Model.

2.- Directed Hypergraphs and Horn Systems

In order to work with the dual of linear problems formulated on the incidence matrix of a graph, Berge [1] introduced the definition of hypergraph, a generalization of the concept of graph, which consist of a set of vertices and a family of hyperedges, each one a subset of vertices with cardinality not restricted to be equal to two (as it is on graphs). In this way any matrix with only zero or one valued entries is the incidence matrix of a hypergraph and so it is its transpose. Usually empty hyperedges are undesired, and isolated vertices are also forbidden, in order to assure that the dual holds the same property. The same idea may be trivially extended to directed graphs, leading to the concept of directed hypergraphs or dihypergraphs.

2.1 - *Definition*: Let V be a finite nonempty set, and let $E \subseteq \mathcal{P}(V)^2$. Then the pair $H = (V,E)$ is a directed hypergraph iff $\forall\, e \in E$ such that $e = (t(e),h(e))$:

(i) $t(e) \cap h(e) = \emptyset$

(ii) $t(e) \cup h(e) \neq \emptyset$

An element of V will be called a vertex and an element of E will be called a hyperedge or simply an edge when not leading to confusion. Henceforth, the set of all vertices of a directed hypergraph H will be denoted as V_H and the set of all its edges as E_H, when not leading to confusion the subindex H will be omitted. If $e = (V', V'')$ is an edge of a directed hypergraph, V' will be called the tail of e, and V'' the head of e. The tail of an edge e will also be denoted as $t(e)$ and its head as $h(e)$.

A directed hypergraph is therefore a pair of sets: a set of vertices and a set of hyperedges, where the head and the tail of a hyperedge are allowed to be any pair of disjoint set of vertices, not both empty.

In this work any finite set is allowed to be an index set. A vector A with index on a set I will be denoted by $A = (a_i : i \in I)$, and a_i will be the entry of A indexed by i. A matrix M indexed on two sets I and J, will be denoted by $M = (m_{i,j} : i \in I, j \in J)$, where $m_{i,j}$ will be the entry of M indexed by i and j. M may be interpreted as a vector indexed on the cartesian product of I and J, that is $M = (m_{(i,j)} : (i,j) \in I \times J)$. A function F with domain in a finite set D will be represented by the vector $\bar{F} = (f_d : d \in D)$ where $f_d = F(d)$.

2.2 - *Definition:* Let H be a directed hypergraph. Then the incidence matrix of H will be $\bar{H} = (h_{v,e} : v \in V, e \in E)$, where:

$$h_{v,e} = \begin{cases} -1 & \text{if } (v \in t(e)) \\ 1 & \text{if } (v \in h(e)) \\ 0 & \text{if } (v \notin t(e) \cup h(e)) \end{cases}$$

Any matrix with only 0, 1 and -1 valued entries is then the incidence matrix of a directed hypergraph if it does not have two equal columns or a zero column.

Now we extend the concept of path to directed hypergraphs.

2.3.- *Definition:* A path, from s_0 to s_k ($k \geq 0$), in a directed hypergraph H is a sequence of form:

$$s_0 e_1 s_1 e_2 s_2 \ldots s_{k-1} e_k s_k$$

Where:

(i) $\forall i \in [0..k] (s_i \in \mathcal{P}(V))$

(ii) $\forall i \in [1..k] (e_i \in E)$

(iii) $\forall i \in [1..k] ((t(e_i) \cap s_{i-1} = \emptyset) \wedge (h(e_i) \cap s_i \neq \emptyset))$

The edge sequence associated to a path is $e_1 e_2 \ldots e_k$. The length of a path P (denoted by length(P)) is the total number of occurrences of edges in

its edge sequence, i. e. length($s_0e_1s_1e_2s_2...s_{k-1}e_ks_k$) = k . The set of edges in a path P will be denoted as S(P), i. e. S ($s_0e_1s_1e_2s_2...s_{k-1}e_ks_k$) = { e_i : $i \in [1..k]$ }.

This is a very weak path concept. Next, we introduce the stronger concept of increasing paths where the initial set of vertices s_0 grows in the path by mean of the edges in it.

2.4 - *Definition:* An increasing path in a directed hypergraph H is a path of the form:

$$s_0\, e_1\, s_1\, e_2\, s_2\, \ldots\, s_{k-1}\, e_k\, s_k$$

Where:

(i) $\forall i \in [1..k]\ (t(e_i) \subseteq s_{i-1})$

(ii) $\forall i \in [1..k]\ (s_i = s_{i-1} \cup h(e_i))$

Now we present a generalization of the problem of flow in networks [4] to directed hypergraphs. Most of notation and definitions are extended from [10].

2.5 - *Definition:* Let H be a directed hypergraph. Then a flow F in H is just a function of the edges of H into the real numbers, i. e. $F : E \to \mathbb{R}$. The value F(e) will be called the flux of the edge e. A flow F is an integral flow iff for all e in E, the flux of e is integer, that is Range(F) $\subset \mathbb{N}$. For a flow F, the set of edges with positive flux will be denoted as P(F), i. e. P(F) = { $e \in E : F(e) > 0$ }. A flow F will be called a nonnegative flow iff it has nonnegative flux for all edges, i. e. P(-F) = $\emptyset$.

2.6 - *Definition:* Let H be a directed hypergraph and let F be a flow in H. The divergence function of F is $Div_F : V \to \mathbb{R}$, where:

$$Div_F(v) = \sum_{\{e \in E : v \in h(e)\}} F(e) - \sum_{\{e \in E : v \in t(e)\}} F(e)$$

Note that divergence function for v_i can be calculated by multiplying i^{th} row of the incidence matrix of H by the vector $\bar{F}$. Therefore:

2.7 - *Lemma:* Let H be a directed hypergraph and F a flow in H, then:

$$\bar{H} * \bar{F} = \overline{Div}_F$$

□

It is of special interest the consideration of nonnegative flows with restrictions on the divergences. These restrictions will be, for the divergence of each node, of one of the following types: equal to a given value, less than or equal, greater than or equal, or unrestricted. Now, if R is a vector of restrictions and D a constant vector, both indexed on V_H then the set of nonnegative flows satisfying such restrictions, will be equal to the set of feasible solutions to the linear programming problem $\overline{H} * \overline{F}\ R\ D\ ;\ \overline{F} \geq 0$. Since for the scope of this paper only nonnegative flows are relevant, thereafter whenever a flow appears it will be assumed to be a nonnegative flow.

2.8 - *Definition:* Let H be a directed hypergraph, V' and V'' disjoint subsets of V, and let F be a flow in H. Then F is an <u>unit flow</u> from V' to V'' iff:

(i) $\forall v \in V''$ $(Div_F(v) = 1)$

(ii) $\forall v \in V - (V' \cup V'')$ $(Div_F(v) = 0)$

Note that a unit flow is a special case of restricted flow, so a flow is a unit flow iff it is a feasible solution of the linear problem $\overline{H} * \overline{F}\ R\ D\ ;\ \overline{F} \geq 0$, where r_v is an equality for every vertex in V'' - V' and no restriction for every vertex in V', and d_v is 1 for all vertices in V'' and 0 for all vertices in $V - (V' \cup V'')$. This can be stated in the following lemma:

2.9 - *Lemma:* Let H be a directed hypergraph, V' y V'' two disjoint subsets of V, and F a flow in H. Then F is a unit flow iff it is a feasible solution of the linear problem $\overline{H} * \overline{F}\ R\ D\ ;\ \overline{F} \geq 0$, where R and D are as described above.

□

Now we introduce the logic model we use in this work. It is based in a particular class of clauses, very important for the simplicity of the derivations with them and their suitableness for expressing most of knowledge, are Horn clauses [6] in which there is only one atomic proposition at the right side, i. e. clauses of the form: $\wedge_{\{p \in P\}} p \rightarrow p'$. If a formula is a finite conjunction of Horn

clauses it is said that it is in Horn clausular form. Note that a clause of the form: $\wedge_{(p \in P)} p \rightarrow p'$, where $p' \in P$ is a tautology, so any formula in Horn clausular form can be assume to have not such clauses.

2.10.- *Definition:* Let P be a set of atomic propositions then a Horn system S is a pair (P,C), where C is a set of nontautological Horn clauses referencing only the atomic propositions in P.

A Horn system is then naturally modelled by a directed hypergraph with all edges having a unitary head. In this model vertices represent the atomic propositions of the system and edges represent the clauses.

2.11.- *Definition:* A directed hypergraph H is a rule hypergraph iff:

$$\forall e \in E (|h(e)| = 1)$$

Any edge of H will be called a rule and any vertex an atom of H. A rule of the form $(T, \{v_h\})$ will be denoted as $T \rightarrow v_h$. Any subset of V will be an information state of H, and the set of all information states of H will be denoted by IS_H. ($IS_H = \mathcal{P}(V)$).

Then every atomic proposition p of a logical system will be represented by an atom v_p and every clause of the form: $\wedge_{(p \in P)} p \rightarrow p'$ ($p \notin P$) by a rule $e = V_P \rightarrow v_{p'}$, where $V_P = \{ v_p : p \in P \}$. In this way any Horn system S can be modelled by a rule hypergraph, that will be denoted H(S).

Now, we introduce a path-based concept of derivation:

2.12.- *Definition:* Let H be a rule hypergraph. A derivation of d from O in H, where d is a vertex and O an information state, is an increasing path, of the form

$$s_0 e_1 s_1 e_2 s_2 \ldots s_{k-1} e_k s_k$$

Where:

(i) $s_0 = O$

(ii) $d \in s_k - O$

The edge sequence of a derivation will be called rule sequence. A rule contained in the rule sequence of a derivation is said to be applied in it. A rule e is said to be applicable given an information state O iff t(e) ⊆ O.

Note that the concept of derivation in rule hypergraphs corresponds to that of logical derivation with Horn clauses by *modus ponens*. If information states are interpreted as the conjunction of the propositions represented by its atoms, which are assumed to be true, then a rule is applicable if its preconditions are true, and the set of true propositions after the rule is applied is augmented by its consequent.

This completes the path leading to the main results of [11], that state that existence of a derivation in a rule hypergraph is equivalent to existence of a feasible unit flow in it. This reduces the problem of logical deduction with propositional Horn clauses to a linear programming problem.

2.13.- *Theorem:* ([11]) Let S = (P,C) be a Horn system and let P' be a subset of P. Then the nontautological Horn clause $\wedge_{\{p' \in P'\}} p' \rightarrow p''$ is logically derivable from the clauses in S iff there is an unit flow from P' to p'' in H.

□

2.14.- *Theorem:* ([11]) Let S = (P,C) be a Horn system and let P' be a subset of P. Then the nontautological Horn clause $\wedge_{\{p' \in P'\}} p' \rightarrow p''$ is logically derivable from the clauses in S iff there is a feasible solution of the linear problem $\overline{H(S)} * \overline{F}$ R D ; $\overline{F} \geq 0$, where r_v is an equality for every vertex in $V - \{ v_{p'} : p' \in P' \}$ and no restriction for every vertex in $\{ v_{p'} : p' \in P' \}$, and d_v is 1 for $v_{p''}$ and is equal to 0 for all vertices in $V - (\{ v_{p'} : p' \in P' \} \cup \{ p'' \})$.

□

3.- Computing Certainty Factors in the Linear Model

Several Certainty Factors Algebras have been developed to deal with uncertain knowledge in Expert Systems. The most important models ones are the models given by Probabilistic Logic [9] and Fuzzy Logic [8,12], even when many others exist [3,7]

Using the result of (2.11) any Certainty Factors Algebra can be reduced to the optimization of a functional over a linearly bounded space. It is of special interest the models that can be translated to linear functions thus reducing the searching problem in Knowledge Bases with uncertainty to linear programming problems. In this sense, even when Probabilistic Logic produces a complex product functional, Fuzzy Logic allows a linear formulation, that we present in this section.

In Fuzzy Logic the Certainty Factors of a derivation is the minimum of all the factors of the rules and facts used in the derivation. Hence, the problem is to find a derivation which maximize this minimum, Since we consider facts as special rules, let α_i be the Certainty Factor of rule i. Our problem is then:

$$\max \{ \min \{ \alpha_i : \text{rule i is used in solution } \bar{F} \} : \bar{F} \text{ is a solution} \}$$

This is a max-min problem with setup costs α_i if $f_i > 0$ or 0 if $f_i = 0$ over solutions F to $\overline{H(S)} * \bar{F} \; R \; D$; $\bar{F} \geq 0$ by theorem (2.14), where R is as described in such theorem.

Let $\bar{X}$ be an integer vector with indexes on the rules of S, where x_i is equal to 0 if $f_i = 0$, and is equal to 1 otherwise. The feasible set is:

$$3.1.- \quad \overline{H(S)} * \bar{F} \; R \; D \; ; \; \bar{F} \geq 0 \; ; \; \bar{F} \leq M * \bar{X}$$

where M is a bound for f_i (like n, the number of rules in S)

Now then, the functional is:

$$\max \{ \min \{ \alpha_i : x_i = 1 \} : (\bar{F},\bar{X}) \text{ is a solution of (3.i)} \}$$

But; $\min \{ \alpha_i : x_i = 1 \} = \min \{ \alpha_i * x_i + L * (1 - x_i) : i = 1 .. n \}$, where: $L > \max \{ \alpha_i : i = 1 .. n \}$.

Therefore, to find the best Certainty Factor is equivalent to the problem:

$$\max \min \{ \alpha_i * x_i + L * (1 - x_i) : i = 1 .. n \}$$

Over:

$$\overline{H(S)} * \bar{F} \; R \; D \; ; \quad \bar{F} \geq 0 \; ; \quad \bar{F} \leq M * \bar{X} \; ; \quad \bar{X} \in (0,1)^n$$

And this problem can be solved as a mixed integer linear programming, since any max-min functional can be translated to a linear one. This can be stated, for this case as follows:

3.2.- *Theorem:* Let S = (P,C) be a Horn system with Fuzzy Certainty Factors associated to each rule i. Then the Certainty factor of the clause $\Lambda_{\{ p' \in P' \}} p' \rightarrow p''$ is:

$$\max b$$

Over

$$b \leq \alpha_i * x_i + L * (1 - x_i) : i = 1 .. n ;$$
$$b \geq 0 ;$$
$$\overline{H(S)} * \bar{F} \; R \; D \; ;$$
$$\bar{F} \geq 0 \; ,$$
$$\bar{F} \leq M * \bar{X} \; ,$$
$$\bar{X} \in (0,1)^n$$

Where $L > \max \{ \alpha_i : i = 1 .. n \}$, r_v is an equality for every vertex in $V - \{ v_{p'} : p' \in P' \}$ and no restriction for every vertex in $\{ v_{p'} : p' \in P' \}$, d_v is 1 for $v_{p''}$ and is equal to 0 for all vertices in $V - (\{ v_{p'} : p' \in P' \} \cup \{ p'' \})$, and M is a bound for f_i (like n, the number of rules in S).

□

5.- Conclusions:

In this paper we have shown the usefulness of looking at logical derivation and calculation of Certainty Factors in Expert and Knowledge-based Systems from a Combinatorial Optimization point of view . In particular, the problem of derivation with propositional Horn clauses with Fuzzy Certainty Factors is reduced, by theorem (4.7), to a mixed integer linear programming problem associated to the existence of a Unit Flow in Directed Hypergraph. Such result is not only of theoretical interest, but also practical, since many Knowledge-based Systems or big subsystems of them can be modelled or translated to propositional Horn systems, and Fuzzy Logic is one of the most widely used Certainty Factor Algebra

This work opens two lines for future research. First, the study of more general or structured combinatorial models for searching in Knowledge Bases with Uncertainty. Second, the inclusion of other Certainty Factor Algebras in the linear model.

References

[1] Berge, C., *Graphes et Hypergraphes,* Second Edition, Dunod, Paris, 1976.

[2] Buchanan, B. G., and Shortliffe, E. H. (Editors), *Rule-Based Expert Systems,* Addison Wesley, Massachusetts, 1984.

[3] Duda, R. O., Hart, P. E., and Nilson, N. J., "Subjective Bayesian Methods for Rule-Based Inference Systems", TR-124, Artificial Intelligence Center, Stanford Research Institute, California, 1976.

[4] Ford, L. R., and Fulkerson, D. R., *Flows in Networks,* Princeton University Press, New Jersey, 1962.

[5] Hayes-Roth, F., Waterman, D. A., and Lenat, D. (Editors), *Building Expert Systems,* Addison-Wesley, 1983.

[6] Horn, A., "On Sentences Which are True of Direct Unions of Algebras", *Journal of Symbolic Logic,* Vol. 16, Nº 14, 1951.

[7] Michalski, R. S., and Winston, P. H., "Variable Precision Logic", *Artificial Intelligence,* Vol. 29, Nº 2, pp. 121-146, 1986.

[8] Negoita, C. V., and Ralescu, D. A., *Applications of Fuzzy Sets to Systems Analysis,* John Wiley & Sons, 1975.

[9] Nilsson, N. J., "Probabilistic Logic", *Artificial Intelligence,* Vol. 28, Nº 1, pp. 71-87, 1986.

[10] Rockafellar, R. T., *Network Flows and Monotropic Optimization,* John Wiley and Sons, 1984.

[11] Torres, A., and Aráoz, D., "Combinatorial models for searching in Knowledge Bases", Acepted for publication in *Acta Cientifica Venezolana.*

[12] Zadeh L. A., "Fuzzy Sets", *Information and Control,* Vol. 8, pp. 338-353, 1965.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa.
Buenos Aires, Setiembre 1988.

VERIFICAÇÃO DE BASE DE CONHECIMENTO

JOSE CARLOS BARBOSA DAMSKI

SMART SYSTEMS CONSULT. E INF. S/C LTDA

Av. Francisco José Longo 149/117

cep 12245 São José dos Campos - S.P.

Tel. (Ø123) 22-3244

RESUMO : No presente trabalho é mostrado como verificar Bases de Conhecimento para Sistemas Especialistas, juntamente com a Aquisição de Conhecimento. E visto tambem como implementar um sistema que faça parte da verificação e análise da B.C.

INTRODUÇAO :

Dentro da área de Inteligência Artificial, os Sistemas Especialistas destacam-se pelas várias aplicações em que podem atuar de forma simples e flexível.

Existem muitos ambientes para desenvolvimento de Sistemas Especialistas disponíveis hoje no mercado internacional, para diversos equipamentos (de grande a pequeno porte) e variadas finalidades (desde de ambientes demonstrativos até ambientes profissionais).

Com o uso destes ambientes são desenvolvidas muitas aplicações, desde a escolha de um vinho até a configuraçaõ automática de ambientes computacionais.

Mas, curiosamente, muitos destes ambientes não auxiliam em duas fases importantes na construção de Sistemas Especialistas, que são: a Aquisição de Conhecimento e a Verificação da Base de Conhecimento.

A Aquisição de Conhecimento é um problema muito complexo que aborda desde a relação homem x homem até a filosofia (epistemologia). Neste trabalho não abordaremos objetivamente este assunto.

Depois do "conhecimento" adquirido, este é expresso em uma (várias) representação(ções) formando a Base de Conhecimento.

A partir daí pode-se desenvolver testes para a

verificação (quanto a correção) desta Base.

Nosso trabalho visa apresentar soluções para esta etapa, tão importante para a confecção de Sistemas Especialistas.

DESENVOLVIMENTO DE SISTEMAS ESPECIALISTAS

Existem diversas formas de representar "conhecimento" em Sistema Especialista: regras de produção, quadros ("frames"), redes semânticas, etc. A mais difundida atualmente é a de regras de produção, inclusive por sua semelhança ao cáculo proposicional e de predicados da lógica matemática.

Dentro do paradigma de representação por regras, existem diversas formas sintáticas de expressá-las. Existem tambem diversas formas de fazer inferências e calcular o valor das condições das regras.

Nosso escopo de trabalho é em regras sob uma forma sintática geral o suficiente para servir de exemplo de solução para outras formas sintáticas.

Acreditamos que o sistema de verificação de Base de Conhecimento deve ser feito "ad hoc", i.e., para cada ambiente de desenvolvimento de Sistemas Especialistas deve ser feito um particular sistema de verificação de Base de Conhecimento.

Para confeccionar um Sistema Especialista é necessário passar pelas fases :

1. Aquisição do Conhecimento
2. Formalização do Conhecimento
3. Verificação da Base de Conhecimento
4. Teste de campo do Sistema Especialista
5. Uso

que podem fazer ciclos entre diversas dessas fases. A fase 3 auxilia a fase 1 e garante melhor resultado em 4 e 5.

A fase de verificação é importante, pois a Aquisição de Conhecimento não garante que todo o conhecimento (ou que se deseja) está expresso e se esta correto semântica e pragmáticamente em relação ao especialista.

Tipos de Verificação

Para se concretizar a fase 3 do tópico anterior, dividimos a validação em 3 etapas:

a) Validação em tempo de concepção das regras

a.1) Quando a regra é concebida, por quem quer que seja, o Engenheiro de Conhecimento deve pedir a aprovação por parte do especialista quanto a semântica das condições que compõem a regra. Deve ser feita também a avaliação pragmática da regra.

a.2) O especialista deve ser solicitado a avaliar tambem os cachos decisoriais (conjunto pequeno de regras que levam

a um objetivo).

b) Validação em tempo de prototipação

b.1) Quando o Sistema Especialista está na fase de protótipo, pode-se fazer teste de validação colocando o Especialista frente a frente com o Sistema Especialista. O Especialista deve fazer todos os testes que julgar conveniente e lançar mão dos recursos que o ambiente computacional pode lhe oferecer : explicação do raciocinio usado; mostrar todos os caminhos que o Sistema Especialista tenta para obter a solução, etc.

b.2) Uma variante do caso anterior é o Engenheiro de Conhecimento criar situações de teste e oferecer, separadamente, ao Sistema Especialista e ao Especialista e posteriormente confrontar as respostas.

c) Validação Automatizada

Nos 2 itens anteriores os métodos de validação estão centrados no homem, que por um lado é bom devido a sua grande capacidade de análise e por outro é ruim pois pode ficar incompleto e não abarcar toda a Base de conchecimento. Visto que é praticamente inviável um Engenheiro de Conhecimento conceber toda a Base de Conhecimento em sua memória (para B.C. de tamanho médio, mais de 500 regras).

Para contornar este problema, deve-se desenvolver uma ferramenta computacional que faça a análise da Base de Conhecimento, não excludente com as feitas nos itens a e b acima, mas que tenham um enfoque mais sintático e exaustivo em

relação a base total.

Este sistema deve atuar em:

1. Verificação da consistência

1.1 Regras redundantes

1.2 Regras conflitantes

1.3 Regras mais gerais (que outras)

1.4 Condições desnecessárias (nas premissas)

1.5 Detecção de encadeamento ciclico nas regras

2. Verificação da Completeza

2.1 Referência de atributos fora do escopo (contexto)

2.2 A não referência de atributos do escopo

2.3 O encadeamento completo das regras

2.4 Mostrar as condições não dedutiveis pelas regras

2.5 Verificar se todas as conclusões finais recaem sobre o universo final da aplicação

3. Análise dos Fatores de Certeza

3.1 Conclusões com F.C. conflitantes

3.2 Verificação dos problemas citados no item 1 acima, levando em consideração os F.C.

3.3 Conclusões que nunca atingem o limiar de veracidade

No item "c" acima, os sub-itens 1 e 2 são discutidos no próximo tópico e o sub-item 3 em [5].

IMPLEMENTAÇÃO DA VERIFICAÇÃO MECANIZADA

Nossa implementação é baseada na representação de regras de preodução na forma :

SE

P1 (T1,T2, ..., Tn) E

P2 (T1,T2, ..., Tm) E

ENTÃO

Pn (T1,T2, ..., Tj)

onde Pi são simbolos predicativos e Ti são termos que pode ser constantes ou variaveis (estas são expressas por um "*" antes do nome).

Além do conjunto de regras, temos também a estrutura do dicionário.

O dicionário é constituido pela definição do escopo de cada termo. O escopo é o conjunto de valores (atômicos) que tornam o predicado verdadeiro. Na realidade este tipo de definição já existe em alguns ambientes de desenvolvimento de Sistemas Especialistas, como sendo a definição de um parametro. Por ex:

COR (*X)

é definido no dicionário todas as cores que são válidas no lugar da variável "x".

A partir deste conceito resolveremos muitos problemas [3], conforme um deles, abaixo.

Um dos problemas é a ocorrência de ciclos ("loops") durante o processo de inferência sobre uma Base de Conhecimento. Por exemplo, suponha a seguinte Base de Conhecimento com 3 regras:

R1	-	SE	A(*X)
		ENTÃO	B(*X)
R2	-	SE	C(*Y)
		ENTÃO	A(*Y)
R3	-	SE	B(*Z)
		ENTÃO	C(*Z)

em inferência regressiva (encadeamento para trás). Notamos claramente o ciclo a partir da regra 1, que para resolver B(*X) é necessário resolver A(*X). Para resolver A(*x), unifica-se com a conclusão da regra 2, tornando-se necessário resolver C(*Y). Resolvendo com a regra 3, precisa-se resolver B(*Z) voltando a regra 1 fechando o ciclo.

Para saber se este ciclo é possivel de acontecer, e caso seja, em qual escopo, devemos calcular: (E abrevia escopo ; # simboliza intersecção)

1. Na regra 1 : E [*X] = E[B(*X)] # E[A(*X)]
2. fazendo *Y = *X temos E[A(*Y)] = E[*X]
3. Na regra 2 : E[*Y] = E[A(*Y)] # E[C(*Y)]
4. fazendo *Z = *Y temos E[C(*Z)] = E[*Y]
5. Na regra 3 : E[*Z] = E[C(*Z)] # E[B(*Z)]

Se E[*Z] # E[B(*X)] ≠ vazio então

O ciclo pode acontecer no escopo resultante da intersecção

Se-não

O ciclo nunca irá acontecer.

Como foi visto nesta solução, existem outros problemas que podem ser resolvidos de maneira semelhante, conforme detalhado em [3].

CONCLUSÃO

Poucos são os sistemas que possuem um módulo para verificação da Base de Conhecimento.

Devido a diversidade de formas de escrevermos regras de produção, acreditamos que cada usuário deva desenvolver seu módulo de verificação automatizada e executar os processos manuais com bastante frequência e critério.

Continuaremos nosso trabalho no sentido de dar mais poder de expressão ao escopo e tentar resolver o problema dos predicados pré-definidos (extra-lógicos) que não tem (pode não ter) uma definição precisa de seu escopo.

REFERENCIAS BIBLIOGRAFICAS :

1. CASANOVA, M.A.; GIORNO, F.A.C.; FURTADO, A.L. , Programação em Lógica, V Escola de Computação, 1986.
2. CHANG, CHIN-LIANG ; LEE, R.C. , Simbolic Logic and

Mechanical Theorem Proving, Academic Press, 1973.

3. DAMSKI, J.C.B. ; OMAR, N. - Um sistema para análise de uma Base de Conhecimento, Anais do IV SBIA, Uberlândia, 279-286,1987.

4. GONÇALVES, C.A. , Aquisição e Representação do Conhecimento para Sistemas Especialistas, Tese de Doutorado, FEA/USP, 1986.

5. SENNE, E.L.F., Compatibilidade de regras de produção usadas como representação do conhecimento em sistemas de consulta, Tese de Mestrado, ITA, 1980.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

Una Interfase de Lenguaje Natural para Programar

Adolfo Kvitca
Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires, Argentina

> ...When I use a word -Humpty Dumpty said, in a rather scornful tone- it means just what I choose it to mean, neither more nor less...
> Lewis Caroll, Through the looking glass.

Resumen:

En este trabajo se describe una Interfase de Lenguaje Natural (ILN) que permite crear programas a partir de la información contenida en una gramática semántica que determina el subconjunto de LN apropiado para interactuar con el sistema y la aplicación.

Un postulado básico del proyecto consiste en obtener un sistema cuya principal característica resida en la sencillez de utilización y la rapidez de adaptación, no siendo necesario conocimientos especializados sobre linguística para especificar la gramática.

1. Ventajas de una Interfase de Lenguaje Natural p/Programar:

Una gran capacidad de almacenamiento y un sencillísimo juego de instrucciones, son suficientes para construir cualquier programa, incluida la simulación completa de un nuevo computador (Turing 1930) [MAN/74]. Sin embargo, la noción que un computador cualquiera pueda simular a otro cualquiera, tanto existente como futuro, puede tal vez tener importancia filosófica, pero no es la solución de todos los problemas de computación: para programar es no sólo conveniente sino necesario que el programador cuente con mecanismos de abstracción [ABE/85].

Los diferentes lenguajes de programación ofrecen múltiples maneras de especificar una computación, transformando al computador en una máquina virtual cuyas características y capacidades son determinadas por la programación. Aunque el soporte material no ha cambiado, el programador puede pensar en términos de variables y no en celdas de memoria, en archivos de datos y no en canales de entrada/salida, en fórmulas algebráicas y no en registros y sumadores...

La utilización de una ILN para programar facilita y potencia las capacidades de expresión y comprensión de programas. Despues de todo, la manera más fácil de comunicarse con una computadora es la manera en que nos comunicamos con las personas.

Este trabajo ha sido financiado por el Programa Nacional de Informática y Electrónica (0190-243/86) SECYT - Argentina

Su uso presenta además las siguientes ventajas:

- El Lenguaje Natural, por su naturaleza declarativa, es el medio adecuado para proveer la descripción de problemas facilitando su expresión en terminos cercanos a la concepción original. (Los lenguajes de programación convencionales estan orientados a los procesos, no describen el problema sólo un método para encontrar la solución). [SCH/86]
- Aunque no sepa programar, cualquier persona puede entender que hace un programa solamente leyendo el codigo. Esta característica facilita también, el desarrollo de módulos por varios programadores y el mantenimiento de programas durante el ciclo de vida de un sistema.
- Al no tener que aprender rígidos detalles de sintaxis, se hace mas sencilla la tarea de programación, sobre todo, a niños y usuarios ocasionales.
- Es posible utilizarla para comunicarse con el usuario de la aplicación, proveyendo una ILN en Sistemas de Consulta a Bases de Datos, Sistemas Expertos, Aplicaciones Educativas, etc.
- La interfase constituye una poderosa herramienta para realizar complejos programas de una manera "natural" debido a su capacidad de realizar inferencias, determinar contextos y manipular complejas estructuras de datos.
- La utilización de la terminología propia del dominio de aplicación, que incluye conceptos intermedios de alto nivel, es esencial para razonar efectivamente y comunicarse con el usuario, facilitando el tratamiento de lineas complejas de razonamiento, tanto para el usuario como para el sistema. [KVI/88]

2. Antecedentes:

La técnicas actuales de la IA permiten diseñar programas que procesan satisfactoriamente frases del usuario restringidas de alguna forma: a) el usuario está limitado en la variación estructural de sus sentencias (sintaxis restringida por una gramática artificial) o, b) en el número de cosas que pueden significar (dominios con semántica restringida). [BOL/81]

La utilización de LN fue propuesta por muchos autores [BAL/75],[GRE/77],[WOO/77] como la mejor manera para programar. Entre ellos merecen ser destacados los siguientes sistemas: El sistema SAFE [BAL/77] permite realizar especificaciones de programas que pueden estar incompletas o ambigüas, pero en una sintaxis y vocabulario limitado. TEIRESIAS [DAV/82] es un sistema que permite al usuario el ingreso y depuración de conocimiento para un sistema experto de diagnóstico médico, en un subconjunto reducido de LN. En [YEO/84], [DAH/81] se describe como traducir un subconjunto de LN a lógica procedural. Simmons [SIM/84] muestra una aplicación de ayuda a la documentación de programas, utilizando una gramática para hacer corresponder procedimientos a descripciones en LN. Heidorn [HEI/76] desarrolló un sistema que permite especificar "problemas de colas", mediante diálogos interactivos en LN. En [VAN/80] se describe un sistema para producir funciones Lisp a partir de descripciones en LN. Biermann y Ballard comparan la eficacia de programación en lenguaje natural en la manipulación de matrices, concluyendo que puede programarse más rápido y con menos errores en LN [BIE/76].

Bertram Raphael [BER/68] propuso la idea de comparar frases contra plantillas fijas, y el sistema de consulta a Bases de Datos, LIFER [HEN/76] ha retomado esta idea. El autor ha desarrollado también un sistema de consulta a base de datos que utiliza estas ideas [KVI/84].

Como se vé, estos sistemas presentan interfases limitadas. El uso del lenguaje en forma fluida, típico de los humanos, es aún elusivo y constituye un área activa de investigación en Inteligencia Artificial. [SIM/86].

3. Dificultades para la utlización de una ILN para Programar:

El término "Lenguaje Natural" es muy vago. Cada uno uno tiene una intuición de qué significa comunicarse en "Lenguaje Natural", pero es difícil hacer esta noción precisa... es difícil separar las capacidades linguísticas de otras capacidades humanas como la memoria, el razonamiento, la resolución de problemas, la formación de hipótesis, la clasificacion, el planeamiento, el aprendizaje.

Por un lado, sin estas capacidades un sistema de computación usará el lenguaje en forma diferente que un humano y parecerá que no usa realmente el lenguaje, por otro lado no podemos esperar a implementar todas estas capacidades para hacer algo en términos de comunicación en "Lenguaje Natural".

Para realizar una ILN "competente", se necesita no sólo una gramática con la correspondiente sintaxis y semántica del lenguaje, sino también, y quizás más importante, la capacidad de realizar inferencias a partir del conocimiento del dominio de discurso y sobre las intenciones y objetivos del usuario [WIN/72], [WOO/77].

Estas características determinan que realizar una ILN sea una tarea muy compleja, que requiere mucho esfuerzo, y sólo para ser aplicada, como dijimos antes, a un dominio limitado y un subconjunto limitado del lenguaje.

Este alcance limitado, determina que los usuarios de estos sistemas tengan dificultades para aprender qué constituye una entrada aceptable al sistema, pues en lugar de aprender una sintaxis relativemente simple de un lenguaje de programación claramente definido, deben aprender un voluminoso y muy detallado conjunto de reglas de qué palabras y frases pueden ser usadas y cómo pueden ser combinadas. Se argumenta además, que nadie utilizaría LN para programar, aún si existiese, pues sería demasiado verborrágico. Finalmente se sostiene que si se desea manipular información en forma precisa y eficiente en una máquina, debe ser usado un lenguaje claramente definido y no ambiguo, lo que invalida la utilización de LN. [BIE/80].

4. Propuesta para una ILN para Programar:

Teniendo en cuenta las consideraciones anteriores, se pensó que era necesario idear un mecanismo para diseñar ILN cuyas principales características sean:

- sencillez de empleo
- rapidez de adaptación a diferentes dominios
- no debe requerir conocimientos linguísticos para ser definida

aunque esto fuera en detrimento de la potencia y generalidad de la interfase.

Estas características permitirían que la interfase sea definida por el mismo programador de la aplicación, evitándose el inconveniente del aprendizaje de la sintaxis.

La rapidez de adaptación a diferentes dominios aseguraría un amplio espectro de uso y permitiría enfrentar efectivamente el problema del alcance limitado respecto al dominio y subconjunto de LN, pues cada aplicación corresponde a un reducido dominio expresable mediante un conjunto pequeño de frases de LN.

Para disminuir a un mínimo la verborragia se debía manejar distintas formas de elipsis y coordinación, además de la posibilidad de realizar inferencias, deducir información implícita y utilizar la terminología del dominio de aplicación.

5. Aspectos de Diseño:

Especificar en forma abstracta las estructuras de datos y los algoritmos de un programa requiere una notación formalizada, en la cual el significado de cada enunciado lícito este definido con precisión, sin ambigüedad. Esto no impide, sin embargo, la utilización de una sintaxis natural, que permita más libertad para expresar conceptos. [WIN/72], [WOO/70]. Un postulado básico del proyecto es que el análisis del lenguaje puede hacerse mediante transformaciones de frases que utilicen plantillas definibles por el usuario (esta hipótesis no es un intento de proponer bases psicológicas para un modelo del procesamiento del lenguaje natural en el hombre, antecedentes sobre estas ideas se pueden encontrar en [CHO/68] donde se discute los méritos del uso de sentencias primitivas).

Otra hipótesis fundamental, consiste en que el usuario sabe utilizar LN con sus reglas de concordancia y los mecanismos de elisión que le son propios, por lo tanto, no es necesario controlar estos aspectos sintácticos. Los aspectos semánticos, que determinan el significado de las expresiones utilizadas, son los importantes, razón por lo cual se utiliza una gramática semántica para expresar en forma no ambigüa este aspecto.

Finalmente se eligió el lenguaje PROLOG [KOW/79],[CLO/81] como lenguaje de implementación y lenguaje objeto al que se traducirían las frases de LN, pues permite al programador organizar y especificar lo que intenta que el programa realice en terminos cercanos a la concepción original facilitando entonces la especificación del mismo en LN.

6. Gramáticas Semánticas:

Una gramática semántica es una gramática libre de contexto donde los símbolos no terminales y las reglas de producción estan gobernados principalmente por funciones semánticas, es decir que las clases gramaticales tradicionales son reemplazadas por clases motivadas por conceptos del dominio de aplicación [RIC/83].

Las características principales de las gramáticas semánticas que determinan su utilización en este trabajo son:

- Pueden ser utilizadas inmediatemente, sin requerir una interpretación semántica "a posteriori".
- Muchas de las ambigüedades que aparecen durante un "parsing" estrictamente sintáctico pueden ser evitadas, pues las interpretaciones que no tienen "sentido" no son generadas.
- Es más fácil de construir pues no requiere conocimientos linguisticos especializados.
- La desventaja consiste en su alcance limitado y la dificultad de generalización, por lo que generalmente son extensas. Sin embargo, las utilización en dominios reducidos, como corresponde a cada programa de aplicación, reduce a un mínimo estos inconvenientes.

7. Implementación de la Capacidad de Elipsis y Coordinación:

La capacidad de realizar elipsis, es decir omitir en las frases una o más palabras y/o reemplazarlas por referentes (Ej: ¿Cuánto gana Raúl y cuánto Jorge?; Calcular la raiz cuadrada de 2 e imprimirla), y la capacidad de coordinación, que corresponde a construcciones con conjunciones (Ej: Juan y Raúl van al cine) son en general difíciles de tratar pues es necesario utilizar información sintáctica y semántica para hacerlo (Ej: la frase "a y b son rectas" es equivalente a decir "a es una recta" y "b es una recta", en cambio la frase "a y b son paralelas" no debe interpretarse como "a es una paralela" y "b es una paralela"). [DAH/86], [MEL/85], [FON/85].

El enfoque de esta propuesta consiste en utilizar un conjunto de reglas heuristicas que, si bien no cubren todos los casos, se comportan adecuadamente en la mayoría, y que permiten que las excepciones sean "reconocidas" facilmente por el usuario. Estas heuristicas estan basadas fuertemente en la hipótesis que el usuario domina LN y que utiliza adecuadamente la elisión y coordinación. De esta manera aunque se reconocen frases sintácticamente mal formadas, se supone que estas no son utilizadas por el usuario, y las bien formadas son aceptadas en forma correcta por el sistema.

Los mecanismos de elipsis y coordinación se basan en un principio común:

> Se puede omitir toda palabra que ya haya aparecido en el contexto, así como todo artículo definido y preposición.

El contexto corresponde en general a la frase, pero puede especificarse alcances mayores, es decir con respecto a frases anteriores.

Para facilitar el uso, todas las frases son "filtradas" de manera de separar palabras como "del" en "de el", "al" en "a el", etc. Además las palabras correspondientes a acciones terminadas en "lo/la" como "imprimirlo", "ponerla", etc. son separadas de este sufijo, pues se reconoce esta caracteristica mediante las definiciones de la gramática.

Todas las conjugaciones de los verbos "ser" y "estar" pueden ser usadas indistintamente e inclusive pueden ser ignoradas si no son "esperadas".

Las reglas anteriores permiten resolver satisfactoriamente casos como:

¿Cuánto gana Raul y cuánto Jorge?.
Calcular la raiz cuadrada de 2 e imprimirla.
Hay una evidencia fuertemente sugestiva de que un organismo es de la clase neisseria si proviene de un lugar normalmente esteril y tiene un tinte gram negativo y forma esferica.

que se convierten respectivamente a las siguientes cláusulas:

```
gana(Raul,X),gana(Jorge,Y).
raiz(2,X),imprimir(X).
evidencia(clase(ORGANISMO,neisseria),5):-
  lugar(ORGANISMO,LUGAR),tipo(LUGAR,esteril),
  tinte_gram(ORGANISMO,negativo),forma(ORGANISMO,coco).
```

La capacidad de coordinación para términos que aún no aparecieron se obtiene mediante un segundo principio:

En todo lugar donde se "espera" un objeto puede venir una conjunción de objetos del mismo tipo, y la frase será interpretada como correspondiente a cada objeto.

Esto permite interpretar la frase:

Juan y Raul van al cine.

como:

Juan va al cine y Raul va al cine.

Este principo no resuelve el problema planteado mediante el ejemplo de "las paralelas", pero el usuario reconoce facilmente este tipo de excepciones.

9. Descripción del Sistema:

El sistema esta compuesto por tres módulos:

1. Módulo de Definición (MD): el diseñador de la aplicación define el subconjunto de LN apropiado para una aplicación, mediante una gramática semántica. La misma esta compuesta por las declaraciones correspondientes a la estructura jerárquica de los objetos típicos del dominio, y la terminología del dominio compuesta por las relaciones y funciones que se pueden establecer en el mismo. Es interesante notar que el MD cuenta con una interfase de LN construida con el mismo sistema.

2. Módulo de Traducción (MT): es generado automáticamente a partir de la definición de la gramática y utilizado para traducir el programa de LN a PROLOG. Es posible utilizar otros lenguajes objeto, por ejemplo comandos de SQL, mediante el simple recurso de invocarlos a través de un comando en lenguaje PROLOG.

3. Módulo de Generación (MG): es generado automáticamente a partir de la definición de la gramática y utilizado para traducir el programa PROLOG a lenguaje natural, necesario para fines de depuración del programa y exhibición de trazas para explicaciones. En la generación de código no se realiza elipsis ni concordancia pues el sistema no conoce las reglas sintácticas y semánticas correspondientes, que sí conocía el usuario cuando ingresó las frases. Además, al utilizarse por ejemplo en trazas de ejecución, las frases pierden el contexto original y no sería posible mantener las características de elisión y concordancia originales.

A continuación se exhibirán las características del sistema dando ejemplos de utilización del MD y comportamiento del MT generado. El MD reconoce como variables aquellas palabras que tengan todas sus letras en mayúscula, en caso contrario serán constantes. En las frases traducidas por el MT serán reconocidas como variables, los nombres de las clases declaradas en el MD (seguidas opcionalmente por un número) y las palabras de una sola letra. La utilización de paréntesis en las plantillas denota, en todos los casos, información de uso opcional. Los comentarios serán indicados entre {}. Las definiciones ingresadas a cada módulo serán precedidas por el nombre del módulo y una flecha "<-" (Ej: "MD<-"), la salida obtenida será indicada mediante el nombre del módulo y una flecha "->" (Ej: "MT->". La lectura del resto del trabajo puede requerir conocimientos del lenguaje PROLOG [CLA/82], [STE/86].

9.1 Definición de la estructura jerárquica de los objetos:

Mediante esta información se establece la estructura de los objetos del dominio y sus relaciones jerárquicas. Con estas simples declaraciones es posible procesar las siguientes características del LN:

a) interpretar anomalías semánticas
   (corresponde a controlar que los argumentos sean del "tipo" esperado)
b) utilización de objetos estructurados
   (Por ejemplo, "un camión con capacidad 2000 kg, color azul, chapa 3820264, trayecto con origen en Buenos Aires y destino en Mar del Plata")
c) independencia del orden de los argumentos
   (la frase "un rectángulo de ancho 10 cm y alto 20mm" es interpretada en la misma forma que "un rectángulo de alto 20 mm y ancho 10 cm")
d) generación de código dependiente de los datos
   (el MT genera el código adecuado en función de los argumentos recibidos, por ejemplo interpretando adecuadamente que "mover un canguro" corresponde a "saltar" y "mover una víbora" corresponde a "reptar")
e) herencia de información y estructuras
   (la declaración de una jerarquía de clases permite "heredar" la información y las estructuras correspondientes en forma similar al mecanismo implementado mediante "frames" [MIN/75], [KVI/84], [BOW/75], pero sin la posibilidad de redefinir información pues esto impone problemas de no-monotonía)

f) valores asumidos en forma implícita y métodos de obtención de la información asociados a los atributos de los objetos.
(la declaración de objetos estructurados permite especificar los valores típicos a asumir "valores default" y los métodos de obtención de información "valores if-needed" [WIN/75]).
g) incorporar conocimiento típico del dominio
(es posible incorporar objetos, relaciones y fórmulas típicas del dominio de aplicación. Ej: información correspondiente a registros de personal, conversiones entre unidades de medidas, etc.)

Objetos simples:
La plantilla general para declarar "objetos simples" es la siguiente:

ART_INDEF OBJETO (SALIDA<-ENTRADA):-EXPRESION_PROLOG

Donde,
ART_INDEF: corresponde a un artículo indefinido (un, uno, unos, una, unas) que son reconocidos como prefijos para la declaración de objetos.
OBJETO: es el nombre que se utilizará para referenciar a la clase correspondiente.
(SALIDA<-ENTRADA): indica el "patron" que deben seguir las palabras correspondientes a la ENTRADA de la clase declarada y la correspondiente conversión a SALIDA, respetando en ambos casos las convenciones de unificación de PROLOG.
Si no se especifica, se asume que la SALIDA será idéntica a la ENTRADA y que podrá contener sólo una palabra. La variable para referenciar en la expresión prolog estará formada por el nombre de el objeto con todas las letras en mayúscula y los espacios reemplazados por guiones. En caso de especificarse solamente la ENTRADA se asumirá que la salida es idéntica. Debe tenerse en cuenta que mientras las entrada puede corresponder a una o más palabras, la salida debe representar una única estructura.
EXPRESION_PROLOG: permite expresar las condiciones de validación y conversión de la ENTRADA y SALIDA.
Para aumentar la legibilidad del programa traducido por el MT, en todos los casos que haya que generar nombres internos se utiliza el nombre del objeto. Si este tiene espacios son reemplazados por "_".

A continuación se ilustra el uso de esta plantilla mediante algunos ejemplos:

```
MD<-      un barco.
MD<-      una casa.
    (no hay restricciones para barco y casa)
MD<-      una cancion :- publicada(CANCION).
    (al no especificarse ENTRADA<-SALIDA se asume que se
    utilizará CANCION para referenciarlas. Se supone que el
    usuario define además el predicado "publicada" que
    establece las opciones válidas para CANCION)
MD<-      un color :- member(COLOR,[rojo,azul,verde]).
    (las constantes aceptables para color serán rojo,azul y
    verde)
```

```
MD<-      un número (N) :- num(N).
     {se utiliza como nombre de variable la letra N, y se
     utiliza la primitiva "num" para validación. No se realiza
     conversión alguna}
MD<-      un hombre ([X|XS] <- X|XS).
     {si se ingresa por ejemplo "Juan Ramón Perez" será
     convertido a la estructura [Juan,Ramón,Perez]}
MD<-      un sistema automatico.
     {observar que el nombre del objeto puede estar formado por
     más de una palabra, no siendo necesario utilizar "guiones"
     o algún otro mecanismo para que simular esta
     característica.
MD<-      una velocidad maxima (vel(X,Y,Z) <- X Y por Z) :-
               num(X),longitud(Y),tiempo(Z).
     {si se ingresa "10 km por hora" sera verificado que
     correspondan a unidades de longitud y tiempo. El formato de
     salida de esta información será "vel(10,km,hora)." }
MD<-      un tamaño (W <- X Y) :-
            num(X),longitud(Y),convertir_a_metros(X,Y,W).
     {si se ingresa "10 metros" será verificado y luego será
     convertido a "10". Si se ingresa "10 km" será convertido a
     "10000"}
```

La posibilidad de convertir la ENTRADA al formato deseado de SALIDA, permite independizar los procedimientos de la interfase de LN diseñada. La utilidad es impresionante, permite por ejemplo, escribir procedimientos que se independicen de las unidades de medida: al definir un proceso que dibuje figuras geométricas de un tamaño indicado se lo realiza en función de un número que corresponde a la cantidad de puntos de la pantalla. Luego se especifica la equivalencia de la unidad de medida en la interface, junto con el conocimiento de conversión de unidades de medida. La interface se encarga entonces de la conversión, en forma transparente para el procedimiento. Si se establece que 1 mm corresponde a 2 puntos, "dibujar un rectángulo de ancho 10 cm y alto 20 mm" será convertido adecuadamente, indicando que el ancho es 200 y el alto 40, inclusive, gracias a la posibilidad de generar código dependiente de los datos podrían especificarse factores de conversión distintos para el ancho y el alto.

La declaración de los objetos es utilizada para realizar el control semántico y determinar los "no-terminales" de la gramática semántica. Mediante esta última se especifican las relaciones que se establecen en el dominio.

Organización jerárquica:

Existe actualmente la convicción de que los sistemas orientados a objetos son concebidos e implementados de forma más próxima a como la mente humana percibe los sistemas de la realidad: como una interacción entre objetos distintos, cada cual poseyendo propiedades y comportamientos propios, y donde cada objeto puede tener otros objetos como componentes de su estructura [JON/87]. Por esta razón la interfase provee mecanismos de alto nivel para manipular objetos estructurados: los objetos declarados se pueden organizar jerárquicamente, estableciendo relaciones entre clases y sub-clases.

Esta información determina la posibilidad de representar generalizaciones y especializaciones al poder reunir en clases, los objetos con propiedades similares.
La plantilla para declarar objetos se extiende entonces a:

ART_INDEF OBJETO (SAL<-ENT) es un CLASE:-EXPRESION_PROLOG

Donde,
CLASE: corresponde a la clase "primitiva" denominada "objeto", o a una clase definida por el usuario. Todo objeto declarado especificando la relación "es un" se puede utilizar como clase. Cuando sea necesario distinguirlos en el texto, referenciaremos a estos últimos como "objeto-clase" y a los otros como "objeto-ordinario". Por ejemplo:

MD<- una fragata es un barco.
MD<- un nodo es un objeto.
MD<- un arco es un objeto.
MD<- una entidad es un nodo.
MD<- un archivo y un proceso son entidades.
(el término "es" puede reemplazarse por "son" para poder utilizar en forma natural la facilidad de elipsis del sistema. Notar además que pueden usarse los plurales de los términos declarados)
MD<- un flujo es un arco.

Estas declaraciones permiten que el MT, al esperar un objeto de determinado "tipo", acepte también los "sub-tipos" correspondientes. Para el ejemplo anterior, en las declaraciones donde se utilice "entidad" también son aceptados objetos de tipo "archivo" y "proceso".

Al referenciar un objeto, puede precederse en forma opcional por el nombre de su clase (Ej: "Sarmiento" o "la fragata Sarmiento"; "facturación" o "el proceso de facturación"), este nombre es utilizado por el MT para el control semántico y luego es descartado, para poder independizarse de la forma en que se ingresen los datos.
Es posible parametrizar al MG para que produzca la conversión a lenguaje natural incluyendo, o no, el nombre de la clase a que pertenecen los objetos. También puede parametrizarse el MT para que genere código que realice control de tipos en tiempo de ejecución, en forma similar a [DAH/81].

Objetos estructurados:

Como se expresó anteriormente, los objetos pueden tener otros objetos como componentes de su estructura, permitiendo una gran flexibilidad para organizar y representar los objetos del dominio y las relaciones entre ellos. Se implementó un mecanismo para representación del conocimiento mediante objetos estructurados similar al implementado mediante "frames" pero sin posibilidad de redefinición de información por los problemas de "no-monotonia" que trae aparejados. Es decir que si se estableció un valor "default" para un atributo no puede ser declarado distinto para alguna instancia de su clase.

Nuevamente generalizamos la plantilla anterior para declarar la estructura de un objeto-clase:

ART_IND OBJETO (SAL<-ENT) es un CLASE con ESTRUCTURA:
EXPRESION_PROLOG

donde,
ESTRUCTURA: corresponde a una declaracion de atributos permitidos para el OBJETO. Estos atributos deben estar a su vez declarados como objetos. En cada atributo es posible especificar en forma opcional, encerrandolos entre paréntesis, un valor "default" o un mecanismo de obtención del valor "if-needed".

Aunque en la plantilla se especifica la constante "con" para reconocer el inicio de la estructura, el MT reconoce, en forma indistinta, los términos "con", "de", "tiene" "que tiene", "cuyo", "cuya". Esta equivalencia puede alterarse para agregar o eliminar términos.

A continuación se muestran algunos ejemplos de declaración y utilización de objetos estructurados. Al utilizarlos, es importante observar que se debe nombrar el atributo y a continuación su valor. No es necesario respetar ni el orden ni la cantidad de los atributos.

```
MD<-    un medio de transporte es un objeto
        con velocidad máxima.
    {permite aceptar frases como "el vehiculo que tiene
    velocidad máxima 200 km por hora"}
MD<-    un auto es un medio de transporte
        con dueño y color y puertas y capacidad(4 personas).
    {se asume que la capacidad típica de un auto es 4 personas}
MD<-    un hombre es un objeto.
MD<-    un dueño es un hombre.
MD<-    una capacidad (X) :- num(X), X >= 0.
MD<-    una capacidad (X <- X personas) :- num(X), X >= 1.
    {la definición de capacidad muestra que es posible
    establecer más de una construcción para un objeto, que
    permite, en este caso, utilizar un número, o un número
    seguido de la palabra personas}
MD<-    una puerta es un objeto
MD<-        con modelo y cantidad.
MD<-    un modelo :- member(MODELO,[comun,especial]).
MD<-    una cantidad (X) :- num(X), X > 0.
```

Observar que "puerta" está utilizado como atributo dentro de "auto", y que "auto" es un "medio de transporte" que tiene como atributo "velocidad máxima". Por lo tanto es posible aceptar frases como "el auto de color rojo, dueño Jorge, puertas modelo especial y velocidad máxima 250 km por hora". La capacidad se asume de 4 personas. Al no especificarse la cantidad de puertas, no es posible determinarla de manera alguna. Notar que para declarar esta información no es válido escribir "cantidad de puertas 4 modelo especial" pues siempre debe especificarse primero el nombre del atributo. La manera "adecuada" sería, por ejemplo, "puertas modelo especial cuya cantidad es 4".

```
MD<-    un colectivo es un medio de transporte
MD<-        con capacidad, peso(mult(CAPACIDAD,90,PESO)).
    {La expresión entre paréntesis determina que en caso de
    requerirse el peso hay que evaluar la expresion Prolog que
    se encuentra a continuación de la barra, el resultado
    estará en la variable "PESO"}
```

Utilizando estas declaraciones el MT crea además funciones de acceso a los atributos de un objeto, que permiten frases como "la capacidad del colectivo", "el color de el auto", etc. Todas estas funciones son de la forma:

ART_DEFINIDO ATRIBUTO de ART_DEFINIDO OBJETO.

El comportamiento del MT es diferente para los objetos ordinarios y las clases:

- Objeto-Ordinario: se procede a realizar la validación correspondiente y se genera una variable para referenciarlo.
- Objeto-Clase precedido de artículo indefinido: si está en el consecuente de una regla se lo trata como un objeto ordinario, pero si está en el antecedente o en una consulta, se crea una nueva instancia del objeto.
- Objeto-Clase precedido de artículo definido: si existe una referencia a él en el contexto, se lo trata como un objeto ordinario, en caso contrario se genera el procedimiento para extraer el objeto de la base de datos del sistema.

Los siguientes ejemplos ilustran mejor estas diferencias:

```
MT<-      un auto de color verde, dueño alberto.
MT->      es_un(o1,auto). color(auto,o1,verde).
          dueño(auto,o1,alberto).
MT<-      jorge mira un auto de color rojo.
MT->      es_un(o2,auto). color(auto,o2,rojo). mira(jorge,o2).
     {se crea un auto de color rojo}.
MT<-      jorge mira el auto de color verde.
MT->      es_un(X,auto),color(auto,X,verde),mira(jorge,X).
     {se halla el auto de color verde que mira jorge}
MT<-      Quien es el dueño?
     {elipsis ¿Quien es el dueño de el auto de color verde?}
MT->      es_un(X,auto),color(auto,X,verde),dueño(X,Y).
MT<-      dibujar un auto de color rojo y uno de color azul y
          movilizar el de color rojo.
MT->      es_un(o3,auto). color(auto,o3,rojo).
          es_un(o4,auto). color(auto,o4,azul).
          dibujar(o3),dibujar(o4),movilizar(o3).
     {la referencia al auto de color rojo corresponde al ultimo
     nombrado}
MT->      dibujar el objeto de color rojo.
MT<-      es_un(X,Y),color(Y,X,rojo),dibujar(X).
     {objeto referencia a cualquier objeto}
```

El MT acepta también que el identificador de los objetos sea establecido por el usuario mediante la plantilla:

IDENTIFICADOR es ART_INDEFINIDO OBJETO

Observar sin embargo, que esto no define nuevas clases, solamente corresponden a identificadores para instancias del programa:

```
MT<-      margarita es un barco.
MT->      es_un(margarita,barco).
MT<-      mario, alberto martinez y raul son hombres.
MT->      es_un([mario],hombre).
          es_un([alberto,martinez],hombre).
          es_un([raul],hombre).
```

```
MT<-      k27 es un sistema automatico.
MT->      es_un(k27,sistema_automatico).
```

9.2 Definición de Terminología:

Una vez declarados los objetos del dominio, es necesario declarar la terminología del dominio, compuesta por las relaciones y funciones aplicables a los mismos. El sistema realiza una traducción automática a PROLOG de estas relaciones y funciones, sin embargo, el usuario puede especificar su propia traducción.

La plantilla general corresponde al siguiente esquema:

FRASE (RESULTADO) / TRADUCCION:-EXPRESION_PROLOG.

donde,
FRASE: indica la forma esperada de la frase. En la misma se deben utilizar las referencias a los objetos declarados anteriormente.
RESULTADO: en caso de indicarse, corresponde al nombre de la variable donde se devuelve el resultado de la evaluación de la TRADUCCION. Notar que la FRASE actúa como una función que retorna un valor.
TRADUCCION: refiere al formato interno al que debe ser traducida la FRASE. Esta traducción es opcional, y puede ser generada automáticamente por el sistema con las siguientes convenciones: el nombre del predicado estará formado por las constantes de la frase, las variables corresponderán a los objetos de la frase en el orden en que aparecen.
EXPRESION_PROLOG: expresa las condiciones en las que se debe efectuar la traducción.

Las siguientes definiciones (MD) permiten que el MT traduzca adecuadamente frases como la siguiente:

```
MT<-      un proceso consulta a un archivo <-
          un flujo esta destinado al proceso y
          esta originado por el archivo.
MT->      consulta_a(PROCESO,ARCHIVO):-
          destinado(FLUJO,PROCESO),originado(FLUJO,ARCHIVO).

MD<-      un flujo esta destinado a una entidad /
          destinado(FLUJO,ENTIDAD).
MD<-      un flujo esta originado por una entidad /
          originado(FLUJO,ENTIDAD).
MD<-      un proceso consulta a un archivo.
     (la traducción generada es "consulta_a(PROCESO,ARCHIVO)")
```

En este ejemplo se puede observar el uso de elipsis en la palabra flujo y la utilización de articulos definidos para referenciar a proceso y archivo. Como se explicó, los objetos declarados son reconocidos como variables, generándose nombres adecuados. El mecanismo de validación semántica reconoce que proceso y archivo son entidades.

En el siguiente ejemplo se puede observar un uso más complejo de elipsis:

```
MT<-    Un bloque de color verde esta sobre una mesa y uno
        azul esta sobre el verde y uno rojo sobre el azul.
MD<-    un bloque es un objeto con color.
MD<-    un apoyo es un objeto.
MD<-    un bloque es un apoyo.
MD<-    un bloque esta sobre un apoyo.

MT->    es_un(o5,bloque).   color(o5,verde).   es_un(o6,mesa).
        esta_sobre(o5,o6).
        es_un(o7,bloque).   color(o7,azul).
        esta_sobre(o7,o5).
        es_un(o8,bloque).   color(o7,rojo).
        esta_sobre(o8,o7).
```

A continuación se muestra una interfase para un sistema experto en diagnóstico médico tipo MYCIN [DAV/82].

```
MD<-    un organismo.
MD<-    un lugar.
MD<-    un tipo de lugar.
MD<-    un tipo de lugar (X <- normalmente X).
MD<-    un tinte gram :-member(TINTE_GRAM,[positivo,negativo]).
MD<-    una forma (coco <- esferica).
MD<-    una forma (bacilo <- alargada).
MD<-    una evidencia (5 <- fuertemente sugestiva).
MD<-    una evidencia (3 <- sugestiva).
MD<-    una evidencia (1 <- debilmente sugestiva).
MD<-    una clase.
MD<-    un organismo proviene de un lugar /
            lugar(ORGANISMO,LUGAR).
MD<-    un organismo proviene de un lugar un tipo de lugar/
            lugar(ORGANISMO,LUGAR),tipo(LUGAR,TIPO_DE_LUGAR).
MD<-    un organismo tiene un tinte gram /
            tinte_gram(ORGANISMO,TINTE_GRAM).
MD<-    un organismo tiene una forma / forma(ORGANISMO,FORMA).
MD<-    hay una evidencia de que un organismo es de una clase /
            evidencia(clase(ORGANISMO,CLASE),EVIDENCIA).
```

que permite expresar en forma "clara", reglas de gran complejidad:

```
MT<-    Hay una evidencia fuertemente sugestiva de que un
        organismo es de la clase neisseria <- proviene de un
        lugar normalmente esteril y tiene un tinte gram
        negativo y forma esferica.
MT->    evidencia(clase(ORGANISMO,neisseria),5):-
        lugar(ORGANISMO,LUGAR),tipo(LUGAR,esteril),
        tinte_gram(ORGANISMO,negativo),forma(ORGANISMO,coco).
MT<-    Hay una evidencia sugestiva de que un organismo es
        contaminante <- proviene de la traquea y tiene forma
        alargada.
MT->    evidencia(clase(ORGANISMO,contaminante),3):-
        lugar(ORGANISMO,traque),forma(ORGANISMO,bacilo).
```

Un sistema de consulta y actualización de una base de datos sobre personal podría especificarse mediante las siguiente gramática:

```
MD<-      un empleado ([X|Xs] <- X|XS) es un objeto con
          sección y sueldo.
MD<-      un sueldo (Z <- X Y) :- num(X),moneda(X,Y,Z).
MD<-      moneda(X,australes,X).
MD<-      moneda(X,W,Y) :- cambio(W,Z), Y is X * Z.
MD<-      una sección.
MD<-      un empleado gana un importe/sueldo(EMPLEADO,IMPORTE).
MD<-      un empleado cobra un importe/sueldo(EMPLEADO,IMPORTE).
     (también podría haberse escrito las dos últimas
     definiciones mediante: un empleado X un importe /
     sueldo(EMPLEADO,IMPORTE) :- X = gana ; X = cobra.)
MD<-      Cuanto gana un empleado?/
          sueldo(EMPLEADO,SDO),
          write(EMPLEADO,gana,SDO,australes).
MD<-      imprimir un objeto / write(OBJETO).
MD<-      el sueldo de un empleado (un sueldo) /
          sueldo(EMPLEADO,SUELDO).
```

Esta gramática permite consultar y actualizar la base de datos en forma interactiva, como se muestra a continuación:

```
MT<-      Jorge Perez es un empleado cuya sección es pintura y
          sueldo 100 australes.
     (se utilizó "cuya" para la estructura y se "agregó" la
     palabra "es" para aumentar la legibilidad (recordar que
     esto es permitido))
MT->      es_un([Jorge,Perez],empleado).
     seccion(empleado,[Jorge,Perez],pintura).
     sueldo(empleado,[Jorge,Perez],100).
MT<-      Raul Gonzalez es un empleado de la sección
          calibración.
     (se utilizó "de" para la estructura y se "agregó" la
     palabra "la" para aumentar la legibilidad)
MT->      es_un([Raul,Gonzalez],empleado).
     seccion(empleado,[Raul,Gonzalez],calibracion).
MT<-      Raul Gonzalez cobra 200 dolares.
MT->      sueldo(empleado,[Raul,Gonzalez],1200).
     (observar que en la definición de "cobra" se utilizó una
     traducción compatible con la generada por el MT para los
     objeto-clase)
MT<-      ¿Cuanto gana el empleado de la sección pintura?.
MT->      es-un(X,empleado),sección(empleado,X,pintura),
          gana(empleado,X,Y),write(X,gana,Y,australes).
  ->      Jorge Perez gana 100 australes.
MT<-      imprimir el sueldo de Jorge Perez.
MT->      sueldo(empleado,[Jorge,Perez],X),write(X).
  ->      100.
```

En el próximo ejemplo se puede observar que la generación de código en función de los datos es sencilla de realizar, en el mismo se define la respuesta que debe dar cada animal a el saludo "hola":

```
MD<-  un perro :- es_un(PERRO,perro).
MD<-  un pato :- es_un(PATO,pato).
MD<-  Hola un perro () / ladrar(PERRO).
MD<-  Hola un pato () / graznar(PATO).
MD<-  ladrar(X) :- write(guau!).
MD<-  graznar(X) :- write(cua-cua!).
```

```
MT<-  Mendieta es un perro.
MT->  es-un(Mendieta,perro).
MT<-  Donald es un pato.
MT->  es-un(Donald,pato).
MT<-  Hola Mendieta.
MT->  guau!
MT<-  Hdla Donald.
MT->  cua-cua!
```

Una de las características más atrayentes de la utilización de funciones, es que no es necesario la introducción de variables intermedias como pasa en PROLOG. Por ejemplo:

```
MD<-    un hombre es el tio de un hombre_2 /
            tio(HOMBRE,HOMBRE_2).
MD<-    un hombre es el hermano de un hombre_2. /
            hermano(HOMBRE,HOMBRE_2).
MD<-    el progenitor de un hombre (un hombre-2) /
            progenitor(HOMBRE-2,HOMBRE).
MT<-    un hombre es el tio de un hombre_2 <-
            es el hermano del progenitor del hombre_2.
        (observar utilización de elipsis y de la función
        esposo).
MT->    tio(HOMBRE_1,HOMBRE_2):-
            hermano(HOMBRE_1,X),progenitor(X,HOMBRE_2).
```

En muchas oportunidades, existen predicados que pueden usarse como relaciones y como funciones. En el caso anterior se podría haber definido también la relacion "un hombre es el progenitor de un hombre_2" o la función "tio". Para estos casos se provee la siguiente facilidad: Si el nombre de la variable a devolver corresponde a uno de los objetos utilizados en la frase, se define automaticamente además de la relación, una función cuyo nombre corresponde a las constantes de la relación a partir del verbo ser o estar:

```
MD<-    un hombre es el progenitor de un hombre_2 (HOMBRE).
        (define la relación "es_el_progenitor_de" y la función
        "el progenitor de").
MD<-    un hombre es el hermano de un hombre_2 (HOMBRE).
MD<-    un hombre es el tio de un hombre_2 (HOMBRE).
MD<-    un hombre es el tio abuelo de un hombre_2.
MT<-    un hombre es el tio de un hombre_2 <-
            es el hermano del progenitor del hombre_2.
MT->    es_el_tio_de(HOMBRE_1,HOMBRE_2):-
            es_el_hermano_de(HOMBRE_1,X),
            es_el_progenitor_de(X,HOMBRE_2).
MT<-    un hombre es el tio abuelo de un hombre_2 <-
            es el tio del progenitor del hombre_2.
MT->    es_el_tio_abuelo_de(HOMBRE,HOMBRE_2):-
            es_el_tio_de(HOMBRE_1,X),
            es_el_progenitor_de(X,HOMBRE_2).
MT<-    un hombre es el tio abuelo de un hombre_2 <-
          es el hermano del progenitor del progenitor del
          hombre-2
MT->    es_el_tio_abuelo_de(HOMBRE,HOMBRE_2):-
          es_el_tio_de(HOMBRE_1,X),es_el_progenitor_de(X,Y),
          es_el_progenitor_de(Y,HOMBRE_2).
```

(Se puede ver que las funciones pueden anidarse varias veces sin necesidad de introducir variables intermedias)

También podrían ser definidas funciones matemáticas, pero en la versión actual no hay forma de declarar precedencia de operadores ni utilizar paréntesis para indicarla. Sin embargo si esto no es problema puede utilizarse la interfase para manipularlas:

```
MD<-    el factorial de un numero es un numero_2 /
            fact(NUMERO,NUMERO_2).
MD<-    un numero_1 * un numero_2 (un numero) /
            mult(NUMERO_1,NUMERO_2,NUMERO).
MD<-    un numero_1 - un numero_2 (Z) /
            sum(NUMERO,NUMERO_2,NUMERO_1).

MT<-    el factorial de #X es #X * el factorial de #X - 1.
MT->    fact(X,Y):-sum(Z,1,X),fact(Z,W),mult(X,W,Y).
```

Los objetos y la terminología del dominio de aplicación constituye una fuente importante de conocimiento que es utilizada para entender las intenciones del usuario, como se puede ver en el siguiente ejemplo:

```
MT<-    enviar cartas a los clientes atrasados en los pagos.
MT->    atrasado(X),enviar_carta(X).
MD<-    un cliente.
MD<-    un periodo (X <- X dias).
MD<-    un periodo (X <- Y meses) :- mult(30,X,Y).
MD<-    un periodo es una cantidad.
MD<-    enviar carta a un cliente / enviar_carta(CLIENTE).
MD<-    un cliente esta atrasado en los pagos (CLIENTE) /
            atrasado(CLIENTE).
MD<-    la antigüedad de la deuda de un cliente
            es un periodo (PERIODO)
antigüedad(CLIENTE,PERIODO).
MD<-    una cantidad es mayor a una cantidad_2 /
            >(CANTIDAD,CANTIDAD_2).
MT<-    un cliente esta atrasado en los pagos <-
            la antigüedad de la deuda es mayor a 2 meses.
MT->    atrasado(CLIENTE) :-
            antigüedad(CLIENTE,PERIODO),mayor(PERIODO,60).
```

En el ámbito educativo se podría lograr una rápida inserción debido a la naturalidad de programación. El lenguaje LOGO [PAP/80], ampliamente difundido en las escuelas, no es más que una adaptación del lenguaje LISP [WIN/81] a fin de lograr una sintaxis mas sencilla. Sin embargo, mientras que mediante la interfase es posible escribir "dibujar un rectángulo de ancho 5 cm y alto 18 mm" en LOGO hay que hacerlo mediante la criptográfica instrucción RECTANGULO 18 50. La utilización de la ILN permitiría además utilizar un lenguaje capaz de realizar inferencias y razonar sobre los objetos que manipula! [KOW/84].

9. Derivación del MT y el MG:

Debido a la extensión del trabajo, no se describe como se deriva automáticamente el MT y el MG a partir de la información contenida en la gramática. El mismo puede ser consultado en el informe respectivo en la oficina del Programa Nacional de Informática y Electrónica.

11. Limitaciones y futuras extensiones:

Es por todos conocido los beneficios de contar con potentes diagnósticos y mecanismos de recuperación de errores, sin embargo, la mayoría de los sistemas de LN existentes sólo cuentan con un rudimentario diagnóstico de errores. Al contemplar la posibilidad de implementar estas características en nuestro sistema se presentaron graves problemas: al utilizar una estrategia de "parsing" en paralelo era posible diagnosticar "inteligentemente" los errores de las frases y proveer recuperación de los mismos, pero no era posible extenderlo para implementar elipsis, concordancia y generación de código en función de los datos; al utilizar una estrategia de análisis en profundidad se lograron las características exhibidas en el trabajo pero el diagnóstico se limita a aceptar o rechazar las frases del usuario. Se optó por una interfase más general, sin embargo, esta limitación merece ser estudiada con mayor dedicación.

El sistema fue diseñado con el objetivo principal de producir una interfase sencilla de diseñar aunque fuera en detrimento de la potencia y generalidad de la interfase. Sin embargo se puede observar las siguientes limitaciones de la versión actual que podrían ser corregidas en versiones posteriores:

- Manejo adecuado de conjugación de verbos y concordancia de género y número, mediante reglas generales y un diccionario de términos más usados.
- El mecanismo implementado para la representación de objetos estructurados es muy general, pero no es eficiente. Debería poder parametrizarse distintas formas de representación.
- La utilización de los atributos de los objetos no es del todo natural. Una solución consistiría en declarar el formato esperado para cada atributo.
- Solo se maneja coordinación de conjunciones, es necesario extenderla para manejar disyunciones.

12. Conclusiones:

El presente trabajo describe un sistema para diseñar ILN realizado con un enfoque no tradicional: en vez de buscar potencia y generalidad se ha puesto énfasis en la sencillez de empleo, la rapidez de adaptación a diferentes dominios y el no requerir conocimientos linguísticos para ser definida.

Las gramáticas semánticas resultan de suma utilidad, por su sencillez y potencia para capturar la terminología del dominio: una versión simplificada de esta interfase se utilizó en el proyecto micro-Ethos en una herramienta de desarrollo de software. La misma permite comunicarse con el usuario utilizando la terminología del dominio de aplicación y facilitó el procedimiento de formalización del conocimiento proveyendo una metodología para la captura del conocimiento [KVI/88].

Teniendo en cuenta la simplicidad del sistema, la variedad de frases aceptadas es importante, exhibiendo características avanzadas como especificación declarativa de algoritmos, validación semántica, elipsis y concordancia, manejo de objetos estructurados, capacidad de realizar inferencias, expresión natural de relaciones y funciones, utilización de información implícita, etc.

13. Bibliografia:

[ABE/85] Abelson H,Sussman G. Structure and Interpretation of Computer Programs. MIT Press.

[BAL/75] Balzer R. A Global View of Automatic Programming. Proc.3rd IJCAI

[BAL/77] Balzer R. Goldman N. Wile D. Informality in program specification. Proc.5th IJCAI,pp 389-397.

[BER/68] Bertram Raphael. SIR, Semantic Information Processing

[BAR/81] Barr A, Cohen A, Feigenbaum E, Eds. The Handbook of Artificial Intelligence. M.Kaufmann.

[BIE/76] Biermann A. Aproaches to Automatic Programming. Advances in Computers, Volume 15. Academic Press, NY.

[BIE/80] Biermann A, Ballard B. Toward Natural Language Computation. AJCL volume 6-2.

[BOL/81] Bolc L. Natural Language comunication with computers.

[BON/79] Bonnet A. Strategies for Understanding Structured English. Memo HPP-79-25.Stanford University.

[BOW/75] Bobrow D, Collins A. Ed. Representation and Understanding. Academic Press, New York.

[CLA/82] Clark K,McCabe G. Programming in Logic. Prentice Hall.

[CHO/68] Chomsky N. On the notion of 'Rule of Grammar.

[CLO/81] Cloksin W,Mellish C. Programming in Prolog. Springer.

[DAH/81] Dahl V. Translating Spanish into logic through logic. AJCL, vol 13, pp 149-164.

[DAH/86] Dahl V.,McCord M. Treating Coordination in Logic Grammars. AJCL.

[DAV/82] Davis R. Teiresias: Applications of Meta-Level Knowledge in Knowledege-Bases Systems in Artificial Intelligence. McGraw-Hill, NY.

[FON/85] Fong S, Berwick R. New Approaches to Parsing Conjunctions Using Prolog.

[GRE/77] Green C. A Summary of the PSI Program Synthesis System. Proc.5th IJCAI.

[HEN/76] Hendrix G. The LIFER manual:a guide to building practical NL interfaces.TR 138.Menlo park

[HEI/76] Heidorn G. NLPQ proyect: Natural Language for Programing Queuing Simulation. Handbook of Artificial Intelligence, Academic Press.

[JON/87] Jonathan Miguel. Orientacao para Objetos em SMALLTALK. I Simposio Brasileiro de Engenharia de Software. RJ

[KOW/79] Kowalski R., Logic for Problem Solving. Elsevier.

[KOW/84] Kowalski R. Logic as a Computer Language for Children.

[KVI/84] Kvitca A. En colaboración. Un lenguaje de consultas a bases de datos con interfase de Lenguaje Natural.JAIIO/84, Argentina.

[KVI/84] Kvitca A. Bases de conocimiento. Una propuesta de unificacion. Anales de JAIIO/84, Argentina.

[KVI/88] Kvitca A. En colaboración: Proyecto ETHOS. Apendice Informe uETHOS. FABI. Argentina.

[MAN/74] Manna Z. Mathematycal Theory of Computation. McGraw Hill, NY.

[MEL/85] Mellish C. Computer Interpretation of Natural Language Descriptions. Ellis Horwood.

[MIN/75] Minsky M. A framework for Representing Knowledge. The Psychology of Computer Vision. McGraw Hill, NY.

[PAP/80] Papert S. Desafio a la Mente, Ed.Galapagos, Argentina

[PER/80] Pereira F,Warren D. Definite clause grammars for language analysys — a survey of the formalism and a comparison with transition networks. Artificial Intelligence, vol 13. pp 231-278.

[RIC/83] Rich E. Artificial Intelligence. McGraw Hill
[ROB/80] Robinson J. DIAGRAM: A Grammar for Dialogues. TR 205, SRI International, Menlo Park.
[SCH/86] Schor Marshall. Declarative Knowledge Programming: Better than Procedural?. IEEE EXPERT, Spring 1986.
[STE/86] Sterling L, Shapiro E. The Art of Prolog. MIT Press.
[SIM/84] Simmons R. Computations from the English. Prentice Hall. New York.
[SIM/86] Simmons R. Man-Machine Interfaces: Can They Guess What You Want?. IEEE EXPERT. Spring 1986.
[VAN/80] Van Baalen J. Using Natural Language to Solve Recursive Programming Problems.
[YEO/84] Yeong Yu. Translating Horn Clauses from English.
[WIN/72] Winograd T. Understanding Natural Language. Ac Press.
[WIN/75] Winograd T. Frame Representations and the Declarative Procedural Controversy,in Representations and Understanding. Academic Press, New York.
[WIN/81] Winston P. Horn B. LISP. Addison-Wesley.
[WOO/70] Woods W. Transition Network Grammars for Natural Language Analysis. Comm. ACM. volume 13-10.pp341-398.
[WOO/77] Woods W. A personal View of Natural Language Understanding", SIGART Newsleter, 2/77,pp 17-20.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

GRAMÁTICAS DE DETERMINAÇÃO: UMA PROPOSTA METODOLÓGICA

CLARISSE SIECKENIUS DE SOUZA

EMPRESA BRASILEIRA DE TELECOMUNICAÇÕES
AVENIDA PRESIDENTE VARGAS 1012
RIO DE JANEIRO - RJ / BRASIL

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DEP. DE INFORMÁTICA
R. MARQUES DE SÃO VICENTE 225
RIO DE JANEIRO - RJ / BRASIL

RESUMO: O desenvolvimento de interfaces em linguagem natural para bancos de dados requer que se desenvolvam metodologias capazes de proporcionar ao usuário um máximo de flexibilidade na maneira de expressar-se, juntamente com toda a precisão esperada para a recuperação de informações em sistemas desta natureza. As GRAMÁTICAS DE DETERMINAÇÃO (GD's) apresentam uma proposta metodológica para o desenvolvimento da base de conhecimentos linguísticos em interfaces "naturais" para bases de dados.

1 - INTRODUÇÃO

O processamento de linguagem natural (PLN) comporta pesquisas com as mais diversas orientações, dentre as quais, para fins deste trabalho, interessa destacar três. São elas: (a) a investigação acerca do mecanismo processador da linguagem natural (de interesse eminentemente psicolingüístico); (b) a investigação acerca dos formalismos destinados a representar o conhecimento inconsciente que todo falante tem sobre a gramática de sua língua (de interesse eminentemente sintático); e (c) a investigação acerca da maneira mais eficiente de se conjugarem os dois temas anteriores, visando o desenvolvimento de bases de conhecimentos lingüísticos em sistemas destinados à compreensão/produção automática de textos em linguagem natural.

Existe uma afinidade entre a pesquisa do tipo (a) e a psicologia, entre a pesquisa do tipo (b) e a matemática e, finalmente, entre a pesquisa do tipo (c) e o que Coelho [1981] chamou de engenharia da linguagem, embora se imprima aqui alguma modificação. O termo engenharia, aqui, caracteriza a ênfase na construção de sistemas e de todos os aspectos envolvidos nesta atividade. O presente trabalho trata especificamente desta questão: é uma proposta de metodologia para o desenvolvimento de sistemas processadores de língua portuguesa, na aplicação de consultas a bases de dados.

Embora haja diversas propostas com objetivo semelhante, a metodologia das GRAMÁTICAS DE DETERMINAÇÃO (GD's) apresenta a particularidade de ter sido desenvolvida sob a ótica da lingüística aplicada, não no sentido de testar no campo computacional

hipóteses produzidas no âmbito da teoria lingüística, mas antes no sentido de utilizar insumos desta teoria para produzir uma alternativa metodológica **sem comprometimento teórico a priori**, motivada apenas pelas soluções exigidas pela aplicação de consulta a Bases de Dados (BD's).

O desenvolvimento completo da proposta relativa às BD's encontra-se em Souza [1988] e as informações acerca do primeiro sistema em operação elaborado segundo esta metodologia podem ser encontradas em Bicharra,Carvalho & Fortuny [1986] ou Souza [1987]. Tal sistema - denominado FARAO - esteve disponível na EMBRATEL (empresa Brasileira de Telecomunicações), por um período superior a doze meses, para usuários de uma base de dados financeira.

2 - A CONSULTA A BASES DE DADOS

A consulta a bases de dados, em especial no momento de transição entre a utilização exclusiva de linguagens artificiais ou telas-menu e a possibilidade de expressão em uma língua natural, apresenta algumas características que devem ser levadas em conta no desenvolvimento de uma interface. Destacam-se (a) a reminiscência de formalismos próprios das consultas formuladas na **query language** anteriormente utilizada na aplicação, (b) a influência da estrutura da base de dados sobre a estrutura da consulta (marcando a recuperação de dados, e não tanto a recuperação de informações) e (c) o estilo "telegráfico", ou abreviado, revelador de uma sintaxe para-gramatical da língua. Exemplos de cada um destes fenômenos podem ser encontrados em Souza [1987,1988].

O desenvolvimento de interfaces em linguagem natural que não contemple as características expressivas dos usuários pode vir a ser tributado com um alto índice de rejeição de consultas formuladas, devido a uma excessiva rigidez gramatical na análise sintática. Trabalhos como o de Waltz [1978] ou o de Burton [1976] já davam conta do fenômeno. No primeiro caso, o autor propunha que, mediante fracasso na análise sintática, o processador tentasse recuperar o sentido da consulta através de uma análise semântica; no segundo, a proposta de análise pautava-se essencialmente em critérios semânticos, desde o primeiro passo, formulando-se uma gramática semântica. Nesta linha encontra-se também o trabalho de Hendrix e outros [1978].

O invariante dos três trabalhos citados, e de outros menos diretamente relacionados à aplicação de consulta a BD, é priorizar a informação semântica, mesmo em estágio de análise (teoricamente) sintática. Sob esta influência, as GRAMÁTICAS DE DETERMINAÇÃO funcionam com base em categorias sintáticas **semanticamente motivadas**. A denominação das BD's é inspirada na relação de **determinação** (subordinação) que se pode encontrar em constituintes complexos das gramáticas das línguas humanas, nos quais um elemento é o núcleo (**determinado**) do constituinte, enquanto outro(s) funciona(m) como modificador(es) (**determinante(s)**).

Para melhor esclarecer os pontos de vista defendidos pela proposta metodológica que aqui se apresenta, vejam-se os exemplos abaixo, onde se utiliza a linguagem de consulta **ADASCRIPT**, durante bastante tempo empregada no gerenciador **ADABAS**, e as formulações correspondentes à mesma consulta, em português.

(a) CONSULTA EM ADASCRIPT

```
FIND 59 WITH C4 = 3101 OR 3102
 IF RE NOT EQUAL SU REJECT
ACCUM A1 A2 A3 A4 A5 A6
CONTROL C4 EX
SUMMARY DISPLAY RE C4 EX A1 A2 A3 A4 A5 A6
```

(RE -> REGIÃO; SU -> SUL; C4 -> CONTA; EX ->EXECUTANTE)

(b) POSSÍVEL CONSULTA CORRESPONDENTE EM PORTUGUÊS

Qual é o orçado de janeiro a junho dos executantes da região sul, nas contas de mão de obra e material de consumo?

(c) POSSÍVEL CONSULTA CORRESPONDENTE EM PORTUGUÊS

Para os executantes da região sul, com material de consumo e mão de obra, dê o valor orçado para os meses de janeiro a junho.

Entretanto, o fato de a entrada da consulta dever ser digitada, bem como a transição entre uma forma de expressão pautada em códigos e formalismos e uma expressão "natural", aponta para a importância de se processarem consultas equivalentes a (a), (b) e (c), acima, com os seguintes perfis:

(d) CONSULTA "TELEGRAFADA"

Executantes da região sul, orçado janeiro a junho, mão de obra e material de consumo.

(e) CONSULTA UTILIZANDO CÓDIGOS CONHECIDOS PARA O USUÁRIO

Executantes da SU, contas 3101 e 3102, orçado janeiro a junho

(f) CONSULTA "ARTIFICIAL" DE SINTAXE LIVRE

RE = SU, C4 = 3101 E 3102 ORÇADO JANEIRO A JUNHO
CONTROLE EX E C4

Embora (d), (e) e (f) não tenham nada de "natural", as condições de contorno da atividade de consulta a bases de dados poderiam favorecer o aparecimento deste tipo de enunciado, considerando-se os fatores ergonômicos (menor esforço na digitação) e os resquícios de uma tradição de consultas em linguagem artificial.

Observe-se que o invariante entre todos os enunciados (de (a) a (f)) são as "chaves" de seleção, recuperação e (opcionalmente) publicação de dados. Isto justifica a opção por se trabalhar com categorias sintáticas semanticamente motivadas. Na realidade, os exemplos dão mostras de "palavras-chave", ou "pontos cardeais" para a consulta. Estes são os elementeos de um padrão mínimo na sintaxe da mesma. Assim como, de maneira geral, as linguagens artificiais apresentam um padrão

C -> S R P

onde C é a consulta, S é o constituinte correspondente aos critérios de seleção, R o correspondente à recuperação e P o correspondente às instruções para a publicação de dados em tela ou impressora, também em linguagem "natural" (ou pseudo-natural, face às considerações acima) há um padrão mínimo invariante de consulta

B -> F P

onde B é um binômio (consulta) formado por uma parte F "fornecida" (critérios de seleção) e uma parte P "pedida" (solicitação de recuperação). Note-se que a publicação, nas consultas "naturais", é parte do bom senso: se não for especificada, deve apresentar a informação inferida a partir de F e P; caso contrário, obedece a uma sintaxe

B -> F P N

onde N é o constituinte que contém as informações relativas ao modo de publicação de dados.

Deve-se notar que o princípio da determinação fica bastante claro na sintaxe obtida. Um binômio B compõe-se de duas partes: uma parte determinada F e uma parte determinante P. Cada uma, por sua vez, possui uma parte determinada e uma parte determinante. No caso de F, o determinado são as regiões e/ou executantes e o determinante as contas. No caso de P, o determinado é o valor orçado ou realizado, enquanto o determinante são os meses. A atribuição de função determinada ou determinante aos elementos da base de dados emana da modelagem da base de dados e da modelagem do usuário (veja-se a este respeito Souza [1988]).

Por convenção das GD's, as classes determinadas ganham índice 1 e as classes determinantes ganham índice 3. O índice 2, também utilizado, é reservado a classes híbridas.

A título de exemplo, veja-se a consulta (g), a seguir, onde só as palavras em negrito são significativas para o sistema.

(g) Qual o **realizado** no **mês de junho** de toda a **região norte** nas contas de **mão de obra** . Dê o resultado por **executante** e por **conta**.

PALAVRA RECONHECIDA	CATEGORIA SINTÁTICA CORRESPONDENTE
REALIZADO	P 1
JUNHO	P 3
NORTE	F 1
MÃO DE OBRA	F 3
EXECUTANTE	N 1
CONTA	N 1

(h) GRAMÁTICA DE DETERMINAÇÃO (NUCLEAR)

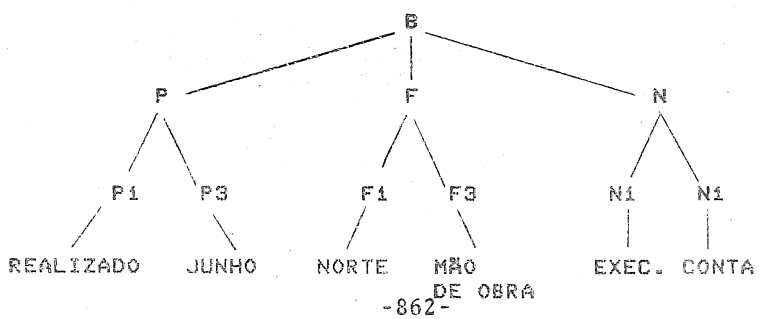
B -> F P N

F -> F1 F3

P -> P1 P3

N -> N1
N -> N N1

(i) ÁRVORE CORRESPONDENTE À ANÁLISE BÁSICA DE (G)



A observação detalhada da árvore (i) poderia levar a supor que as GRAMÁTICAS DE DETERMINAÇÃO não passam de uma sofisticação sintática para o polêmico processo de análise por palavra-chave. Para afastar tal suposição, vejam-se as consultas abaixo, corretamente interpretadas pelo sistema FARAO, mencionado anteriormente, nas quais fica clara a maior potencialidade das GD's no que concerne os fenômenos da língua por ela tratados. Antes de cada exemplo será mencionado o caso a que ele se refere, e as palavras que foram necessárias para o processamento estão sublinhadas.

(j) COORDENAÇÃO SENTENCIAL E SUB-SENTENCIAL

Quero saber o orçado e o realizado anual do DED e do DIR e do DIE quero saber apenas o realizado em junho.

(k) ANÁFORAS COM PRONOMES PESSOAIS RETOS

Diga qual é o realizado em junho da região sul e do Distrito de São Paulo com a receita operacional de telex e diga depois qual era o orçado deles na conta de facilidades.

(l) RECUPERAÇÃO DE LISTAS ORDENADAS/QUANTIFICADAS POR PRONOMES INDEFINIDOS E/OU NUMERAIS ORDINAIS/CARDINAIS

Diga o realizado dos distritos de Manaus, Belém, Boa Vista e Rio Branco nos meses de abril e maio e depois dê o orçado dos três últimos nos mesmos meses.

(m) NEGAÇÃO E EXCLUSÃO

Quero saber o valor orçado de todos os executantes da área nacional, exceto o DRB, para a receita operacional de junho.

(n) ELIPSES

Qual o realizado dos distritos da região centro-oeste por mês e qual o orçado por conta?

Vários outros casos, de origem gramatical ou não (i.e. particularidades da base de dados consultada) são corretamente processados. Entretanto, os exemplos acima já são suficientes para demonstrar que as GD's não se restringem a um mero tratamento por palavra-chave. Na seção a seguir serão enumeradas as características fundamentais desta metodologia, que garantem a devida interpretação de um grande volume de consultas em linguagem natural ou pseudo-natural a uma base de dados.

3 - GRAMÁTICAS DE DETERMINAÇÃO

Para obter os resultados indicados na seção anterior, as **gramáticas de determinação** incorporam uma série de dispositivos e estratégias, que serão descritos a seguir. Dentre eles figuram a **leitura seletiva da entrada** e uma interpretação semântica próxima de uma **tradução para comando de consulta**, que não são propriamente parte do formalismo sintático, mas que são pressupostas por ele.

A **leitura seletiva da entrada** (ou consulta) é o que assegura a flexibilidade da expressão do usuário. Nem todas as palavras por ele empregadas são **reconhecidas** pelo sistema interpretador. Apenas aquelas que correspondam a uma informação semântica relevante ou a um vocábulo da língua portuguesa que sinalize processos sintáticos tratados (e.g. pronomes, conjunções e partículas diversas) serão reconhecidas à primeira varredura da consulta pelo módulo de análise lexical. Exemplos da seção 2 mostram, em consultas possíveis, quais palavras são "vistas".

O interpretador semântico orientado para uma **tradução** da consulta em linguagem (pseudo)natural para uma **query language** é fortemente inspirado pela proposta dos compiladores de linguagens de programação. Ressalta-se aqui, igualmente, a distinção entre a consulta a uma **base de dados** (estática) e a consulta a uma **base de conhecimentos** (dinâmica). Neste último caso, visto se tratar de **sistemas inteligentes**, a base é ampliada pela informação contida nas consultas (através de inferências). Assim, o nível de análise fina do interpretador **precisa ser** mais sofisticado do que na consulta a bases de dados sem pretensões inferenciais complexas.

Concentrando-se no aparato **sintático**, as GD's estão aliadas a um processo de parsing onde figuram os seguintes componentes: (a) **regras** e (b) **meta-regras**.

REGRAS

As regras de uma gramática de determinação podem ser **declarativas** ou **programáticas**. Quando declarativas, são **regras de reescritura**, que refletem uma análise **bottom-up** da entrada, no padrão:

(j) A B C -> X Y Z

onde A, B e C, X, Y e Z são constituintes sintáticos, terminais ou não-terminais, restringidos por condições específicas que subcategorizam as regras em (1) livres de contexto, (2) sensíveis a contexto e (3) transformativas. Quando programáticas, as regras efetuam modificações drásticas na estrutura em análise (e.g. cópia de constituintes) ou decidem ambiguidades entre elementos não-contínuos da mesma.

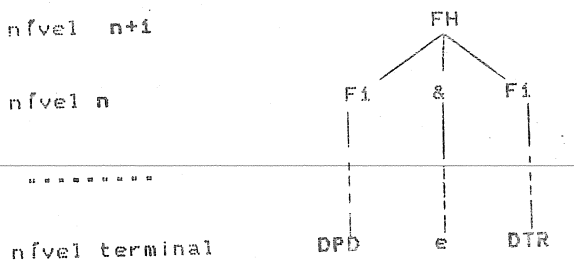
REGRAS LIVRES DE CONTEXTO

A nomenclatura adotada pela GD é mais alusória do que definitória, no sentido de que expressões como livre de contexto ou sensível a contexto não refletem exatamente os preceitos a elas associados na teoria das linguagens formais. Por exemplo, o sentido da leitura de uma regra declarativa da GD é o inverso do sentido normalmente utilizado (gerativo). Como a análise da entrada é bottom-up, as regras se enunciam dos terminais para a raiz, e não da raiz para os terminais.

O padrão de regra declarativa enunciado em (j), quando apresenta a restrição de que à direita da seta só se pode ter um símbolo e de hierarquia superior aos símbolos da esquerda, especifica uma regra livre de contexto. Em outras palavras, as regras livres de contexto agregam constituintes de nível n sob um único constituinte de nível n+1. A operação destas regras reflete um movimento vertical na árvore representativa da análise.

EXEMPLO DE REGRA LIVRE DE CONTEXTO

Fi & Fi -> FH



REGRAS SENSÍVEIS A CONTEXTO

Quando o padrão das regras declarativas (j) apresenta a restrição de o módulo da cadeia à esquerda da seta ter de ser igual ao módulo da cadeia à direita (i.e. o mesmo número de constituintes tem de aparecer de um e de outro lado da seta) e apenas um dos constituintes à esquerda é reescrito à direita, trata-se de uma regra sensível a contexto. Estas regras não efetuam agregação, e sim uma mudança de classe, o que significa que os constituintes de nível n, à esquerda da seta, permanecem no mesmo nível n, após a reescritura de um deles, no contexto especificado pelos demais. O movimento relativo a estas regras, na árvore representativa da análise da entrada, é horizontal.

Para ilustrar o efeito de uma regra sensível a contexto, seja o seguinte:

- os numerais 4 e 5 têm classe sintática Q3, significando quantificação atribuída.

- as palavras divisão ou divisões têm classe ND, arbitrariamente correspondendo a um nome de campo.

Claramente, as construções sintáticas (k) e (l), abaixo, representam um caso em que o contexto desambigua a classificação dos numerais, resolvendo se eles são (1) a quantidade de divisões ou (2) a designação das divisões.

(k) ... 4 ou 5 divisões têm orçamento controlado ...

(l) ... a divisão 4 ou 5, do DPD, tem orçamento controlado ...

As regras sensíveis a contexto capazes de solucionar este problema são (m) e (n) abaixo.

(m) ND Q3 -> ND Fd (porque 4 é um fornecido transformado)

(n) Fd ou Q3 -> Fd ou Fd (para propagar a transformação para todos os Q3 ambíguos que houver)

Este exemplo ilustra também um dispositivo muito útil das GD's que são as classes transientes. Estas classes são manipulações simbólicas em contexto, nas quais uma determinada classe assume temporariamente uma outra denominação, a fim de que determinadas reescrituras possam ser feitas. No caso, quando 4 e 5 são designações de divisões, a classe final adequada é F1, mas reescreve-se temporariamente como Fd para que se possa propagar a transformação de Q3 em F1 a todos os elementos coordenados de uma cadeia. Uma regra sensível a contexto final transforma todos os Fd em F1, para normalizar a análise.

REGRAS TRANSFORMATIVAS

As regras transformativas das GD's operam, como o nome já o diz, transformações estruturais. O contexto geral de reescritura (j) é restringido de maneira especial, conforme se trate de uma ou outra transformação.

Para as transformações de apagamento, a restrição de (j) é que o lado esquerdo da seta tenha módulo maior ou igual a 2 e que o lado direito da seta seja menor que o módulo do lado esquerdo. Para esclarecer o motivo e o funcionamento das regras transformativas de apagamento, seja a consulta (o), abaixo:

(o) Diga o orçamento da região norte para o mês de agosto.

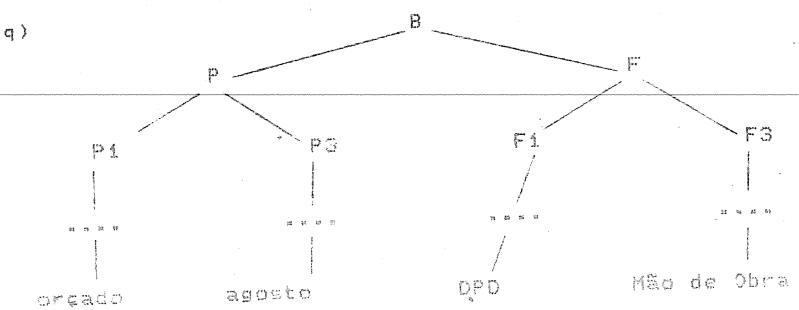
Observe-se que **norte** só pode ser região e que **agosto** só pode ser mês. Isto revela uma **redundância** entre os dois componentes das expressões em negrito de (o). Uma vez detectada esta duplicação, pode-se perfeitamente dispensar as palavras **região** e **mês**. Isto se faz através da regra **transformativa de apagamento** (p), a seguir, que é um caso particular de regra sensível a contexto, isto é: não opera na vertical, mas na horizontal, relativamente à árvore representativa da estrutura de (o).

(p) Nr Fr -> Fr (pois a **semântica de norte** já presume região)
Nm Fm -> Fm (por motivo semelhante ao caso anterior)

Portanto, todo N? F? -> F?, seja ? o símbolo que for.

A outra regra transformativa prevista para uma GD é a de **permutação**, em que a ordem dos elementos constantes do lado esquerdo da seta é alterada do lado direito. O contexto geral (j) fica restringido da seguinte forma: **para a regra de permuta, o lado esquerdo da seta tem de ter módulo maior ou igual a 3 e o módulo do lado direito da seta tem de ser igual ao do lado esquerdo (e constem as mesmas classes em amboso os lados).** Trata-se, novamente, de uma **subcategorização** de regra sensível a contexto. As **permutas** têm a função de normalizar estruturas para reescrituras em estágio avançado de análise. Se se lembrar que o objetivo final da estruturação da consulta é atingir algo como (q), abaixo,

(q)



pode acontecer de estes elementos terem aparecido na consulta em ordem diferente daquela que consta em (q). Veja-se (q')"

(q') Quero o orçado do DPD para mão de obra em agosto.

A certa altura da análise, o que se terá é algo como:

– P1 (Orçado) F (F1 (DPD) F3 (Mão de Obra)) P3(Agosto)

Para que se possa agregar P1 e P3 sob P, pela regra livre de contexto P1 P3 -> P, é preciso que se permutem os elementos da consulta (q'), o que é feito através da regra (r) abaixo.

(r) P1 F P3 -> P1 P3 F (note-se que, sendo sensível a contexto, esta regra **não agrega elementos**, isto é, opera na **horizontal** da árvore)

As regras **programáticas** da GD são procedimentos especiais para mudanças muito drásticas na estrutura em análise. Pode-se exemplificá-las através de dois casos típicos: **cópias** e **apagamentos incondicionais**.

Os casos de **cópia** estão envolvidos em todas as **anáforas**, onde um pronome está ocupando o lugar de um elemento já mencionado na sentença (ou no contexto geral do diálogo). Contudo, a estrutura final deve apresentar não o pronome, mas o elemento em questão, reproduzido. Por exemplo:

(s) Quero o realizado da região sul e do distrito de São Paulo e, depois, o orçado deste último para o telex nacional.

A expressão **deste último**, no processo de resolução da anáfora refere-se ao **distrito de São Paulo**, que deverá constar da árvore final, **duplicado** (e não pronominalizado). Ora, para isto é preciso **copiar** o constituinte **distrito de São Paulo**, o que se faz através de uma **regra programática** (ou procedure).

Outra utilização importante das regras programáticas é o **apagamento incondicional**. Para exemplificá-lo, considere-se o emprego das vírgulas (,) na língua portuguesa. De maneira geral, ao escrever, muitas pessoas empregam erroneamente as vírgulas. Devido a este problema, elas não podem ser confiáveis, e, consequentemente, não precisam ser "vistas" pelo processador na maior parte do processo de **parsing**. Entretanto, em construções coordenadas (entre outras), elas desempenham um papel importante. Veja-se o grupo (t) de regras referentes à coordenação (recursiva) de F1's sob um nódulo FH.

(t) F1 e F1 -> FH

F1 , FH -> FH

O grupo (t) processa estruturas como: "DPD, DTR, DTF e DCD", onde cada elemento tem classe F1.

Passado este ponto, as vírgulas da consulta não têm mais utilidade para a análise sintática proposta. Neste caso, uma regra programática apaga todas as vírgulas da sentença e a estrutura de trabalho do analisador fica descarregada de elementos espúrios para as etapas seguintes.

META-REGRAS

As meta-regras das GD's são o controle do processo de análise. As regras citadas até este momento são divididas em ciclos ou blocos, que se aplicam sequencialmente em um mesmo estágio. Cada bloco contém regras que se encontram ordenadas entre si, para que a aplicação de uma não impeça a aplicação de outra, a seguir. As meta-regras são regras que chamam os blocos de regras declarativas e programáticas, possibilitando - inclusive - que determinado ciclo seja re-aplicado, por exemplo, depois de uma operação de cópia de constituinte (o que é indispensável).

Esta estratégia tem a vantagem de impedir que todas as regras da base de conhecimento do sistema sejam tentadas a cada passo da análise. Assim, as regras em que só constam elementos altamente agregados (i.e. perto da raiz da árvore) são carregadas apenas em estágio compatível com esta exigência. Se os elementos no estágio corrente são terminais, a regra final P F -> B não é chamada por qualquer meta-regra.

O efeito de controle das meta-regras pode ser graficamente explicado pela figura a seguir.

META-REGRA 1	META-REGRA 2	META-REGRA 3
A1 B1 C1	B2 C2 A2 C2	B3 A3 B3 C3

CICLO A1 regra1 regra2 ... regran	CICLO A2 regra1 regra2 ... regran	CICLO A3 regra1 regra2 ... regran
CICLO B1 regra1 regra2 ... regran	CICLO B2 regra1 regra2 ... regran	CICLO B3 regra1 regra2 ... regran
CICLO C1 regra1 regra2 ... regran	CICLO C2 regra1 regra2 ... regran	CICLO C3 regra1 regra2 ... regran

Observe-se que uma vez chamados os ciclos, as regras que deles constam são lidas sequencialmente. Também, as meta-regras são chamadas sequencialmente, ou seja: primeiro a meta-regra1, que é toda processada, depois a 2 e a 3.

4 - POTENCIALIDADE DAS GRAMÁTICAS DE DETERMINAÇÃO

As Gramáticas de Determinação apresentam algumas vantagens bastante interessantes para a aplicação de interface a bases de dados. Dentre elas destacam-se as que se discutem a seguir.

FLEXIBILIDADE EXPRESSIVA

O principal atrativo das GD's é a flexibilidade que o usuário tem na expressão de suas consultas. A leitura seletiva da entrada, aliada ao fenômeno de sintatização de categorias semânticas, permite que consultas em linguagem corrente, em linguagem telegrafada, em linguagem semi-formal e, por conseguinte, em boa parcela de casos de agramaticalidade sejam devidamente interpretadas, sem que se obrigue o usuário a uma expressão correta ou treinada.

EFICIÊNCIA ANALÍTICA

O segundo grande atrativo das GD's é a utilização de categorias sintáticas que são e enquanto são relevantes em termos interpretativos. Isto pode ser percebido em pelo menos duas etapas distintas: inicialmente na leitura seletiva da entrada e, em seguida, nas regras de apagamento. Estes recursos contribuem para que a estrutura final que será entregue ao interpretador seja limpa e desbastada. Certamente,esta característica contribui para aumentar a eficiência representativa da estrutura sintática de trabalho e a eficiência do processo de interpretação, visto que praticamente não há elementos de semântica vazia chegando ao processador do significado da consulta (o que pode ocorrer em outras abordagens alternativas, sobretudo naquelas que seguem propostas tradicionais para a gramática do português).

FERRAMENTA METODOLÓGICA

Outro ponto interessante das Gramáticas de Determinação é o fato de que - considerando que elas foram desenvolvidas para a aplicação de consultas a bases de dados - suas categorias sintáticas apresentam uma correspondência com a modelagem da base de dados utilizada. Por exemplo, os valores assumidos pelos campos da base têm de apresentar uma classe global F, ao passo que os nomes de campos tenderiam a ter uma classe global P. Os atributos tenderiam a ter função determinante 3, ao passo que as entidades deveriam ter função determinada 1, em seus respectivos constituintes. Esta possibilidade tem motivado o desenvolvimento de pesquisas sobre as GD's, no sentido de se conseguir automatizar o mapeamento da modelagem de dados para uma proto-gramática GD, na qual estejam constando as macro-relações de determinação para as consultas a serem processadas.

ELASTICIDADE

Por elasticidade entende-se, aqui, a capacidade de a GD expandir-se em sentido vertical - isto é, tornando-se mais profunda e perspícua, para perceber fenômenos da língua de maior sutileza - e em sentido horizontal - isto é, tornando-se aplicável a várias bases de dados, o que remete ao fator transportabilidade. No que tange à perspicuidade, uma Gramática de Determinação pode ser tão profunda e detalhada quanto se queira, não havendo - nos mecanismos utilizados - um indício evidente de que determinadas construções esperadas na consulta em linguagem natural a bases de dados não possam ser compreendidas. A manipulação simbólica facultada pelas regras de reescritura não parece restringir a priori o tratamento de fenômenos de pressuposição, por exemplo, marcados por quantificadores ou outras categorias. Já no que tange à transportabilidade, a dependência de domínio das interfaces em linguagem natural é uma imposição de ordem prática. Entretanto, a separação das regras em ciclos pode perfeitamente isolar fenômenos linguísticos supra-ambientais (coordenação, cópias, permutas e outros) que estes são transportáveis (porque são essencialmente sintáticos).

A desvantagem que apresentam as GD's em seu atual estágio de desenvolvimento é a sua complexidade. O número de regras é grande e a novidade das categorias sintáticas com que elas trabalham pode dificultar bastante a manutenção de uma base de conhecimentos linguísticos. O próprio processo de manipulação simbólica por reescritura não é amplamente dominado e pode representar um obstáculo para o desenvolvimento de interfaces para GD's através desta metodologia. Entretanto, como já se disse, há pesquisas sendo realizadas para minorar este problema, possibilitando que se tirem todas as vantagens das GD's naquilo que elas apresentam de positivo e atraente.

5 - CONCLUSÕES

O campo de processamento de linguagem natural, no Brasil, embora interdisciplinar por definição, não tem contado com o desenvolvimento de metodologias efetivamente interdisciplinares. A maioria dos trabalhos se desenvolve apenas no campo da informática, sem a participação de lingüistas. Se é verdade que todo cientista computacional brasileiro fala português, não é verdade que todo lingüista possa desenvolver sistemas computacionais. Isto encoraja pesquisas unilateralmente realizadas e - obviamente - o lado produtivo não é o da linguística.

As GRAMÁTICAS DE DETERMINAÇÃO foram desenvolvidas por um especialista em linguística, há alguns anos participando ativamente em pesquisas na área de inteligência artificial. A leitura detalhada de sua bibliografia básica [Souza,1988] mostra

claramente a tendência do trabalho. Contudo, o objetivo de seu desenvolvimento é o de sensibilizar os pesquisadores do processamento de linguagem natural aplicada ao português para a necessidade da criação de um novo espaço de investigações conjuntas, onde se possam contemplar as especificidades do processador (computador) e da linguagem (semi-natural, uma vez que atualmente digitada), obtendo-se interpretadores mais eficientes em termos computacionais e flexíveis em termos expressivos para a atividade de consulta a bases de dados.

REFERÊNCIAS BIBLIOGRÁFICAS

BURTON,R. **Semantic Grammar: an Engineering Technique for Constructing Natural Language Understanding Systems**, S.L., S.P.,1976.

COELHO,H. **Sobre a Engenharia da Linguagem**, in Anais do XIV Congresso Nacional de Informática, SUCESU, São Paulo, 1981.

GARCIA,A.,CARVALHO,E.,FORTUNY,J. **Aspectos da Implementação de um Processador de Linguagem Natural em PASCAL**, in Anais do III Simpósio Brasileiro de Inteligência Artificial, SBC, Rio de Janeiro, Novembro de 1986.

HENDRIX,G.,SACERDOTTI,E. et alii **Developing a Natural Language Interface to Complex Data**, in Transactions of the ACM on Data Base Systems, 3: 105-147, Junho,1978.

SOUZA,C. **Consulta a Bases de Dados em Português Corrente**, in Anais da X Reunião Anual do ADAGRUPO, Águas de Lindóia, Abril, 1987.

_____ **Gramáticas de Determinação: uma Ferramenta para o Processamento da Língua Portuguesa**, Tese de Doutorado, Depto. de Letras, PUC/RJ, Janeiro, 1988. (Versão Final)

WALTZ,D. **English Language Question-Answering System for a Large Relational Data Base**, in Communications of the ACM 21:526-536, Julho, 1978.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

CORREÇÃO DE ERROS DE ORTOGRAFIA

ATRAVÉS DA FONÉTICA

em textos escritos em Francês

por

COURTIN, Jacques, KOWARSKI, Irène, DUJARDIN, Danielle,
GENTHIAL, Damien, DE LIMA, Vera Lúcia Strube

Laboratoire de Génie Informatique
IMAG - Campus
38041 Grenoble cedex FRANCE

1 - INTRODUÇÃO

Motivação

Torna-se cada vez mais importante, para um sistema que realiza o tratamento de textos escritos em língua natural, a disponibilidade de uma análise eficaz a nível léxico, sintático e mesmo semântico. A detecção e a correção de erros de Ortografia aparecem aí como problemas fundamentais, uma vez que permitem passar à verificação da estrutura sintática e ao exame do texto buscando seu significado.

O problema da detecção e correção de erros de Ortografia tem sido amplamente analisado, tanto no que tange às grafias contendo um único erro de omissão, inserção, substituição ou transposição, como proposto por Damerau [DAM 64], como no que diz respeito às grafias contendo vários erros, segundo os estudos baseados na distância de Levenshtein [LEV 66].

Até o momento, todavia, os sistemas vêm propiciando uma correção normalmente fundada sobre uma única técnica, e especialmente para palavras contendo apenas um erro ortográfico. Raramente é proposta uma análise a nível sintático ou semântico.

Procuramos elaborar um sistema combinando várias técnicas de detecção/correção, com possibilidades, inclusive, de detecção e correção a nível sintático e semântico, para uma aplicação específica.

Atualmente, no sistema PILAF, já implementamos técnicas de detecção/correção a nível de palavra, bem como um verificador de regras de sintaxe a partir das estruturas de dependência associadas a uma frase. Os trabalhos em andamento têm por meta a detecção/correção de erros a nível sintático e semântico, propondo diferentes arquiteturas de detecção/correção para uma aplicação.

Objetivos

Neste artigo, descrevemos o método de correção de erros de Ortografia através da Fonética concebido e implementado no sistema PILAF.

O método aqui apresentado foi proposto por Courtin [COU 77] e compreende a tradução fonética de uma palavra mal ortografada, seguida da reconstituição de sua correspondente correta, utilizando estruturas de dados disponíveis a nível léxico. Um método de correção visando a composição fonética da palavra foi sugerido, também, por Peterson [PET 80], para a língua inglesa. Os componentes fonéticos, podendo se relacionar a diferentes cadeias gráficas, permitiriam calcular grafias alternativas para a palavra mal ortografada. Para a língua francesa, a alternativa de

substituição de cadeias gráficas já foi analisada em trabalhos como o de Maret [MAR 87].

Com a finalidade de proporcionar uma compreensão global do assunto, apresentamos aqui uma descrição geral do sistema PILAF - parte Morfologia e parte Sintaxe - inclusive os métodos de correção de erros de Ortografia disponíveis no sistema.

Em seguida, a correção de erros de Ortografia através da Fonética é detalhada em todas as suas etapas, desde a tradução fonética da palavra errada, até o enunciado de grafias alternativas.

Como conclusão, são abordados aspectos considerados importantes na proposição de arquiteturas de correção de erros de Ortografia, bem como a continuação dos trabalhos no sistema PILAF.

2 - PILAF

O sistema

O sistema PILAF (Procédures Interactives Linguistiques Appliquées au Français) é um software de tratamento interativo da língua natural. Analisando uma frase ou um texto, a partir da construção de estruturas de dependência e da pesquisa num dicionário, PILAF efetua a detecção de erros em dois níveis - Morfologia e Sintaxe. Os erros detectados podem ser corrigidos por diferentes mecanismos encontrados no sistema.

A análise Morfológica PILAF trata os erros detectados a nível de palavra, e propõe um conjunto de possibilidades corretas que podem substituir a palavra errada. A proposição de correção é feita utilizando os diferentes métodos de correção disponíveis no sistema PILAF.

O módulo de análise sintática detecta e corrige os erros a nível da construção da frase: os erros de linguagem.

Os níveis de correção

Consideremos, por exemplo, o tratamento da seguinte frase escrita em francês:

Les jirraphes mange.

A nível de palavra, PILAF detecta e corrige **jirraphes**, substituindo-a por **girafes**. A frase original, então, toma a forma:

Les girafes mange.

A nova versão da frase, após um tratamento a nível de palavra, ainda não está correta. O sujeito (no plural) não concorda em número com o predicado (no singular).

É a análise sintática, então, que se ocupa da detecção/correção desse segundo tipo de erro, propondo alternativas para o acordo sujeito - predicado, como

Les girafes mangent.

ou

La girafe mange.

Os métodos de correção de erros de Ortografia

Os métodos disponíveis no sistema PILAF, para a correção de erros de Ortografia, são quatro:

- a Geração pela Morfologia;
- o método da "clé squelette";
- o dicionário personalizado;
- a correção através da Fonética;

A decisão pela aplicação de um determinado método depende dos tipos de erros encontrados no texto. O algoritmo de aplicação dos métodos de correção depende da arquitetura de correção utilizada, e esta arquitetura é definida analisando-se um conjunto significativo de erros encontrados no texto a ser tratado.

Os métodos de correção de erros de Ortografia utilizados no sistema PILAF serão descritos brevemente, mais adiante, neste trabalho.

3 - O TRANSDUTOR DE ESTADOS FINITOS NO SISTEMA PILAF

Uma descrição geral do transdutor utilizado no sistema PILAF

PILAF utiliza o transdutor proposto e descrito por Courtin [COU 77].

Este transdutor é bastante geral. Ele permitiu, no início, a realização da Análise Morfológica PILAF e, a seguir, outras aplicações tais como a Geração pela Morfologia, a Tradução Fonética etc.

Tomemos, por exemplo, a primeira aplicação do transdutor: a Morfologia.

O objetivo da análise morfológica, utilizando o transdutor, é de segmentar uma frase para obter cadeias de componentes, e determinar, ao mesmo tempo, um certo número de informações lingüísticas sobre as cadeias resultantes da segmentação.

O princípio da segmentação é de determinar as palavras ou grupos de palavras componentes da frase, denominados segmentos. A determinação de um segmento consiste, então, de sua decomposição em prefixo, base, sufixo e desinência.

Para proceder à segmentação, dispomos:

- de um dicionário de prefixos, bases, sufixos e desinências, que permite, por identificação, a decomposição de uma parte contígua da cadeia de entrada, e
- de uma gramática que deve infirmar ou confirmar a concatenação destes elementos na constituição do segmento procurado.

As informações lingüísticas são associadas a modelos lingüísticos. Os componentes do dicionário de prefixos, bases, sufixos e desinências são lingüísticamente descritos através de modelos.

Para cada segmento determinado, é preciso fornecer as informações lingüísticas que serão utilizadas pelo modelo seguinte. Trata-se de fornecer todas as interpretações de todas as decomposições de um segmento.

Para certas aplicações, como a Tradução Fonética, é necessário produzir uma nova cadeia, à partir da cadeia de entrada. A cadeia resultante é produzida pela concatenação de códigos associados aos modelos aplicados durante o processo de tradução.

Os códigos associados aos modelos são denominados códigos de derivação, e só existem no caso das aplicações visando a derivação de uma nova cadeia a partir da cadeia de entrada.

O mesmo transdutor aqui descrito se adapta não só à Tradução Fonética, como também à fase de reconstituição gráfica, no método de correção de erros de Ortografia pela Fonética, empregando-se aí um dicionário e um conjunto de modelos e regras apropriados. As diferentes estruturas de dados utilizadas pelo transdutor nestas aplicações serão descritas quando do detalhamento do método de correção pela Fonética.

4 - MÉTODOS DE CORREÇÃO A NIVEL DE PALAVRA

A Geração pela Morfologia

Este método consiste em encontrar uma base morfológica dentro da palavra mal ortografada, e calcular todas as formas possíveis para

esta base, a partir dos modelos e regras apropriados.

Por exemplo, seja **bretone** uma grafia incorreta detectada por PILAF. O método da Geração encontra, nesta palavra, a base **breton**. A partir desta base, são geradas as grafias **breton**, **bretons**, **bretonne** e **bretonnes**. Do conjunto das palavras geradas, a opção **bretonne** é filtrada como a mais adequada.

O método da "clé squelette"

Este método é baseado na construção de uma chave de pesquisa, a partir de uma palavra.

A chave proposta é calculada tomando certas das letras que formam a palavra, segundo critérios bem determinados. A primeira letra é guardada; em seguida figuram todas as consoantes, na mesma ordem em que aparacem na palavra, sem repetição. As vogais são então anexadas à chave, na mesma ordem em que aparecem na palavra, sem repetição.

O método consiste em obter a "clé squelette" para uma palavra mal ortografada, e então recuperar todas as palavras existentes no dicionário que se caracterizem por possuirem esta mesma "clé squelette", ou, eventualmente, palavras possuindo uma "clé squelette" bastante próxima da chave calculada.

Tomemos como exemplo a grafia **aréoport**. Calculando a chave proposta, obteremos a cadeia **arpteo**.

As variantes mal ortografadas da palavra **aéroport** se caracterizam pela mesma "clé squelette", que é também a chave obtida para a palavra **aréoport**. Fazendo-se uma pesquisa no dicionário, as palavras possuindo a mesma chave ou chaves vizinhas são recuperadas, e entre elas se encontra a grafia correta.

O dicionário personalizado

Cada usuário de um sistema de tratamento de textos pode apresentar suas próprias características com relação aos problemas de datilografia e de ortografia.

O dicionário personalizado é uma técnica que permite memorisar, para um dado usuário ou uma dada aplicação, os erros mais habituais, e suas correções adotadas no passado.

A correção pela Fonética

Um usuário que conhece pouco de uma língua, no que diz respeito à

Ortografia, mas que conhece razoavelmente bem a língua falada, pode ter dificuldades com relação à escrita, propondo uma grafia errônea para uma palavra que sabe pronunciar corretamente.

O modo de correção através da Fonética se aplica a palavras contendo um ou mais erros de Ortografia, desde que preservada sua boa forma oral. Por exemplo, seja **eaukurense** uma forma errônea de escrever, em francês, a palavra **occurrence**: ambas as grafias podem ser pronunciadas da mesma maneira.

O algoritmo de correção através da Fonética calcula todas as cadeias orais possíveis para uma palavra mal ortografada. Utilizando estas cadeias orais, reconstituimos as grafias, usando as bases e as flexões armazenadas no sistema PILAF. As grafias produzidas são verificadas quanto a sua correção, e somente aquelas que constituem palavras da língua francesa são propostas como resultado.

5 - A CORREÇÃO DE ERROS DE ORTOGRAFIA ATRAVÉS DA FONÉTICA

A arquitetura geral de correção

No sistema PILAF, foi concebido um algoritmo para resolver, em três níveis, o problema da correção de erros de Ortografia através da Fonética. Este algoritmo é mostrado na Figura 1.

Num primeiro nível, é efetuada uma tradução fonética, a qual tem por objetivo calcular todas as cadeias orais possíveis para pronunciar a palavra mal ortografada, segundo os princípios da Lingüística Francesa.

Num segundo nível, cada cadeia oral é analisada com relação às grafias que permite obter. São então propostas grafias alternativas para a palavra mal ortografada.

A reconstituição gráfica para uma cadeia oral pode produzir uma ou várias cadeias gráficas, que são então verificadas quanto a sua correção, no terceiro nível do algoritmo.

Todos os três níveis que compõem a técnica de correção através da Fonética fazem uso do autômato concebido para a Morfologia PILAF, preservado seu princípio de funcionamento, porém empregando estruturas de dados específicas.

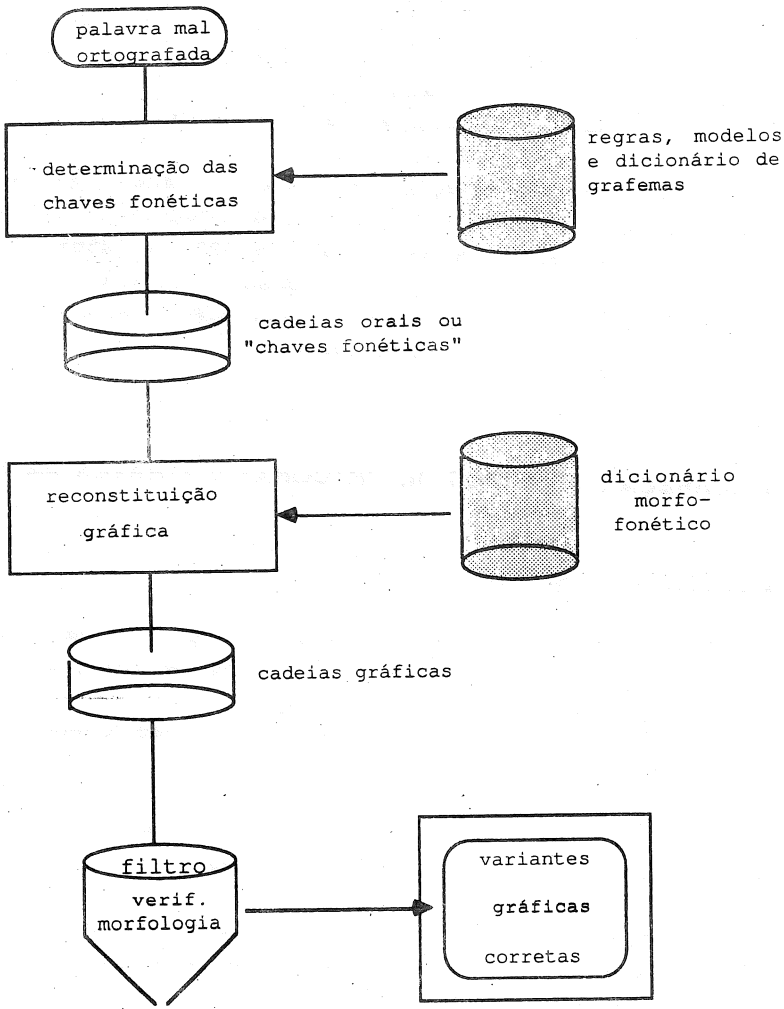


FIGURA 1 - ARQUITETURA GERAL DE CORREÇÃO ATRAVÉS DA FONÉTICA

6 - A CADEIA ESCRITA E A CADEIA ORAL

Grafemas e Fonemas

Um grafema pode ser caracterizado como a menor unidade distintiva ou significativa da cadeia escrita [CAT 80], composto de uma letra ou de um grupo de letras ou sinais auxiliares, possuindo uma referência fônica na cadeia oral.

O conceito de fonema encontra-se mais propriamente ligado ao oral: um fonema é a menor unidade distintiva da cadeia oral.

Considerando-se a correção de erros de Ortografia através da Fonética, o conceito de grafema foi extendido para permitir cadeias escritas mais longas, tratadas aqui como pseudo-grafemas ou, simplesmente, p-grafemas. Por exemplo, utilizamos a cadeia "enou" (concatenação de vários grafemas elementares) como um p-grafema.

Um conceito de pseudo-fonema foi também criado, a fim de permitir cadeias orais mais longas.

Os pseudo-fonemas ou, simplesmente, p-fonemas, são sempre constituídos ou por fonemas elementares, ou por cadeias de fonemas elementares.

A relação p-fonema --> p-grafema é do tipo 1 : n, $n \geq 1$ e inteiro. Por exemplo, seja o p-fonema "s." definido no sistema PILAF. Ao p-fonema "s." estão associados os p-grafemas "s", "ss", "c", "ç" etc.

A relação p-grafema --> p-fonema é do tipo 1 : m, $m \geq 1$ e inteiro. Por exemplo, seja o p-grafema "ch" definido no sistema PILAF. Ao p-grafema "ch" são associados os p-fonemas "ch." e "k.".

Escolha dos fonemas ligados à correção de erros de Ortografia

A geração de cadeias orais, quando aplicada à correção de erros de Ortografia, obriga a uma formalização e, até mesmo, uma simplificação, com relação a todas as possibilidades de cadeias orais para uma cadeia escrita, na língua francesa.

O problema dos grafemas ligados ao oral, na língua francesa, já foi abordado por vários lingüistas como Catach, Lallias, Demeyère etc. Observando as proposições feitas por estes autores, propusemos um conjunto de 33 fonemas elementares: os fonemas associados às vogais orais, às vogais nasais, às consoantes e às semi-vogais (ver Tabela 1).

Certos fonemas elementares combinados produzem sub-cadeias orais importantes, associadas a p-grafemas de freqüência considerável. Por exemplo, podemos citar o p-fonema "y.on.", encontrado freqüentemente nas flexões verbais, na língua francesa.

Os 33 p-fonemas elementares combinados permitem obter mais de 160 p-fonemas não elementares, associados a cadeias gráficas mais longas. Por exemplo, sejam os p-fonemas "o.in." e "e<.t." definidos no sistema PILAF. O p-fonema "o.in." pode ser associado ao p-grafema "oin ", ou ao p-grafema "oint " etc. O p-fonema "e<.t." é aplicável aos p-grafemas "ette ", "ete ", "ettes ", "etes " etc.

Os p-grafemas idealizados para calcular as cadeias orais de uma palavra são associados a um conjunto de modelos fonéticos, que são validados ou não durante o tratamento da palavra mal ortografada. A validação de um determinado modelo dá lugar a uma sub-cadeia oral: o p-fonema associado a este modelo.

Na determinação das cadeias orais, vários modelos podem ser validados a partir de um mesmo p-grafema, o que permite a produção de várias cadeias orais para uma mesma palavra.

Por exemplo, examinemos a grafia **bo**. Esta palavra, após um tratamento de detecção, é considerada errônea, o que dá origem a um processo de correção. O método de correção através da Fonética, aplicado à cadeia **bo**, calcula, inicialmente, todas as cadeias orais possíveis para esta grafia, observadas as características da língua francesa. Como resultado obtemos, neste caso, uma única cadeia oral: "b.o.".

Passamos então à geração de grafias alternativas, propondo, para a cadeia oral "b.o.", as grafias **beau** et **beaux**, e ainda outras como **beaus**, **bau**, **baux** e **bot**. As grafias geradas são então analisadas quanto à integridade, eliminando-se as cadeias incorretas.

PEDECIBA
INFORMATICA

TABELA 1

FONEMAS	PILAF
a appel, arbre, pâte	b boucle, brebis
e< lait, blé, après, manger,tête	ch tâche, couche, cheval
e> recherche, petit	d sud, deux
i ici, vie	f faveur, oeuf
o beau, mot, gauche	g guichet, gant
u tu, sûr	j jouet, genou
ou doux, tout	k kilo, cave, psychopathe, qui
oe oeil, leur	l lourde, baile
	m sommet, mouche
	n nez, annonce
y fille, mention, yoga	p pile, coupe
w squash, week-end, loin	r route, arrosoir, finir
	s passion, nation, cil, ça, son
	t thème, tard
in sapin, pain	v vif, wagon
an dans, temps, vent, lampe	z maison, zone
on coupon, tombé	
un un, jungle	gn baigner, châtaigne
	ing parking
muet haut, pommes	

7 - OS MODELOS FONÉTICOS

O conjunto de regras, modelos e dicionário

A determinação das cadeias orais para uma palavra é feita pelo transdutor de estados finitos, usando três tipos principais de estruturas de dados: as regras, os modelos fonéticos e o dicionário de p-grafemas.

As regras permitem a análise da palavra reconhecendo a posição de seus grafemas componentes. Nós definimos regras que permitem o reconhecimento do espaço em branco, do início e do fim de palavra e do conteúdo interno aos limites de início e fim de palavra.

Os p-grafemas são armazenados num dicionário, atualmente com mais de mil registros. A cada p-grafema, existe um modelo fonético associado.

Os modelos fonéticos, finalmente, contêm as informações referentes a sua aplicabilidade, num dado estado, pelo autômato. Entre tais informações, encontramos as regras de chegada a este estado, e as regras que validam este modelo. O sistema PILAF se serve de mais de 200 modelos fonéticos.

Quando um modelo é considerado aplicável, o p-fonema associado a este modelo é utilizado pelo autômato para a geração de uma sub-cadeia oral correspondente ao p-grafema analisado.

Tomemos, por exemplo, a grafia **sous**. Nela encontramos os p-grafemas **s**, **ou** e **s**, os quais possuem modelos fonéticos associados. A tradução fonética desta grafia produz duas cadeias orais: "s.ou." e "s.ou.s.". A produção de duas cadeias orais diferentes se deve à existência de várias ocorrências do p-grafema **s** no dicionário de p-grafemas, cada vez associado a um modelo diferente (neste caso, associado ao modelo muet ou ao modelo s).

O código de derivação

Os p-fonemas associados aos modelos fonéticos PILAF são armazenados como códigos de derivação, o que permite ao transdutor de estados finitos calcular uma cadeia oral resultante da tradução.

A cada modelo fonético corresponde zero ou um código de derivação. Os casos de ausência do código de derivação são devidos à existência de letras mudas ou que podem se comportar como mudas, hífen, etc.

Um mesmo código de derivação pode ser utilizado por vários modelos, desde que existam diferenças com relação às combinações de regras utilizadas.

8 - EXEMPLOS DE FORMAÇÃO DE CADEIAS ORAIS A PARTIR DE UMA CADEIA ESCRITA

A análise fonética da palavra mal ortografada pode produzir uma ou várias formas orais: suas diferentes possibilidades de pronúncia, segundo os modelos fonéticos. O sucesso, após a análise de um modelo, determina a concatenação do código de derivação do modelo à cadeia oral gerada.

Vejamos alguns exemplos:

Exemplo 1

palavra errônea:	vinku
cadeia oral:	v.in.k.u

Exemplo 2

palavra errônea:	okurranse
cadeia oral:	o.k.u.r.an.s.

Exemplo 3

palavra errônea:	peti
cadeias orais:	p.e<.t.i.
	p.e>.t.i.

Exemplo 4

palavra errônea:	chevale
cadeias orais:	ch.e<.v.a.l.
	ch.e>.v.a.l.
	k.e<.v.a.l.
	k.e>.v.a.l.

9 - CADEIAS GRAFICAS A PARTIR DE UMA CADEIA ORAL

As bases morfo-fonéticas

O dicionário de bases PILAF dá origem a uma estrutura de dados especificamente criada para melhor gerar as cadeias gráficas a partir de cadeias orais: o dicionário Morfo-Fonético ou, simplesmente, dicionário M-F.

Este dicionário é constituído pelas cadeias orais associadas às

bases morfológicas. Existe uma cadeia oral associada a cada base morfológica. Se várias bases idênticas possuem uma mesma cadeia oral associada, o dicionário M-F conterá uma referência, somente, a esta cadeia oral.

Todas as bases têm cadeias orais associadas, mesmo que sejam cadeias mudas.

Por exemplo, consideremos a base savant. Esta base se apresenta duas vezes consecutivas no dicionário de bases PILAF: está definida sob um modelo de adjetivo e também sob um modelo de substantivo. No dicionário M-F, esta base aparece uma vez somente, sob a forma da cadeia oral "s.a.v.an.". Este fato se deve à pronúncia idêntica das duas bases savant.

Consideremos, agora, a base arrêt. Esta base também esta presente duas vezes consecutivas no dicionário de bases PILAF. Ela está definida sob um modelo de substantivo, e também sob um modelo verbal o qual permite obter todas as formas do verbo "arrêter".

Neste caso, as duas ocorrências da base arrêt não se pronunciam da mesma maneira. Ao substantivo arrêt está associada a cadeia oral "a.r.e<.", enquanto que à base verbal arrêt está associada a cadeia oral "a.r.e<.t." (o fonema t deve estar presente porque ele é necessário em todas as conjugações do verbo "arrêter"). Ambas as cadeias orais aparecerão no dicionário M-F.

Uma terceira situação a mencionar é aquela que torna possível a formação de diferentes cadeias a partir de uma base morfo-fonética. Uma cadeia oral pode aparecer várias vezes no dicionário M-F, cada vez associada a uma base diferente. Tomemos a cadeia oral "s.e<." como exemplo. Esta cadeia aparece duas vezes no dicionário M-F, associada à base sec e à base cet.

Os registros do dicionário M-F são da seguinte forma:

```
o
o
s.a.v.an.                    savant
o
o
a.r.e<.                      arrêt
o
o
a.r.e<.t.                    arrêt

o
o
o
```

A produção de cadeias gráficas

A produção de cadeias gráficas é feita por uma versão simplificada do transdutor de estados finitos. Esta versão não utiliza regras nem modelos, uma vez que opera sobre um dicionário que já contém as cadeias e suas bases associadas: o dicionário M-F.

A cadeia oral é percorrida buscando-se todas as cadeias morfo-fonéticas possíveis. As bases morfo-fonéticas são então substituídas pelas bases morfológicas correspondentes, o que produz variantes gráficas para a palavra errônea. Durante este processo observam-se, para todas as grafias produzidas, as mesmas cadeias orais, ou cadeias orais próximas da palavra errônea.

Se a combinação de sub-cadeias orais não permite uma tradução, com relação ao dicionário M-F, a geração gráfica não produz resultados.

Analisemos a palavra **arête** como exemplo a corrigir. A cadeia oral "a.r.e<.t." é a chave fonética calculada para esta grafia. A reconstituição gráfica, a partir do dicionário M-F, obtém a cadeia "a.r.e<.t." como a mais longa cadeia oral contida na chave fonética calculada. A base associada à cadeia oral "a.r.e<.t." é então utilizada para produzir a primeira parte da grafia resultante: **arrêt**.

A continuação da grafia resultante é calculada analisando-se a seqüência da chave fonética. No caso proposto, a chave fonética foi, integralmente, levada em conta pela análise da sub-cadeia inicial. A geração gráfica termina, então, por adicionar alguns p-grafemas mudos à grafia resultante: **s, x, ent, e, es** etc, segundo regras de lingüística.

Como resultado de correção para o exemplo proposto, temos as grafias **arrête, arrêtent** e **arrêtes**. Utilizando um dicionário mais completo, as palavras **arète** e **arètes** também seriam incluídas no resultado.

Possibilidades de produção

A geração de cadeias gráficas pode ser controlada no que concerne ao número de grafias geradas.

O princípio de tradução utilizado esgota todas as alternativas de combinações de sub-cadeias orais na análise da chave fonética, produzindo, às vezes, grafias com menor probabilidade de corrigir a palavra mal ortografada.

É possível afirmar que a melhor alternativa de correção através da Fonética é certamente a primeira encontrada no dicionário M-F, se este dicionário for suficientemente completo para a aplicação em questão, e convenientemente ordenado.

O algoritmo PILAF de correção de erros de Ortografia através da

Fonética trata esta questão propondo ao usuário a escolha entre a produção da primeira ou de todas as grafias encontradas.

No que concerne às situações em que é possivel chegar-se a um resultado incompleto, devido à geração de somente a primeira grafia encontrada, o sistema toma algumas precauções.

Por exemplo, examinemos um caso de ocorrência de várias bases morfo-fonéticas idênticas. Seja **vinku** uma palavra mal ortografada detectada por PILAF. A palavra **vinku** produz uma única cadeia oral: "v.in.k.u.".

Enquanto isso, no dicionário Morfo-Fonético, temos:

°
°

u.	u

°
°

v.in.k.	vainqu
v.in.k.	vainc

°
°

Portanto, a fase de recuperação gráfica dá origem às grafias **vainquu**, **vainquue**, **vainquus**, **vainquues**, **vaincu**, **vaincue**, **vaincus**, **vaincues** etc, calculando um conjunto de alternativas que inclui todas as duas bases associadas ao verbo vaincrè.

Esta precaução evita interromper a geração antes de ser encontrada a base vainc, que é a mais adequada na correção da palavra **vinku**.

10 - O RESULTADO: CADEIAS GRAFICAS MORFOLOGICAMENTE CORRETAS

A necessidade de uma verificação

A geração gráfica, mesmo feita a partir de bases morfo-fonéticas, pode produzir um mau encadeamento de bases, introduzindo, entre os resultados corretos, grafias que não constituem palavras. Por exemplo, podemos citar a grafia **vainquu** durante a correção de **vinku**.

É necessário, então, realizar uma verificação quanto à correção das cadeias gráficas calculadas.

Esta verificação é feita utilizando-se a Morfologia PILAF, e propondo como resultado somente as grafias que constituam palavras corretas. A ação de verificação elimina do resultado final todas as grafias

consideradas como incorretas pela Morfologia PILAF.

Uma alternativa

A verificação de integridade das grafias geradas pode ser dispensada se um dicionário de formas da língua francesa fizer parte do sistema. É então possível testar diretamente as grafias geradas, à partir da pesquisa no dicionário de formas [COH 88].

11 - CONCLUSÃO

O módulo de correção desenvolvido trata em três níveis o problema da correção de erros de Ortografia: o cálculo de uma chave fonética de pesquisa, a produção de grafias para a chave proposta e uma verificação de integridade das grafias geradas.

Este algoritmo se aplica bem às palavras contendo vários erros de Ortografia, desde que a forma oral seja conservada. Alguns acentos também podem ser recuperados pelo método, e o mesmo ainda deixa aberta a possibilidade de considerar a palavra mal ortografada como a concatenação de várias palavras, sem separador (por exemplo, **equivalait**: omissão dos separadores em **et qui valait**).

O sistema funciona atualmente com um dicionário de teste, o qual estamos enriquecendo. A execução do sistema com um dicionário maior permitirá a obtenção de resultados mais completos.

Uma proposição a fazer, observando-se o comportamento autônomo de cada nível de nosso algoritmo de correção através da Fonética, seria a construção de uma arquitetura funcional paralela, para os três níveis do algoritmo desenvolvido. O paralelismo é uma alternativa que se mostra, neste caso, eficaz e, ao mesmo tempo, consideravelmente natural.

A multiplicidade de resultados gerados, atualmente, pelos métodos de correção de erros de Ortografia, nos leva à proposição de um processo de filtragem, eliminando as opções menos prováveis de correção. No sistema PILAF, uma filtragem sintática está em desenvolvimento, tendo por base a pré-análise sintática da frase contendo a palavra errônea. Partindo-se da construção das árvores sintáticas alternativas, certas características morfológicas da palavra mal ortografada podem ser conhecidas, e algumas alternativas sintaticamente nulas podem ser eliminadas do conjunto final de proposições de correção.

Analisando o perfil do usuário final, devemos agora explorar as diferentes formas de ligação possíveis entre os módulos de correção

disponíveis. Os erros de competência, por exemplo, são mais facilmente corrigíveis por um método que se aperceba das más flexões aplicadas a uma base (como o método da Geração pela Morfologia), ou por um método que procure os equivalentes fonéticos para uma grafia. Dificuldades de performance poderiam ser tratadas com o suporte de um dicionário personalizado, ou da "clé squellette". PILAF propõe um conjunto de arquiteturas destinadas a tratar, distintamente, os erros de competência e de performance, bem como o Ensino assistido por Computador.

A vantagem do método desenvolvido, com relação aos já existentes, reside na formação de grafias diretamente a partir de bases, e não a partir de grafemas alternativos, o que reduz consideravelmente o número de cadeias gráficas foneticamente calculadas. Considerando-se a forma como as bases morfo-fonéticas estão armazenadas, sua recuperação pressupõe que a correção mais provável seja a primeira a ser encontrada.

BIBLIOGRAFIA

CAT 80 CATACH, Nina. L'orthographe française - traité théorique et pratique. Ed. Nathan, Paris, 1980. 334 p.

COH 88 COHARD, Brigitte. Logiciel de détection et de correction des erreurs lexicales. Memoire présenté en vue d'obtenir le Diplôme d'Ingenieur C.N.A.M.. C.N.A.M., Centre régional associé de Grenoble, USTMG, Mars 1988.

COU 77 COURTIN, Jacques. Algorithmes pour le traitement intéractif des langues naturelles. Thèse d'état USTMG - INPG, Octobre 1977.

COU 86 COURTIN, Jacques, DUJARDIN, Danièle, Kowarski, Irène. Le système PILAF. Table Ronde Franco-Allemande sur les Langues Naturelles, Décembre, 1986.

DAM 64 DAMERAU, F. A technique for computer detection and correction of spelling errors. Communications of the ACM, Vol. 7 Num. 3, Mars 1964. pp. 171-176.

FOU 84 FOURNIER, J. "Sauvetage" de raisonnements en langage naturel. Thèse, Orsay Université Paris-Sud, Juin 1984.

LAH 86 LAHENS, François. Un modèle stochastique pour la vérification

et la correction automatique de textes: le système VORTEX. Thèse, Université Paul Sabatier. Toulouse, 1986. 166 p.

LEV 66 LEVENSHTEIN, V. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady, Vol. 10 Num. 8, 1966. pp. 707-710.

MAR 87 MARET, Dominique. Comparaisons de chaînes de caractères, acces lexicaux tolérants et applications. Thèse de Doctorat, Université de Sciences Sociales de Grenoble, Mai 1987.

PER 86 PERENNOU, Guy, DAUBEZE, Pierre, LAHENS,François. La vérification et la correction automatique de textes: le système VORTEX. T.S.I. - Technique et Sciences Informatiques, Vol. 5 Num. 4, 1986. pp. 285-304.

PET 80 PETERSON, James. Computer programs for detecting and correcting spelling errors. Communications of the ACM, Vol. 23 Num. 12, Décembre 1980. pp. 676-687.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa.
Buenos Aires, Setiembre 1988.

TRATAMIENTO DE TEXTOS DE QUIMICA EN CASTELLANO

Osvaldo Cairó Battistutti *

Sección de Computación
Departamento de Ingeniería Eléctrica
CINVESTAV - IPN

Apartado Postal 14-740 07000 - México DF

RESUMEN

En este artículo se presenta un sistema que lleva a cabo la comprensión de textos en lenguaje natural aplicado al campo de la química.
El sistema funcionalmente opera en dos fases. La primera fase corresponde al procesamiento del lenguaje natural y la segunda fase corresponde al mecanismo de inferencia del sistema.
El objeto de este artículo es presentar las técnicas utilizadas en ambas fases.

La información básica del sistema consiste esencialmente de las características físicas y químicas de los elementos de la tabla periódica. En base a esta información el sistema puede deducir propiedades, como así también combinaciones entre elementos (fórmulas y nombres de fórmulas).
El sistema fue implementado en LISP.

* Estudiante de Doctorado de la Sección de Computación.
Departamento de Ingeniería Eléctrica.

1 - INTRODUCCION

El sistema desarrollado lleva a cabo la comprensión de textos en lenguaje natural (dado el caracter de la linguística computacional, complicado en todos sus aspectos) aplicado al campo de la química, y concerniente a las propiedades físicas y químicas de los elementos de la tabla periódica.
La entrada del sistema es un enunciado y la salida del mismo es la respuesta a la interpretación semántica del texto ingresado.

Funcionalmente, el sistema opera en dos fases. En la primera fase se hace un análisis sintáctico y semántico del texto ingresado y en la segunda, el sistema hace inferencias para poder responder a las exigencias del usuario.

Para la primera fase, el sistema hace uso de una Red de Transición Aumentada (ATN) para efectuar los análisis sintáctico

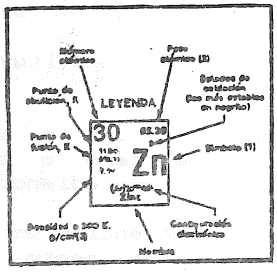
y semántico.
Sintáxis y semántica se presentan, en este trabajo, completamente fusionadas en el tratamiento del lenguaje natural y por lo tanto los análisis se realizan en forma concurrente a lo largo del procesamiento del texto ingresado.
El ATN lo podemos definir como un analizador sintáctico (intérprete) que contiene una gramática que se aplica de manera recursiva y de arriba hacia abajo y un diccionario de datos, los cuales son independientes entre sí. Es decir, que el intérprete, la gramática y el diccionario se encuentran separados, y por lo tanto cualquiera de ellos puede sufrir modificaciones sin alterar el resto del sistema.
Prueba de ello es que el sistema funcionó incluso con una gramática general para el castellano y los resultados fueron excelentes. Solo fue necesario definir otra gramática y otros items del diccionario de datos pero el intérprete utilizado fue el mismo para ambas.

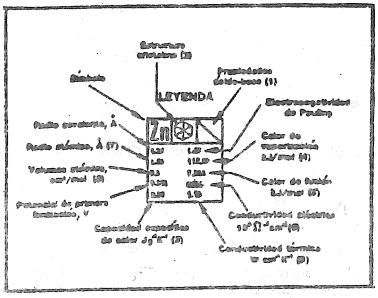
Para la segunda fase, citada anteriormente, el sistema utiliza una máquina de inferencias, un conjunto de reglas, y una base de hechos, las cuales son independientes entre sí. Es decir, cualquiera de ellas puede sufrir modificaciones sin alterar el resto del sistema.
Las reglas contienen la información que el experto utiliza para la resolución de sus problemas y representa el mecanismo válido de razonamiento o inferencia, no existe ambiguedad. El intérprete es el que accesa las reglas.

La información básica con la que se dota al sistema se presenta en la figura 1 y consiste esencialmente de las características físicas y químicas de los elementos de la tabla periódica.
En base a esta información, el sistema deducirá propiedades de los elementos como por ejemplo el ser o no alcalinos, alcalinotérreos, gases, líquidos, sólidos, halógenos, metales, no metales, metaloides,etc, o bien determinar a que grupos o períodos pertenecen. También podrá realizar combinaciones entre elementos para así obtener fórmulas y los nombres de fórmulas correspondientes.

Finalmente, puedo mencionar que el sistema fue implementado en LISP, corre a una gran velocidad y el espacio total de memoria ocupado por el sistema es de 63.146 bytes.

GRAFICA SOBRE LAS PROPIEDADES DE CADA ELEMENTO DE LA TABLA PERIODICA.





2 - EL PROCESAMIENTO DEL LENGUAJE NATURAL

Como se mencionó anteriormente, el sistema hace uso de un ATN para el procesamiento del lenguaje natural. La definición sintáctica del lenguaje de escritura se presenta a continuación y se dá en la Forma Normal de Backus.

Definición sintáctica del lenguaje del ATN

<red tran> ::= (<nombre subred> (<subred>))$^{+}$

<subred> ::= (<arco>)$^{+}$

<arco> ::= <nombre categoría>$^{+}$ <arc>* <estado>
<nombre subred> /
<nombre categoría>$^{+}$ <arc>$^{+}$ [<arco>]

<arc> ::= <estado> <acción> /
<estado> <paquete regla>

<nombre subred> ::= <identificador>
<nombre categoría> ::= <identificador>

<acción> ::= **ARREGLA** / **COMPARA** / **CONTROLA** / **CARGA** / **CARGAO**

<paquete regla> ::= **FAMILIA1** / **FAMILIA2** / **FAMILIA3** / **FAMILIA4**

<estado> ::= 0 / 1

<identificador> ::= <caract. alfabéticos>$^{+}$ <caract. numéricos>*

Se presenta a continuación la definición semántica del lenguaje del ATN.

Definición semántica del lenguaje del ATN

En la primera línea de la definición sintáctica se establece que una <red tran> está constituída por una o más ocurrencias de un <nombre subred>, seguido de una <subred>.
Un <nombre subred> indica el nombre de una <subred> o subgráfica.
La <subred> constituye el cuerpo de una subgráfica, y está formada por una o más ocurrencias de un <arco>.
Por medio de los <arco> se especifica qué elementos constituyen el lenguaje de escritura del ATN.

Se establece que un <arco> está constituído por : una o más ocurrencias de un <nombre categoría>, seguido de cero o más ocurrencias de un <arc> y finalmente de un <estado> precediendo a un <nombre subred>.
Otro modo de definir un <arco> es : una o más ocurrencias de un <nombre categoría> seguido de uno o más <arc> y opcionalmente de un <arco>.

Un <arc> se define como un <estado> seguido de una <acción> o bien, de un <estado> seguido de un <paquete regla>.
Los <arc> son usados para especificar acciones que deben ejecutarse durante el análisis de un texto, y para especificar el llamado a <paquete regla>.

Un <nombre categoría> indica el nombre de una categoría o clase gramatical. Por medio de los <nombre categoría> se establece cuales clases gramaticales son permitidas para definir el ATN. Las clases gramaticales tratadas en este trabajo son presentadas detalladamente cuando se describe el diccionario del ATN.

Una <acción> indica el nombre de una función. Las acciones se ejecutan durante el análisis de enunciados. Si al ejecutar una <acción> se detecta un error, el proceso de análisis se detiene y se emite el correspondiente mensaje de error. Las acciones a ejecutarse son : **ARREGLA, COMPARA, CONTROLA, CARGA, CARGAO.**
Las mismas se describen detalladamente cuando se presenta la red QUIMICA.
Un <paquete regla> indica el llamado a un paquete de reglas.
Los paquetes de reglas son usados como medio de selección de reglas. Cada paquete está formado por varias reglas que comparten ciertas características.

Un <estado> puede definirse como un 0 o un 1. El estado 0 se utiliza para preceder a una <acción>, e indica que la misma debe ejecutarse para poder continuar con el análisis sintáctico y semántico del texto ingresado. También el <estado> 0 se utiliza para preceder a un llamado a un <paquete regla>. El <estado> 1 se utiliza para preceder a un <nombre subred> e indica que el control debe pasarse a la subgráfica especificada por <nombre subred>.

Se presenta a continuación la gramática general del sistema que fue utilizada para tratar oraciones en castellano sobre química. El formato de la misma responde a la definición del ATN presentado anteriormente.

La gramática general

```
(QUIMICA   ( (PRON_INT1  1  FRA1)
             (PRON_INT2  1  FRA6)
             (VERB_3  0  ARREGLA  1  FRA2)
             (PREP_2  PRON_INT2  SUSCOM_1  PRON_ACUS  0  CONTROLA
              VERB_5  0  COMPARA  VERB_6  ARTDET  0  COMPARA
              SUSCOM_1  0  COMPARA  X  0  CARGA  0  FAMILIA2)
             (VERB_7  0  ARREGLA  SUSCOM_5  PREP_1  X4  0  CARGAO
              PREP_3  ARTDET  0  COMPARA  ADJIND  0  COMPARA
              SUSCOM_1  0  COMPARA  CONE  1  FRA7)
             (VERB_8  0  ARREGLA  ARTDET  ADJIND  0  COMPARA
              SUSCOM_1  0  COMPARA  PREP_4  X2  0  CARGAO  CONEC
```

```
                X2 O CARGAO CONE 1 FRA7)
               (VERB_9 O ARREGLA 1 FRA8) ))

(FRA1      ( (VERB_1 O COMPARA 1 FRA2) ))

(FRA2      ( (ARTDET O COMPARA 1 FRA3) ))

(FRA3      ( (SUSCOM_1 O COMPARA 1 FRA4)
             (ADJCAL_1 O COMPARA SUSCOM_1 O COMPARA
              SUSCOM_2 O COMPARA O CONTROLA PREP_1 ARTDET
              O COMPARA X1 O CARGAO O FAMILIA2)
             (SUSCOM_3 O COMPARA PRON_REL1 1 FRA5)
             (SUSCOM_4 O COMPARA PREP_1 SUSCOM_3 O COMPARA
              PRON_REL1 1 FRA5) ))

(FRA4      ( (X2 O CARGAO O FAMILIA1)
             (ART_CON1 X3 O CARGAO O FAMILIA1) ))

(FRA5      ( (VERB_4 ART_CON2 VERB_6 ARTDET O COMPARA
              SUSCOM_1 O COMPARA X O CARGA CONEC X O
              CARGA O FAMILIA3) ))

(FRA6      ( (SUSCOM_3 1 FRA5)
             (SUSCOM_4 1 FRA5) ))

(FRA7      ( (X O CARGA 1 FRA7)
             (NIL O FAMILIA2) ))

(FRA8      ( (DETIND O COMPARA ARTDET O COMPARA SUSCOM_6
              O COMPARA O CONTROLA ART_CON1 SUSCOM_1 O
              COMPARA X O CARGA O FAMILIA4)
             (ARTDET O COMPARA SUSCOM_6 O COMPARA X4 O
              CARGAO ART_CON1 SUSCOM_1 O COMPARA X O CARGA
              O FAMILIA4) ))
```

Se presenta a continuación una descripción de las acciones utilizadas en esta red.

Las acciones utilizadas

ARREGLA : Esta es una acción que obligatoriamente debe ir después de verbos imperativos. La flexibilidad de la gramática permite que estos verbos puedan ingresarse en 2da y 3era persona, sin embargo esto nos puede llevar a rechazar un texto por no concordar con la sintáxis especificada.
Por ejemplo en el siguiente enunciadó :

" Muestra los elementos alcalinos "

En este enunciado " Muestra " es un verbo en segunda persona y " los " es un artículo en tercera persona del plural. Si luego de estas palabras se aplica una comparación de números, el resultado será negativo ya que los mismos no coinciden.
ARREGLA tiene como objetivo solucionar estos inconvenientes.

COMPARA : Esta es una acción destinada a comparar género, número y persona de los estados analizados. Si **COMPARA** detecta un error se detiene el análisis del enunciado y se da un aviso de error.
Por ejemplo en el siguiente enunciado :

" Cuales son el elementos liquidos ? "

En este enunciado existe un error de concordancia entre el número del verbo (3era. persona del plural) y el número del artículo (3era. persona del singular). **COMPARA** detecta este error y emite el siguiente mensaje :

" Por Favor Revise Género, Número y Persona, Hay un Error en su Oración "

CONTROLA : Esta acción tiene por objetivo evitar conflictos de concordancia de números entre ciertas palabras.
Dada la flexibilidad de la gramática, se puede llegar a tener que comparar, en un determinado momento, el número de una palabra en tercera persona del plural con el número de una palabra en tercera persona del singular..pn 7
Sin embargo, en ciertos casos ésto no significa que exista error en la construcción de la oración.
Por ejemplo en el siguiente enunciado :

" Muestre los principales elementos conductores de la electricidad "

En este enunciado, luego de la palabra " conductores " (sustantivo, 3era. persona del plural) sigue la palabra " de " (preposición, neutra). Al existir una palabra con número neutro se ignora, y la comparación se efectúa con la siguiente palabra, en este caso " la " (artículo, 3era. persona del singular). **CONTROLA** evita que el proceso de análisis se interrumpa, ya que aunque no existe concordancia de números entre estas dos palabras, la oración es sintácticamente correcta.

CARGA : Es una función que tiene por objetivo cargar en una pila los elementos de la tabla periódica que se mencionan en el texto ingresado. Esta pila luego será utilizada en el análisis semántico del mencionado texto.
Por ejemplo en el siguiente enunciado :

" Clasifique los siguientes elementos en líquidos y sólidos : mercurio cesio francio oro plata cobre. "

Durante el análisis la acción CARGA irá almacenado en una pila (pila1), los elementos de la tabla periódica que se encontraron en el texto. Al finalizar el análisis pila1 tendrá la siguiente información :

pila1 : (cobre plata oro francio cesio mercurio)

CARGAO : Es una función que tiene por objetivo cargar en una pila las propiedades de los elementos de la tabla periódica que se mencionan en el texto ingresado. Esta pila luego será útil en el análisis semántico del texto ingresado.
Por ejemplo en el siguiente enunciado :

" Clasifique los siguientes elementos en líquidos y sólidos :
mercurio cesio francio oro plata cobre. "

Durante el análisis la acción CARGAO irá almacenado en una pila (pila2), las propiedades de los elementos de la tabla periódica que va encontrando en el texto. al finalizar el análisis pila2 tendrá la siguiente información :

pila2 : (sólidos líquidos).

El Diccionadrio de datos

Si bien el lenguaje utilizado es restringido, la cantidad de verbos, sustantivos, preposiciones, adjetivos, pronombres y artículos utilizado hace necesario dividir estas categorías gramaticales en subcategorías para poder diferenciar el significado q
Se muestran a continuación las categorías y subcategorías gramaticales a las que pertenecen las palabras utilizadas en este trabajo.

PRONOMBRES

<PRON_INT1> ::= Interrogativo.
Cual / Cuales

<PRON_INT2> ::= Interrogativo.
Qué

<PRON_REL1> ::= Relativo.
que

<PRON_ACUS> ::= Personal.
se

VERBOS

<VERB_3> ::= Imperativo
Dé / Diga / Dá / Dí / Muestre / Muestra / Obtén / Obtenga / Mencione / Menciona / Escribe / Escriba

<VERB_7> ::= Imperativo.
Asigne / Asigna

<VERB_8> ::= Imperativo.
Clasifica / Clasifique / Agrupe / Agrupa.

<VERB_9> ::= Imperativo.
Borre / Borra / Elimine / Elimina

<VERB_1> ::= Indicativo.
son

<VERB_2> ::= Indicativo.
es

<VERB_4> ::= Indicativo.
originan / constituyen / forman

<VERB_5> ::= Indicativo.
puede

<VERB_6> ::= Infinitivo.
combinar / juntar / unir / ligar

SUSTANTIVOS

<SUSCOM_1> ::= Común.
elemento / elementos / átomo / átomos

<SUSCOM_2> ::= Común.
conductor / conductores / transportador
transportadores / transmisor / transmisores

<SUSCOM_3> ::= Común.
fórmula / fórmulas

<SUSCOM_4> ::= Común.
nombre / nombres

<SUSCOM_5> ::= Común.
intervalo / intervalos

<SUSCOM_6> ::= Común.
propiedad / propiedades

PREPOSICIONES

<PREP_1> ::= Separable.
de

<PREP_2> ::= Separable.
Con

<PREP_3> ::= Separable.
a

<PREP_4> ::= Separable.
en

DETERMINANTES

<DETIND> ::= Indefinido.
todo / toda / todos / todas

<ARTDET> ::= Artículo Determinado.
el / la / los /las

<ART_CON1> ::= Artículo Contracto.
del

<ART_CON2> ::= Artículo Contracto.
al

ADJETIVOS

<ADJCAL_1> ::= Calificativo.
principal / principales / mejor / mejores

<ADJIND> ::= Indefinido.
siguiente / siguientes

CONECTIVOS

<CONEC> ::= Conjunción de Coordinación.
y

<CONE> ::= Conjunción de Coordinación.
:

Fue necesario definir además, categorías gramaticales que tuvieran como instancias a los elementos de la tabla periódica. El nombre dado a cada una de estas categorías fue elegido arbitrariamente.
La definición sintáctica y semántica de las mismas se dan a continuación :

<X>::= sodio / plata / oro / magnesio / cobre / berilio.....

Un <x> permite tomar como instancias a los nombres de elementos de la tabla periódica.

<X1> ::= electricidad / calor

Un <X1> permite considerar como instancias a las propiedades transmisibles de elementos de la tabla periódica.

<X2> ::= líquidos / sólidos / alcalinos / alcalinotérreos.....

Un <X2> toma como instancias atributos de los elementos de la tabla periódica.

<X3> ::= primer_período / grupo_1 / grupo_2b / .....

Un <X3> se utiliza para denotar propiedades de pertenencia de los elementos de la tabla periódica. los que necesitan de el artículo contracto " del " para poder referirlos.

<X4> ::= NA / PE / PF / DEN / PA / EO / SIM / EP / CE / CT

Un <X4> se utiliza para denotar atributos generales de los elementos de la tabla periódica. Hemos considerado a los siguientes :

NA	⟶	Número Atómico
PE	⟶	Punto de Ebullición
PF	⟶	Punto de Fusión
DEN	⟶	Densidad
PA	⟶	Peso Atómico
EO	⟶	Estado de Oxidación
SIM	⟶	Símbolo
EP	⟶	Electronegatividad de Pawling
CE	⟶	Conductividad Eléctrica
CT	⟶	Conductividad Térmica

Se presenta a continuación un ejemplo, para ilustrar al lector como efectúa el ATN los análisis sintáctico y semántico. Consideremos la oración :

Muestre los elementos líquidos.

El análisis se efectúa palabra por palabra. El intérprete del ATN busca en el diccionario de datos las características gramaticales de " Muestre ", que son :

Clase Gramatical : VERB_3 / Modo : Imperativo
Tiempo : Presente / Persona : 3era. singular

Con la clase gramatical <VERB_3> se comienza a recorrer la red de transición. Cuando <VERB_3> se instancia con el primer elemento de un arco, el análisis continúa a través de dicho arco.
En este caso, se trata del arco :

(VERB_3 0 **ARREGLA** 1 FRA2)

El análisis continúa con el <estado> 0, que indica que a su derecha existe una <acción> (**ARREGLA**) que debe ejecutarse. Esta <acción> permite que la palabra siguiente a " Muestre " pueda estar tanto en 3era. persona del singular, como en 3era. persona del plural. Posteriormente se indica por medio de un <estado> 1, que el control debe pasarse a la <subred> FRA2.

El análisis continúa con la siguiente palabra del enunciado, en este caso " los ", cuyas características gramaticales son :

Clase gramatical : ARTDET / Género : M (masculino)
Número : P (plural) / Persona : 3era. plural

Con <ARTDET> se comienza a recorrer la <subred> FRA2 hasta detectar el <arco> que contiene esta <categoría gramatical>.
En este caso se trata del arco :

(ARTDET 0 **COMPARA** 1 FRA3)

El análisis continúa con el <estado> 0, que indica que a su derecha existe una acción que debe ejecutarse : **COMPARA**.
La parte final del arco indica, por medio del <estado> 1, que el control debe pasarse a la subred FRA3.

Se analiza a continuación la siguiente palabra del enunciado, en este caso : elementos. Sus características gramaticales son :

Clase gramatical : SUSCOM_1 / Género : M (masculino)
Número : P (plural) / Persona : 3era. plural

Con <SUSCOM_1> se comienza a recorrer la <subred> FRA3. El <arco> que comienza con esta <categoría gramatical> es el siguiente :

(SUSCOM_1 O **COMPARA** 1 FRA4)

El análisis continúa con el <estado> 0 y la <acción> **COMPARA**. Posteriormente se detecta el <estado> 1 y se transfiere el control a la <subred> FRA4.
Se analiza la siguiente palabra del enunciado : líquidos. Sus características gramaticales son :

Clase gramatical : X / Género : Masculino
Número : P (plural) / Persona : 3era. plural

Con <X2> se recorre la <subred> FRA4. El <arco> que comienza con la <categoría gramatical> es el siguiente.

(X2 O **CARGAO** O **FAMILIA1**)

El análisis continúa con el <estado> 0 y la <acción> **CARGAO**. Esta <acción> tiene el objetivo de cargar la palabra " líquidos " en una variable de trabajo. Posteriormente el análisis continúa con el <estado> 0 y se detecta un llamado a un <paquete regla>. En este paquete se hace uso de la variable de trabajo y según su contenido se decidirá que regla debe de aplicarse. Para este ejemplo en particular se aplica la regla REGLA4.
REGLA4 nos permite determinar qué elementos de la tabla periódica son líquidos.
Una vez que se ejecuta esta regla, el sistema presenta los resultados obtenidos.

Para que finalmente el sistema pueda proporcionar resultados, el enunciado debe estar bien formado. Se considera bien formado, si durante los análisis sintáctico y semántico no se detectan errores y se llega finalmente al llamado de un <paquete regla>.

3 - EL MECANISMO DE INFERENCIA DEL SISTEMA

La parte medular en el sistema es el mecanismo de inferencia del mismo. Este opera con : una memoria de trabajo, una base de conocimientos, una base de hechos y una máquina de inferencia que aplica las reglas pertinentes a la resolución del problema en curso.

La memoria de trabajo contiene la información (datos o conclusiones) inferida hasta el momento.

La base de conocimientos está constituída por el conocimiento específico y procedimental acerca de la clase de problemas en los que el sistema es experto.

La máquina de inferencia es el conjunto de programas y estructuras de control que hacen deducciones a partir de la selección, interpretación y aplicación de las reglas a la información presente en la memoria de trabajo.

Para que una regla pueda aplicarse es necesario que parte del contenido de la memoria de trabajo coincida (sea igual o pueda hacerse igual mediante sustitución) con parte de la regla.
En el sistema existen dos formas básicas de interpretación de las reglas : el encadenamiento hacia adelante y el encadenamiento hacia atrás.

El Lenguaje de Escritura de las Reglas

A continuación se presenta la sintaxis del lenguaje de escritura de reglas especificado en la Forma Normal de Backus.

<paquete regla> ::= <regla>$^{+}$

<regla> ::= (DEFUN <nombre regla> (<arg1> [<arg2>])
<cuerpo regla>

<cuerpo regla> ::= ((<oper> <premisa>$^{+}$) <alternativa1>
<alternativa2> /
(<predicado> <alternativa1> <alternativa2>

<premisa> ::= (<oper> <predicado> <predicado>$^{+}$) /
<predicado> <predicado>$^{+}$

<alternativa1> ::= <cuerpo regla> / <acción>)

<alternativa2> ::= <cuerpo regla> / <acción>)

<oper> ::= AND / OR

<predicado> ::= (MIEMBRO <op1> <op2>) /
(IGUAL <op1> <op2>) /
(MAYOR <op1> <op2>) /
(MENOR <op1> <op2>) /
(NEGATIVO <op1> <op2>) /
(POSITIVO <op1> <op2>)

<acción> ::= (REGRESA <nombre regla>) /
(AGREGA <nombre regla>) /
(ALMA_MIN <nombre regla>) /
(ALMA_MAX <nombre regla>)

<nombre regla> ::= REGLA1 / REGLA4 / REGLA5 /...../REGLAn

<op1> ::= NA / PE / PF / DEN / PA / EO / SIM / EP /
CE / CT / <nombre variable>

<op2> ::= <números reales> / <nombre variable>

<nombre variable> ::= <identificador>

<identificador> ::= <caracteres alfabéticos>$^{+}$

<arg1> ::= LST

<arg2> ::= SEN / ATOMO

Definición Semántica del Lenguaje

En la primera línea de la definición sintáctica , se establece que un <paquete regla> está formado por una o más reglas. Una <regla> está constituída por una palabra terminal DEFUN, un <nombre regla>, un <arg1> seguido opcionalmente de un <arg2> y finalmente de un <cuerpo regla>. De esta manera se establece como debe formarse una regla, de tal manera que la misma pueda ser reconocida por el intérprete.

En el <cuerpo regla> se dan las especificaciones que establecen qué hace la regla. Es aquí donde el usuario, por medio de predicados y operadores, establece el objetivo de la regla.
Un <cuerpo regla> puede estar constituído de dos maneras. La primera establece que un <cuerpo regla> está formado por dos paréntesis abierto , un <oper>, una o más ocurrencias de una <premisa>, un paréntesis cerrado , una <alternativa1> y una <alternativa2>.
La segunda alternativa establece que un <cuerpo regla> está formado por un paréntesis abierto , un <predicado>, una <alternativa1> y una <alternativa2>.

Una <premisa> está formada por un <oper>, un <predicado>,seguido de una o más ocurrencias de otro <predicado>. También una <premisa> puede estar formada por un <predicado> seguida de una o más ocurrencias de otro <predicado>.

Una <alternativa1> está formada por un <cuerpo regla>, o bien por una <acción> seguida de un paréntesis cerrado.
Una <alternativa2> está formada por un <cuerpo regla>, o bien por una <acción> seguida de un paréntesis cerrado.
La <alternativa1> se ejecuta cuando en el cuerpo de la regla resulta verdadero el cumplimiento de las premisas o predicados, en caso contrario se ejecuta la <alternativa2>.

Un <oper> está formado por un símbolo terminal que indica un operador del tipo lógico, como AND (conjunción) y OR (disyunción).

Un <predicado> está formado por un un nombre nemotécnico que indica la relación existente entre los argumentos <op1> y <op2>. Los predicados definidos por las reglas tratadas en este sistema son : MIEMBRO, IGUAL, MAYOR, MENOR, NEGATIVO y POSITIVO.

MIEMBRO : Se utiliza para verificar si <op1> pertenece a <op2>. Cuando una ocurrencia de <op1> aparece en <op2>, el valor de verdad de este predicado es TRUE. En caso contrario es FALSE.

IGUAL : Se utiliza para comparar <op1> con <op2>. Si los operandos son iguales, entonces el valor de verdad del predicado es TRUE. En caso contrario es FALSE.

MAYOR : Se utiliza para comparar <op1> con <op2>. Si <op1> es mayor que <op2>, entonces el valor del predicado es TRUE. En caso contrario es FALSE.

MENOR : Se utiliza para comparar <op1> con <op2>. Si <op1> es menor que <op2>, entonces el valor de verdad del predicado es TRUE. En caso contrario es FALSE.

NEGATIVO : Se utiliza para verificar si <op1> tiene en <op2> algún valor negativo. En caso afirmativo, el valor de verdad del predicado es TRUE. En caso contrario es FALSE.

POSITIVO : Se utiliza para verificar si <op1> tiene en <op2> algún valor positivo. En caso afirmativo, el valor de verdad del predicado es TRUE. En caso contrario es FALSE.

Una <acción> define una función. Las funciones definidas para las reglas tratadas por este sistema son : REGRESA, AGREGA, INICIA, ALMA_MIN, ALMA_MAX.

REGRESA : Se usa para provocar la bifurcación del control del sistema hacia la regla especificada por <nombre regla>. Cuando <nombre regla> especifica la regla en la cual aparece REGRESA, se produce el llamado recursivo a la regla.

AGREGA : Se usa para agregar un elemento a una lista de resultados. Luego pasa el control del sistema a la regla especificada por <nombre regla>. Implicitamente ejecuta un REGRESA.

ALMA_MIN : Se usa para asignar un valor mínimo a una variable. Luego se transfiere el control del sistema a la regla especificada por <nombre regla>.

ALMA_MAX : Se usa para asignar un valor máximo a una variable. Luego se transfiere el control del sistema a la regla especificada por <nombre regla>.

Un <op1> está formado por símbolos terminales, como NA (número atómico), PE (punto de ebullición), DEN (densidad) o por un <nombre variable>.
Un <op2> está formado por cualquier número real o por un <nombre variable>.

Un <arg1> se define como una LST. Una LST es el nombre de una variable, que recibirá como valor una lista de elementos sobre los cuales se quiere aplicar una regla.
Un <arg2> se define como SEN o ATOMO. La variable SEN se utiliza como bandera. Un ATOMO es el nombre de una variable, que recibirá como valor un elemento de la tabla periódica sobre el cual se quiere aplicar una regla.

Los siguientes son ejemplos de reglas de inferencia del sistema, definidas siguiendo la sintáxis del lenguaje de escritura de reglas presentado anteriormente. Con el objeto de que el lector entienda el significado de las mismas precede a cada una de ellas un texto en lenguaje natural.

Son líquidos, los elementos de la tabla periódica, cuyo punto de fusión (PF) está comprendido entre los valores 234.279 y 302.901

```
(DEFUN REGLA4 (LST)
((AND
      (MAYOR PF 234.279)
      (MENOR PF 302.901) ) (AGREGA REGLA4) )
(REGRESA REGLA4) )
```

Son gaseosos, los elementos de la tabla periódica, cuyo punto de ebullición (PE) está comprendido entre los valores 3.99 y 240.

```
(DEFUN REGLA5 (LST)
((AND
      (MAYOR PE 3.99)
      (MENOR PE 240.01) ) (AGREGA REGLA5) )
(REGRESA REGLA5) )
```

4 - RESULTADOS OBTENIDOS POR EL SISTEMA

Los siguientes son algunos ejemplos de resultados obtenidos pore el sistema.
Cabe aclarar que los textos que ingresa el usuario estarán en negritas, precedidas por los símbolos U>, en tanto que las respuestas del sistema en tipo stándar, precedidas por los símbolos S>.

Errores que detecta el Analizador Sintáctico

U>
(Cuales son los elemento gaseosos ?)

S>
POR FAVOR VERIFIQUE GENERO, NUMERO Y PERSONA, HAY UN ERROR EN SU ORACION.

NOTA : La palabra " elemento " está en singular y debe estar en plural.

U>
(Cuales son elementos gaseosos ?)

S>
LA ORACION INGRESADA NO CONCUERDA CON LA GRAMATICA, POR FAVOR REVISE SU ORACION.

Nota : La oración ingresada no concuerda con la gramática, ya que la misma exige que después del verbo " son " debe ir un articulo : los.

U>
(Muestre el principal elemento conductor de la electricidad.)

S>
FALTA LA PALABRA SOLICITADA Y UD. HA DECIDIDO NO INGRESARLA

NOTA : Esto ocurre cuando el analizador detecta una palabra desconocida en el diccionario de datos, le dice al usuario que la ingrese y éste decide no hacerlo. Supongamos que " el " es una palabra que no se encuentra en el diccionario de datos, entonces el sistema despliega en la pantalla, el siguiente mensaje :

DESEA INGRESAR LA PALABRA SOLICITADA " el " : CS / ND ?

Si el usuario elige la opción N, el sistema muestra el mensaje presentado anteriormente.
Si el usuario elige la opción S, el sitema le pedirá las características gramaticales de la palabra desconocida. Luego incorporará la nueva palabra y sus características gramaticales al diccionario de datos.
Con esta opción, los análisis sintáctico y semántico sobre el enunciado no se interrumpen y el diccionario aumenta su volumen.

Acciones que modifican la Base de Hechos del Sistema

U>
(Borre el elemento oro.)

S>
O. K.

NOTA : Cuando se borra o elimina un elemento, se borran también todas sus propiedades.

U>
(Elimina todas las propiedades del elemento plata.)

S>
O. K

U>
(Borra la propiedad PF del elemento magnesio.)

S>
O. K

U>
(ARGON NA 18 PE 87$30 PF 83$81 DEN 1$784 PA 39$948 SIM Ar CT 0$0001772 NIL) <RETURN>

S>
O. K

NOTA : Esta declaración se utiliza para incrementar la base de hechos del siveces lo desee el usuario.

Información que obtiene el usuario por medio de consultas

U>
(Muestre los elementos alcalinos)

S>

1 - Litio	2 - Sodio
3 - Potasio	4 - Rubidio
5 - Cesio	6 - Francio

U>
(Cuales son los elementos alcalinotwrreos ?)

S>

1 - Berilio	2 - Magnesio
3 - Calcio	4 - Estroncio
5 - Bario	6 - Radio

U>
(Mencione los elementos lhquidos.)

S>

1 - Cesio	2 - Francio
3 - Mercurio	4 - Galio
5 - Bromo	

U>
(Obtenga los elementos metaloides.)

S>

1 - Boro	2 - Silicio
3 - Germanio	4 - Antimonio
5 - Bismuto	6 - Telurio
7 - Polonio	8 -Mercurio

NOTA: Es de destacar que las reglas fueron escritas en función de ciertos conceptos típicos de química, en los cuales me he basado. En este caso se establece que un elemento es metaloide si su electronegatividad está comprendida entre los valores 1.8 y 2.1 y su conductividad eléctrica es menor que 0.03.
Como respuesta a esta órden aparece el elemento mercurio, ya que su electronegatividad es igual a 2 y su conductividad eléctrica es igual 0.0104, sin embargo este elemento no es un metaloide.

U>
(Cual es el principal elemento conductor de la electricidad ?)

S>

EI principal elemento es	: plata
Su conductividad eléctrica es	: 0.630 10**6/ohm*cm

NOTA : Es de destacar que este resultado surge de la aplicación de una regla dinámica, es decir que identifica a " principal " como máximo y luego se busca el elemento cuya conductividad eléctrica sea más alta.
Si por ejemplo borramos el elemento plata y nuevamente aplicamos la pregunta, la respuesta será :

El principal elemento es : cobre
Su conductividad eléctrica es : 0.596 10**6/ohm*cm

U>
(Muestre los principales elementos conductores de el calor.)

S>

Elemento		Conductividad en 10^6 / ohm * cm.
1 -	plata ---->	4.19
2 -	cobre ---->	4.01
3 -	oro ---->	3.17
4 -	aluminio ---->	2.37
5 -	berilio ---->	2.00
6 -	calcio ---->	2.00
7 -	tungsteno ---->	1.74
8 -	manganeso ---->	1.56
9 -	rodio ---->	1.50
10 -	silicio ---->	1.48

NOTA : Es de destacar que en este caso se asocia la palabra " principales " con una lista de elementos que deben cumplir una determinada condición.

U>
(Clasifique los siguientes elementos en alcalinos y alcalinotérreos : litio berilio calcio rubidio cesio oro plata bario.)

S>
Alcalinos : litio - rubidio - cesio
Alcalinotérreos : berilio - calcio - bario

NOTA : En este caso estamos en presencia de una meta_regla. Es decir que luego de analizar sintáticamente la orden hacemos un reordenamiento de la información ingresada.
En este caso la meta_regla actúa como una vía de selección para el conflicto de resolución, detectando y aplicando la regla pertinente.

U>
(Asigne intervalos de PE a los siguientes elementos : escandio titanio vanadio cromo manganeso hierro cobalto níquel cobre cinc.)

S>
El valor inferior es : cinc ----> 1180 K.
El valor superior es : vanadio ----> 3682 K.

NOTA : Este tipo de órdenes son muy útiles cuando queremos establecer los valores mínimos y máximos de una determinada propiedad. Por medio de estas órdenes podemos establecer cuales son los principales y cuales los menores conductores de calor y electricidad, por ejemplo.

U>
(Con qué elementos se puede combinar el elemento sodio ?)

S>

1 -	carbono	2 -	nitrógeno
3 -	oxígeno	4 -	flúor
5 -	fósforo	6 -	azufre
7 -	cloro	8 -	arsénico
9 -	selenio	10 -	bromo
11 -	antimonio	12 -	telurio
13 -	iodo	14 -	renio
15 -	astato		

NOTA : Para este tipo de cuestionamientos se da como respuesta todos los elementos que satisfacen las hipótesis de la regla aplicada. Cabe mencionar que no importa la probabilidad con que estas combinaciones puedan ocurrir. Así por ejemplo tenemos que el flúor con el sodio tienen una probabilidad del 90 % de ocurrir la ligación, en tanto que el renio con el sodio tienen una probabilidad del 19 % de ocurrir la ligación. Estas probabilidades tendrán una mayor credibilidad cuando se trate de elementos con un determinado carácter iónico.

U>
(Muestre las fórmulas que originan al combinarse los elementos cloro y sodio.)

S>
Fórmula : Cl1Na1 ClNa
Probabilidad : 70 %

U>
(Muestre los nombres de fórmulas que originan al combinarse los elementos flúor y manganeso.)

S>
Nombres : Fluoruro de Manganeso VII
Fluoruro de Manganeso VI
Fluoruro de Manganeso II
Fluoruro de Manganeso IV
Fluoruro de Manganeso III

Probabilidad : 76 %

NOTA : Cabe mencionar que se da esta notación por ser la moderna y la más comunmente utilizada por los químicos en la actualidad.

5 - CONCLUSIONES

1- En este trabajo los datos se encuentran independientes del programa. Esta característica se da en las dos fases del sistema.

En la primera fase (procesamiento de lenguaje natural), el intérprete está separado de la gramática y del diccionario. En la segunda fase (mecanismos de inferencia), la máquina de inferencia está separada de la base de conocimientos y de la base de hechos.
Si se cambia la gramática, la base de conocimientos y la base de hechos del sistema, el mismo puede operar en otras áreas del saber.

2- El procesador de lenguaje natural (ATN) es muy general y la gramática utilizada para tratar textos de química es muy diversa, lo que permite al usuario emplear diferentes modos de expresar sus enunciados.

3- La interfase de lenguaje natural facilita la interacción del hombre con la máquina. Esta interfase hace más amigable al sistema. Este guía al usuario por medio de menús y mensajes.

4- El sistema puede arrancar sin ningún dato en el diccionario y en la base de hechos, y a medida que opera puede ir incorporando información. Por lo tanto, se puede afirmar que es dinámico.

5- El sistema está capacitado para aceptar nuevas reglas de inferencia. El usuario, haciendo uso del lenguaje de escritura de reglas, puede definirlas de acuerdo a sus necesidades.

BIBLIOGRAFIA

[1] Cairó, Osvaldo. " Un Sistema Basado en Conocimientos ". Tesis de Maestría. Computación. Dpto de Ing. Eléctrica. CINVESTAV - IPN, México, 1987.

[2] Winograd, Terry. " Understanding Natural Language ". Academic press, New York, 1972.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

PROSEDEM

PROtotipo de Sistema Experto de Diagnostico de Enfermedades Metabolicas en ganado vacuno.

Frassoni, Maria Sandra.
Vigil, Nestor Omar.

ABSTRACT

El presente trabajo muestra el desarrollo y construcción de un prototipo de Sistema Experto de Diagnóstico Veterinario.

El problema se define como diagnosticar y tratar Hipocuprosis, Afosforosis y Ostertagiasis Gástrica, la ausencia de éstas o sus posibles asociaciones, en establecimientos ganaderos que pertenecen a la Cuenca del Salado, provincia de Bs.As. Estos deben dedicarse a la cría extensiva de ganado bovino de razas de origen europeo, con pastoreo directo.

El sistema básicamente tiene la característica de hacer que el diagnóstico sea más sistemático, permitir que una mayor cantidad de médicos veterinarios puedan realizarlo más rápida y precisamente y recomendar tratamientos alternativos.

La metodología de Sistemas Expertos no es hoy en día del todo revolucionaria. Sin embargo, sin su presencia el tratamiento del conocimiento incierto y de las heurísticas existentes en las reglas que simulan el razonamiento del experto como así también la naturaleza intrínseca del conocimiento involucrado, que procede incrementalmente de tareas simples a tareas complejas, serían en extremo costosos en tiempo y esfuerzo.

1. MOTIVACION

El desafío que representaba, al momento de comenzar este trabajo, un tema tan reciente como la Inteligencia Artificial y, en particular, los Sistemas Expertos se constituyó en el principal estímulo que condujo a producirlo.

2. ELECCION DEL PROBLEMA A RESOLVER

Desde un primer momento se piensa en un problema de diagnóstico ya que existen numerosos sistemas de ese tipo con resultados satisfactorios.

Encontrar el experto adecuado para el trabajo no es una tarea sencilla. En el desarrollo de un Sistema Experto que va a ser destinado para producción, la elección del mismo está condicionada a la del dominio del problema. Debido a que el énfasis de este trabajo está puesta en la etapa de desarrollo más que en la etapa de producción, se subordina la elección del tema a la disponibilidad del experto, quien finalmente es un médico veterinario especialista en enfermedades metabólicas.

El tema consiste en encontrar un grupo de enfermedades con síntomas comunes y difícilmente distinguibles entre sí que cumplan los siguientes requisitos:

* Que sean ampliamente conocidas por él.
* Que hayan sido tratadas (resueltas muchas veces).
* Que su diagnóstico requiera un considerable esfuerzo o elaboración.
* Que el tema tenga un grado de dificultad tal que una persona que no tenga la experiencia suficiente no pueda resolverlo o le resulte muy difícil.

Luego de un análisis conjunto de las distintas propuestas sugeridas, se elige trabajar con tres enfermedades sobre las cuales el experto posee un amplio dominio : Hipocuprosis, Afosforosis y Ostertagiasis Gástrica.

3. INTRODUCCION AL DOMINIO DEL PROYECTO

3.1 DESCRIPCION DEL PROBLEMA

Las enfermedades que se tratan son propias del ganado vacuno. Están causadas básicamente por problemas nutricionales (originados, en su mayoría, por la composición química del suelo) o por parásitos encontrados en la región. Son frecuentes en una zona bien delimitada dentro de la provincia de Buenos Aires: la Cuenca del Salado.

Afectan en forma adversa tanto a la producción de carnes como a la reproducción de animales, y son causadas por la

carencia de minerales en los suelos, el abundante crecimiento de la pastura (en determinadas épocas del año), la deficiencia en la alimentación, la baja resistencia de las razas consideradas, la mala calidad y baja frecuencia en la vacunación, la alta concentración de animales y el mal manejo del pastoreo.

3.2. COMO SE TRATA EL PROBLEMA ACTUALMENTE ?

El médico veterinario concurre al campo para identificar los síntomas presentes. Si surgen dudas en el diagnóstico, consulta en forma personal o telefónicamente al experto en el tema que forma parte del Instituto Nacional de Tecnología Agropecuaria. Este último puede requerir diversos análisis, los cuales resultan costosos en tiempo y dinero. Si no se puede contar con ellos, el diagnóstico se realiza con cierto margen de error.

3.3. VALOR DE LA SOLUCION.

Teniendo en cuenta que en promedio los animales pierden hasta 1/2 kg. de peso diario y que el tiempo aproximado para detectar esta disminución es de unos 30 días, considerando un rodeo de unas 200 cabezas de ganado, el lucro cesante puede estimarse aproximadamente en 3000 kg de carne. A esto debe agregarse el costo para que el animal recupere su estado normal. La situación se agrava más aún si se considera que estas enfermedades pueden conducir a la muerte.

4. CARACTERISTICAS QUE TIENE EL PROBLEMA PARA SER DESARROLLADO POR LA METODOLOGIA DE SISTEMAS EXPERTOS.

En primer lugar se determina que el sistema es posible, justificado y apropiado.

POSIBLE porque existe un experto genuino en el área del problema capaz de articular sus métodos, cuyas tareas no requieren en su mayoría de un contacto físico con los animales (y aquellas que sí lo requieren pueden ser descriptas en términos precisos).

JUSTIFICADO porque los cálculos estimados del valor de la solución arrojan cifras suficientemente significativas como para ser tenidas en cuenta; además no abunda la experticia humana en el país. Pero por sobre todo, el interés reside en la investigación de una metodología novedosa

APROPIADO porque la naturaleza del problema exige un manejo heurístico de símbolos, su complejidad no es demasiado reducida como para no justificar el costo y el esfuerzo empleado, y el rango que abarca es lo suficientemente reducido como para que éste sea manejable.

5. ALCANCES DEL SISTEMA.

5.1. DEFINICION DEL PROBLEMA.

El problema se define como: diagnosticar y tratar Hipocuprosis, Afosforosis y Ostertagiasis Gástrica, la ausencia de éstas o sus posibles asociaciones, en establecimientos ganaderos que pertenecen a la Cuenca del Salado. Estos deben dedicarse a la cría extensiva de ganado bovino de razas de origen europeo y con pastoreo directo.

Resulta necesario acotar el dominio del problema de modo que la complejidad y el rango del mismo sean los adecuados para lograr que éste resulte manejable. Se elige limitar los alcances del dominio del sistema a aquellos establecimientos que se dediquen a la cria extensiva porque, al ser las enfermedades principalmente de origen nutricional, sus síntomas se observan más fácilmente en animales pequeños (terneros, novillos). También se acota el dominio del problema a tener en cuenta sólo razas de origen europeo ya que son menos resistentes a este tipo de enfermedades, y las más abundantes de la región. Por las causas que provocan estas enfermedades es evidente que sólo podrán observarse en aquellos establecimientos que practiquen pastoreo directo, es decir, que se alimenten directamente de lo que el suelo provee sin tener en cuenta ningun tipo de suplementación mineral.

5.2 CARACTERISTICAS DEL SISTEMA. USUARIOS Y UBICACION.

Este sistema aumenta la precisión del diagnóstico a medida que recibe más datos, recomienda tratamientos alternativos entre los cuales el usario del sistema puede elegir, hace que el diagnóstico de estas enfermedades sea más sistemático y permite a una mayor cantidad de médicos veterinarios realizar un diagnóstico más rápido y preciso.

Los potenciales usuarios del sistema son médicos veterinarios que carezcan de la experencia necesaria para realizar el diagnóstico preciso. Eventualmente también lo puede consultar cualquier persona que posea conocimiento en el manejo de este tipo de ganado.

Los lugares más apropiados para la ubicación del sistema son instituciones públicas de asesoramiento a los productores ganaderos, como por ejemplo el Instituto Nacional de Tecnología Agropecuaria, así como centros universitarios donde se estudie este tema.

6. ELECCION DE LA HERRAMIENTA

Para la elección de la herramienta se deben tener en cuenta algunos aspectos básicos como experiencia del grupo en el desarrollo de Sistemas Expertos, facilidades de apoyo y

confiabilidad que brinda la herramienta, y características de la tarea.

Tanto las características del problema como las de la aplicación sugieren características necesarias para el Sistema Experto, las que a su vez sugerirán características particulares de la herramienta.

CARACTERISTICAS DEL PROBLEMA

- Espacio reducido de soluciones.
- Asociaciones heurísticas.
- Inseguridad en las respuestas.
- Soluciones basadas en suma de eventos inciertos.

CARACTERISTICAS DE LA HERRAMIENTA

- Backward chaining.
- Reglas.
- Manejo de incerteza.
- Acumulación de incertezas.

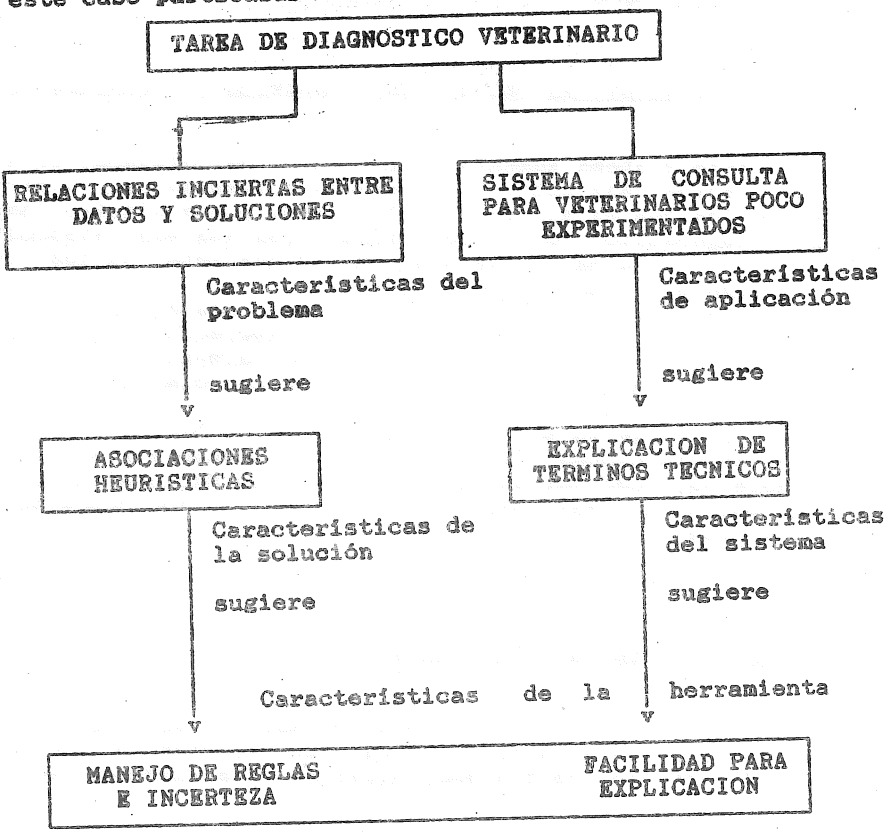
CARACTERISTICAS DE LA APLICACION

-Grupo de desarrollo del Sistema Experto principiantes en la metodología.
-Usuarios no expertos.

CARACTERISTICAS DE LA HERRAMIENTA

-Lenguaje de Ingeniería del Conocimiento y Mecanismo de depuración.
-Mecanismo de explicación.

En este caso particular:



Se tiene la oportunidad de trabajar con lenguajes de programación como LISP y PROLOG, o con otras herramientas de construcción de Sistemas Expertos como INSIGHT2 PLUS y OPS5.

Luego de un análisis comparativo, la herramienta más adecuada es INSIGHT2 PLUS porque:

* Es un lenguaje de Ingeniería del Conocimiento.
* Tiene control de inferencia Backward Chaining.
* La representación del conocimiento se lleva a cabo mediante reglas.
* Cuenta con manejo de incerteza.
* Cuenta con facilidades de explicación.
* Cuenta con facilidades de depuración.

INSIGHT2 PLUS reúne básicamente todas las características necesarias para este problema, excepto la acumulación de incertezas , que se subsana mediante el desarrollo de una rutina externa.

7. CAPTURA DEL CONOCIMIENTO

7.1. ASPECTOS GENERALES

La limitación más seria en el desarrollo de este sistema (como así también en el de cualquier Sistema Experto) es la captura del conocimiento, ya que no existen actualmente metodologías ni herramientas que sean capaces de eliminar o facilitar esta tarea. Por esta causa, adquirir el conocimiento del experto e incorporarlo a la base de conocimiento insume un tiempo prolongado.

En el proceso de captura del conocimiento no se puede ignorar la naturaleza intrínseca del conocimiento del experto, el cual parece estar compilado y reducido a una forma mínima y eficiente que lo hace más difícil de extraer.

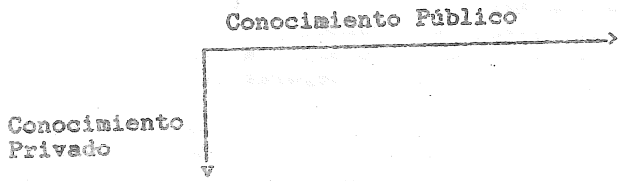
Los expertos usualmente no pueden articular con facilidad cada paso y, lo que es peor, pueden no tener conocimiento de los pasos individuales tomados para alcanzar la solución. Al explicar su conclusión repiten sólo los pasos principales dejando de lado a menudo los más pequeños, los cuales parecen obvios (Conocer qué considerar básico y relevante y no requerir una posterior reevaluación es lo que hace que una persona sea un experto). Por ende, generalmente cuanto más competentes son, menos capaces son de describir el conocimiento que usan para resolver problemas. Aún peor, los estudios han mostrado que cuando los expertos intentan explicar cómo alcanzaron una conclusión a menudo construyen líneas de razonamiento posibles que sólo muestran semejanza con su actividad real de resolver problemas.

Para lograr obtener el verdadero proceso de razonamiento se debe ahondar en la estructura del conocimiento del experto, por lo que es conveniente trabajar solamente con un experto hasta construir el prototipo del sistema ya que de otra forma el resultado es un análisis superficial de las técnicas de resolución de problemas.

La fuente principal de conocimiento en un Sistema Experto

es el experto mismo. Su pericia proviene de libros de texto y reportes (conocimiento público), como así también del estudio de casos, datos empíricos y experiencia personal (conocimiento privado).

Es así que el conocimiento se debe obtener a través de una interacción directa con el experto mediante una serie prolongada y continua de entrevistas, atravesando el conocimiento en forma diagonal.



7.1. COMO ENTREVISTAR AL EXPERTO ?

Existen estudios psicológicos y científicos referidos a las técnicas de resolución de problemas que utilizan los expertos. Estos son el métodos intuitivo y el observacional.

El método observacional se basa en observar al experto en la resolución de problemas concretos sin interrumpirlo con preguntas o comentarios, limitándose a analizar una transcripción de la sesión luego de finalizada.

El método intuitivo se basa en que el ingeniero del conocimiento estudie e interactúe tanto con el experto como con la bibliografía para familiarizarse con el tema, volviéndose de esta forma un pseudoexperto. Así, el ingeniero representa el conocimiento que luego es incorporado en un programa y controlado por otros expertos.

Se puede comprobar que lo más conveniente es usar los dos enfoques, combinándolos en las entrevistas con el experto. Durante las mismas él debe extenderse en sus relatos sobre el tema, en cuyo caso el ingeniero toma una actitud pasiva. Sin embargo, cuando se considera apropiado, se realizan preguntas relevantes, se sugieren posibles razonamientos y se hipotetizan conceptos y reglas, adquiriendo en esta oportunidad un rol activo.

Entonces, al tratar de ayudar al experto a describir su conocimiento y estrategias de inferencia se combina psicología con técnicas de programación simbólica para el desarrollo de Sistemas Expertos.

Dado que en un principio no quedan muy claros qué aspectos son realmente útiles para la construcción del sistema, se adopta la actitud de resolver primeramente problemas particulares (porque rara vez resulta efectivo requerir del experto directamente sus reglas o métodos para resolver un tipo particular de problema).

Durante las entrevistas, el experto introduce nuevos términos, conceptos y relaciones. Cuando esto sucede, se le pide que los defina y relacione con los ya existentes. Esto puede eventualmente requerir una redefinición o combinación de conceptos ya presentes en la base de conocimiento.

Para tratar de capturar el conocimiento se utiliza un

procedimiento basado en la técnica Depth-first. Este consiste en preguntar al experto cómo fue alcanzada una conclusión, cómo fueron alcanzadas las conclusiones sobre las cuales ésta se basa y así siguiendo, hasta que finaliza la explicación.

Luego de formular algunas reglas específicas relacionadas con problemas particulares, se tratá de generalizarlas pero sin dejar de contribuir a la solución en el contexto de los problemas particulares.

Al mismo tiempo, para constatar la consistencia y completitud del conocimiento contenido en las reglas, se testen fragmentos del sistema ya implementado confrontándolos con la opinión del experto.

La forma de trabajo (pasar de casos particulares a una generalización de ellos) constituye la estructura de la secuencia en la captura del conocimiento.

7.2. SUGERENCIAS EN LA CAPTURA DEL CONOCIMIENTO

* Pedir anticipadamente todo aquello que sea necesario para el desarrollo de la próxima entrevista: Aunque esto parezca obvio en el caso de la bibliografía o cualquier otro material, no lo es tanto en aquellas ocasiones que requieren una elaboración más refinada por parte del experto, como lo es la preparación de casos particulares de las enfermedades y de los tratamientos adecuados.

* Grabar las entrevistas: Esto es muy importante pues da la oportunidad de realizar, a posteriori, un análisis más minucioso de los relatos del experto.

* Preparar cuestionarios ordenados temáticamente: Es de gran utilidad ya que permite al experto mantener un orden en sus explicaciones y al grupo de trabajo una secuencia lógica en la transcripción de las entrevistas.

* Las entrevistas deben ser planificadas a intervalos frecuentes y regulares y distribuídas igualmente a lo largo del proyecto: Se comprueba que largos períodos entre sesiones reducen tanto el interés del experto como el impulso del proyecto.

* No interrumpir al experto: Es conveniente permitir al experto que termine su explicación, para que pueda dar una forma acabada a su idea y evitar tener que reveer temas innecesariamente. Una buena práctica consiste en anotar las dudas que surgen cuando él está hablando y preguntarlas ni bien concluya.

* Evitar preguntas irrelevantes: En muchas oportunidades, el tema que se trata despierta el interes del ingeniero del conocimiento quien tiene tendencia a realizar preguntas que, si bien están relacionadas, su respuesta no aporta conceptos importantes (Un ejemplo es el conocer las causas de la enfermedad). Esto conduce a un desvío del tema principal con una consiguiente pérdida de tiempo.

* Mantener el interés del experto en el proyecto: La forma más eficaz de lograrlo es mediante una adecuada retribución. Además se debe lograr que el experto se sienta un miembro activo del grupo haciéndolo participar en la construcción de las reglas, definición de términos específicos y mostrándole cómo el sistema se constituirá en una herramienta útil.

8. ETAPAS DEL TRABAJO

8.1. INTRODUCCION

No se conoce una serie de pasos bien definidos a seguir para la construcción de un Sistema Experto.

Se puede mostrar el desarrollo de este sistema desde dos puntos de vista.

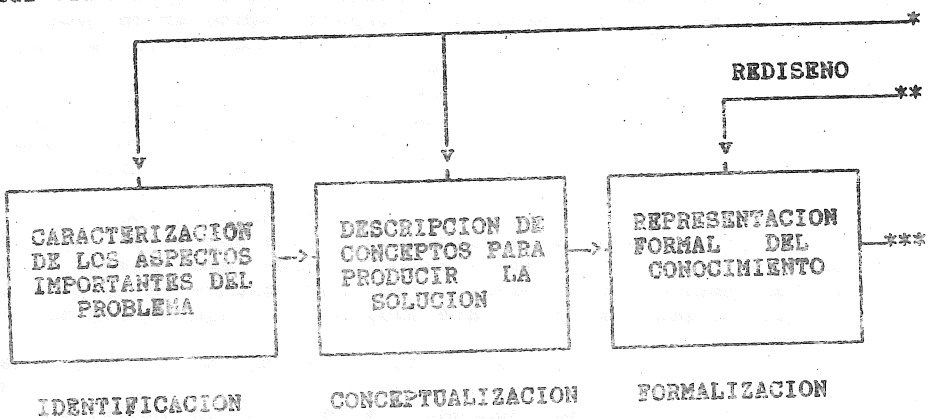
La evolución de un Sistema Experto procede normalmente de tareas simples a tareas complejas, mejorando incrementalmente la organización y representación del conocimiento. Este enfoque incremental permite sacar provecho de lo que se aprende en cada paso del desarrollo.

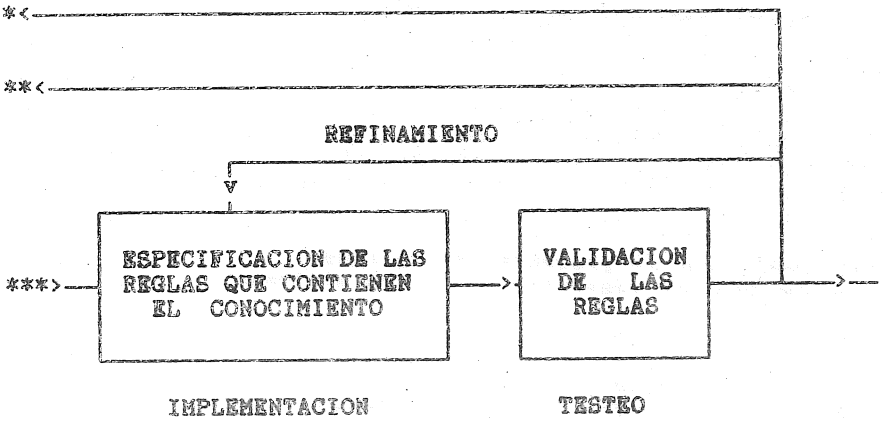
Teniendo en cuenta ésto, se plantea una clasificación del desarrollo del sistema en distintas etapas desde un punto de vista conceptual.

Pero el desarrollo real del sistema está organizado cronológicamente por una clasificación de un nivel de abstracción mayor que el de las etapas de conceptualización. Esta organización cronológica se realiza mediante categorías, cada una de las cuales está conformada por una sucesión de lo que se da en llamar etapas conceptuales.

8.2. ETAPAS CONCEPTUALES

Según Buchanan, el desarrollo de un Sistema Experto puede ser visto como 5 fases superpuestas y relacionadas entre sí:





Aún cuando se puede plantear esta distinción, no hay una manera simple de describir el orden en el cual se desarrollan.

Para este trabajo se tiene :

5.2.1. IDENTIFICACION

Las tareas a realizar son identificar los participantes, los recursos, las características del problema y los objetivos del sistema.

Las primeras tres se han explicado en el transcurso del trabajo. Resta definir los objetivos del prototipo del sistema:

* Dar consejos apropiados sobre casos reales.
* Resolver problemas más rápidamente de la que se hace hoy en día, evitando la necesidad permanente de recurrir al experto, lo que le permitiría dedicarse más profundamente a la investigación.
* Considerar todos los posibles síntomas de las enfermedades.
* Desarrollar una consulta cordial con el usuario.

Se puede esquematizar entonces la descripción inicial del problema de la siguiente forma:

OBJETIVO DEL SISTEMA:
Aumentar la disponibilidad de la experticia apropiada para el diagnóstico y tratamiento de Hipocuprosis, Afosforosis y Ostertagiasis Gástrica.

PROBLEMA PRINCIPAL:
Realizar el diagnóstico de las tres enfermedades.
Sugerir tratamientos adecuados y alternativos.

SUBPROBLEMAS:
Calcular la ponderación de la existencia de cada una de ellas.

Aconsejar análisis para casos dudosos.
Sugerir el tratamiento adecuado.

CARACTERISTICAS DEL PROBLEMA:
Datos incompletos e inciertos.
Asociaciones heurísticas entre datos y soluciones.
Espacio reducido de búsqueda de soluciones.
Soluciones basadas en sumas de eventos inciertos.

DATOS:
Síntomas que se presentan en el rodeo.
Porcentaje de presentación de los síntomas.
Epoca del año.
Edad de los animales.
Raza de los animales.
Zona del establecimiento.
Nivel de minerales en sangre y pasto.
Cantidad de larvas en pasto y en el ganado.

CONCEPTOS IMPORTANTES:
Presentación de síntomas.
Gravedad de la presentación de los síntomas.
Condiciones ambientales.
Características de los animales.
Necesidad de análisis.
Suficiencia de los datos para realizar el diagnóstico.

SOLUCION:
Presencia o ausencia de una o más enfermedades.
Grado de certeza de la presentación de la enfermedad.
Imposibilidad de arribar a una conclusión por falta de evidencias.
Sugerencia del tratamiento adecuado.

8.2.2. CONCEPTUALIZACION.

Durante la etapa de conceptualización se debe decidir qué conceptos, relaciones y mecanismos de control son necesarios para describir la resolución del problema.

Con respecto a los primeros, se puede establecer una distinción entre aquellos que se asocian causalmente con una enfermedad (como pérdida de peso, despigmentación del pelaje, a los que se llama Síntomas), de aquellos que no (edad de los animales, época del año, a los cuales se los denomina Factores Determinantes).

En el transcurso de la captura del conocimiento se detecta una secuencia en el razonamiento del experto. Esta consiste básicamente en relacionar los síntomas observados con los factores determinantes. De esta conjunción surgen indicios que llevan a pensar en alguna de tres posibilidades: un diagnóstico seguro , una fuerte sospecha que induce al pedido de análisis, o la seguridad de descartar cada enfermedad.

En la tarea de extraer reglas del experto observándolo resolver problemas o haciéndole preguntas apropiadas, se tiende a producir reglas demasiado específicas en algunos casos o muy generales en otros. Se debe entonces investigar soluciones alternativas con él para obtener una comprensión de los

principios generales subyacentes en las reglas. Algunas veces se considera de utilidad nombrar conceptos de alto nivel que el experto usa en las reglas, pero que no verbaliza y otras veces resulta conveniente refinar conceptos a niveles más específicos.

Principales conceptos para determinar la presencia de la enfermedad Hipocuprosis:

* Atributos de la despigmentación de pelaje:

  - Edad de los animales afectados: 3 - 24 meses.
  - Incidencia en el rodeo: poblacional (> 10 %).
  - Aspecto del pelo: totalmente despigmentado, viejo, fácilmente desprendible.
  - Estado del resto de los animales: bueno o malo.

* Atributos del trastorno en el crecimiento:

  - Edad de los animales afectados: 3 - 18 meses.
  - Incidencia en el rodeo: poblacional (> 10 %).
  - Características del trastorno: no gana peso a ritmo normal.

De la misma forma se determinan los atributos de otros síntomas, como deformaciones óseas, diarrea, anestro, despigmentación ocular, anemia, fractura espontánea, decúbito.

Principales conceptos para determinar la ausencia de la enfermedad Hipocuprosis:

- La despigmentación de pelaje se presenta en animales mayores de 18 meses en menos de un 7% y con el resto del rodeo bueno.
- El nivel de cobre en pasto es mayor que 4 partes por millón y la relación cobre/molibdeno está entre 1.6 y 2.4.

Existen análisis que proveen información definitiva acerca de la presencia o ausencia de la enfermedad, pero resultan en su mayoría costosos, razón por la cual es necesario evaluar la conveniencia de pedirlos o no a la luz de las condiciones existentes.

Por ejemplo, para el análisis de cobre en sangre:

* Condiciones para pedirlo:
  + Solamente despigmentación de pelaje en animales mayores de 18 meses y el aspecto general del rodeo malo.
  + Solamente animales mayores de 18 meses con despigmentación de pelaje y anestro.
  + Solamente trastornos de crecimiento.
  + En aquellos casos en que clínicamente se sospeche entre un 30 % y un 80 % de la existencia de Hipocuprosis.

* Indica la presencia de Hipocuprosis:

+ Nivel de cobre en sangre < Ø.4 mg por 1ØØ ml.

Conceptos que surgen de la integración de las tres enfermedades:

Otro aspecto importante que se debe tener en cuenta es lo que se denomina la granularidad o grado de detalle de los conceptos. En un principio, y al considerar una enfermedad en particular, las características de determinados síntomas se pensaron sólo en función de ella, en forma aislada. Al incorporar nuevas enfermedades con los mismos síntomas, surge la necesidad de profundizar aún más en los conceptos para encontrar características que permitan distinguirlos entre sí.

8.2.3. FORMALIZACION

La formalización se realiza utilizando un seudolenguaje orientado a reglas SI-ENTONCES para facilitar la implementación.

La siguiente es una muestra de algunas de las reglas que contienen el conocimiento del experto:

```
SI        la edad de todo el rodeo supera los 24 meses
ENTONCES  no es ninguna de las tres enfermedades.

SI        hay despigmentación total de pelaje sólo en
          animales mayores que 18 meses
Y         el resto del rodeo esta en buen estado
Y         no es poblacional
ENTONCES  Casi seguro no es Hipocuprosis.

SI        hay despigmentación total de pelaje no sólo en
          animales mayores que 18 meses
Y         el resto del rodeo esta en mal estado
Y         es poblacional ( más de 6Ø % )
ENTONCES  Definitivamente es Hipocuprosis.

SI        hay disminución de crecimiento principalmente en
          animales chicos ( menores que 18 meses )
Y         la disminución del desarrollo muscular es general
Y         es poblacional ( más de 1Ø % )
ENTONCES  Hay evidencia que es Afosforosis.
Y         Hay evidencia que es Hipocuprosis.

SI        hay animales menores que 18 meses
Y         existe una fuerte sospecha (entre un 3Ø y un 8Ø%)
          que sea Hipocuprosis
ENTONCES  Hacer análisis de cobre en sangre.
```

8.2.4. IMPLEMENTACION

La codificación del conocimiento ya formalizado en el lenguaje utilizado por la herramienta provoca un aumento sustancial del tamaño de las mismas.

Además, la necesidad de mantener consistencia y coherencia en el diálogo con el usuario, como así también mejorar la eficiencia del sistema, requiere un gran esfuerzo y un incremento considerable en la cantidad de reglas.

También se agregaron explicaciones de todos los términos usados por el sistema.

Las características del problema indican la necesidad de ir acumulando evidencias a favor y en contra de la existencia de cada una de las enfermedades. La herramienta utilizada, que en todo lo demás conformaba las necesidades del sistema, no acumula factores de certeza. Fue necesario entonces desarrollar rutinas externas que acumularan los factores de certeza en Pascal.

8.3. ETAPAS CRONOLOGICAS EN EL DESARROLLO DE NUESTRO SISTEMA

Una vez finalizada la tarea de Identificación se comienza a individualizar los casos particulares de cada una de las enfermedades hasta lograr generalizarlos.

De esta forma se desarrolla el sistema en las siguientes etapas cronológicas :

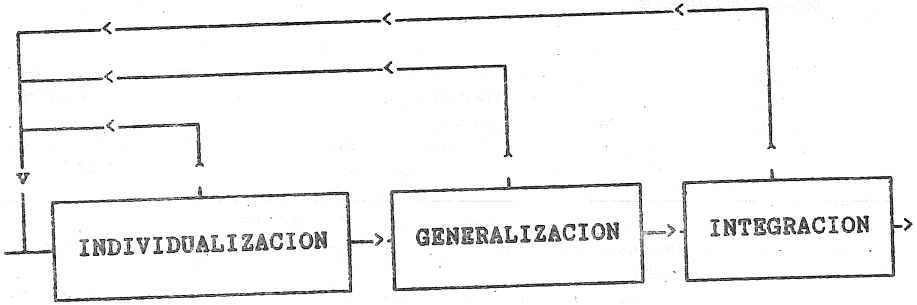
* Individualización de Hipocuprosis.
* Generalización de Hipocuprosis.
* Individualización de Afosforosis.
* Generalización de Afosforosis.
* Integración de las dos enfermedades.
* Individualización de Ostertagiasis Gástrica.
* Generalización de Ostertagiasis Gástrica.
* Integración de las tres enfermedades.

De las cuales se abstraen las siguientes categorías:

* Individualización.
* Generalización.
* Integración.

Cada una de éstas tiene la particularidad de constituir una retroalimentación forzosa de la categoría anterior, estableciendo de esta forma una secuencia fija. Cada una de las categorías planteadas se desarrollan en base a las etapas conceptuales, pudiendo existir secuencias opcionales de reformulación, rediseño y refinamiento.

Estas categorías se pueden esquematizar de la siguiente forma:



Cada enfermedad se desarrolla como un sistema independiente ya que de esta forma resulta más sencillo el testeo, tanto de las soluciones como de las consultas con el usuario.

Al proceso completo de plantear las etapas conceptuales para un caso particular se lo llama Individualización de la enfermedad. Por cada nuevo caso particular, se plantea una nueva Individualización.

Luego se generaliza el conocimiento contenido en las reglas, extendiéndolo de manera tal que contemple todos los posibles casos particulares.

Al contar con más de una enfermedad resulta necesario integrarlas para lograr que gradualmente se comporten como un único sistema (modificando los nuevos aspectos que surgen de la combinación de más de una enfermedad).

En algunas oportunidades es necesario modificar aspectos de la Identificación, como aquellos relacionados con la determinación de los límites del sistema.

9. DESARROLOS FUTUROS : POSIBLES EXTENSIONES DEL PROTOTIPO.

La extensión del prototipo depende básicamente de la repercusión que éste tenga en el ámbito de la medicina veterinaria y de la posibilidad de contar además con nuevos expertos en el área de enfermedades metabólicas.

El prototipo puede crecer en varios sentidos:

* Aumentar la cantidad de enfermedades que se diagnostican.
* Profundizar el conocimiento para tener en cuenta posibles asociaciones entre las enfermedades.
* Abarcar otras regiones del pais.
* Cubrir otros tipos de ganado.
* Considerar otras razas de ganado vacuno.

10. CONCLUSIONES

Si al momento de comenzar este sistema se hubiera elegido el paradigma de los lenguajes de programación convencional para su desarrollo, seguramente hoy día no podría escribirse conclusión alguna. No quiere decir esto que sea imposible desarrollar un Sistema Experto con un lenguaje de programación convencional, sino que sería extremadamente costoso en tiempo y esfuerzo.

La captura del conocimiento, que es la dificultad fundamental inherente a los Sistemas Expertos, obliga a que su desarrollo sea incremental. Esta característica resulta simple de implementar si se cuenta con un lenguaje de Ingeniería del Conocimiento que permita mantener el conocimiento en forma explícita y separada del motor de inferencia. Por el contrario, cualquier modificación en un lenguaje de programación convencional relacionada con el contenido del conocimiento mismo resulta difícil, ya que no sólo se deben realizar modificaciones a nivel de conocimiento, sino también de procedimientos.

A medida que uno se familiariza con los Sistemas Expertos, puede comprobar que en un sentido amplio esta tecnología no es del todo revolucionaria, simplemente es una extensión de los principios básicos de la Ciencia de la Computación a nuevos niveles de sofisticación. Después de todo, los Sistemas Expertos a menudo se ejecutan en los mismos equipos que los programas escritos en lenguajes convencionales y además, en la actualidad, y por razones de eficiencia, se los suele recodificar en estos mismos lenguajes.

A causa de los pocos avances reales que hay sobre el tema, la construcción de un sistema de este tipo requiere gran cantidad de tiempo y dinero, lo que muchas veces causa escepticismo e impaciencia en los niveles de dirección. Además la exageración e imprecisión en torno al tema causa espectativas poco reales que conducen a la frustración o desilusión con respecto a la Inteligencia Artificial y a los Sistemas Expertos.

Todo esto lleva a pensar que a pesar de que los Sistemas Expertos tienen un buen rendimiento, hay importantes áreas en las cuales la experticia humana sería claramente superior a la artificial, pero esto no refleja una limitación fundamental, sino sólo el estado actual del arte...

11. BIBLIOGRAFIA

* TRABAJO FINAL: PROTOTIPO DE SISTEMA EXPERTO DE DIAGNOSTICO DE ENFERMEDADES METABOLICAS EN GANADO VACUNO.

Daga, Mauricio Javier.
Frassoni, Maria Sandra.
Vigil, Nestor Omar.

Universidad Nacional del Centro de la Pcia. de Bs.As.

* A GUIDE TO EXPERT SYSTEMS

Donald A. Waterman — Addison-Wesley Publishing Company.

* ARTIFICIAL INTELLIGENCE IN BUSINESS

Paul Harmon and David King — A Wiley Press Book, John Wiley & Sons, Inc.

* ENGENHARIA DE CONHECIMIENTO E SISTEMAS ESPECIALISTAS

R. Lins de Carvalho — Editorial Kapeluz.

* THE HANDBOOK OF ARTIFICIAL INTELLIGENCE - VOLUME I

Avron Barr and Edward A. Feigenbaum
Department of Computer Science
Stanford University.
Heuristech Press. Stanford California.
William Kaufmann, Inc. Los Altos California.

* DIAGNOSTIC EXPERT SYSTEMS BASED ON A SET COVERING MODEL

James A.Reggia, Dona S. Naut and Pearl Y. Wangs.
Academic Press Inc.

* A FUNDAMENTAL TRADEOFF IN KNOWLEDGE REPRESENTATION AND REASONING (REVISED VERSION)

H.J. Levesque and R.J. Brachman

* MANUAL DE USO DE LA HERRAMIENTA INSIGHT2 PLUS

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa.
Buenos Aires, Setiembre 1988.

UM SISTEMA ESPECIALISTA PARA PREVISÃO DE TEMPO

Vera Helena de Ávila Duarte
Fernando Antônio de Castro Giorno

Centro Científico Rio - IBM Brasil
Estrada da Canoa, 3520
22610 - Rio de Janeiro - BRASIL
Telefone: 021-2713868 Telex: 021-23359

RESUMO

Sistemas Especialistas constituem uma área de pesquisa em Inteligência Artificial com o objetivo de desenvolver sistemas computacionais de desempenho elevado, por meio da incorporação de conhecimento de especialistas e de suas capacidades e habilidades de efetuarem inferências para a solução de problemas. Utilizando-se de técnicas de Inteligência Artificial e de uma base de conhecimento, estes sistemas apresentam um grande potencial de aplicação em diversos domínios de conhecimento.

Na área de previsão de tempo, os sistemas especialistas permitem apoiar o processo de elaboração da previsão de forma eficiente, à medida que o conhecimento do domínio está armazenado em uma base de conhecimento. A previsão pode ser elaborada em um espaço de tempo curto, por meio de interações entre meteorologistas e estes sistemas, permitindo uma dedicação maior dos meteorologistas a outras tarefas, tais como análise de fenômenos atmosféricos e revisão de previsões ocorridas.

Foi com a finalidade de apoiar os meteorologistas na previsão de tempo que foi desenvolvido o sistema especialista BOLETIM, baseado em regras de produção, cujo domínio de conhecimento é a área de previsão de tempo. Sua tarefa é fazer o prognóstico do tempo, para 24 horas, para o Estadó do Rio Grande do Sul.

1.0 *INTRODUÇÃO*

O BOLETIM [1] é um Sistema Especialista (SE) cujo domínio de aplicação é a Meteorologia. Seu desenvolvimento tem como objetivo apoiar os meteorologistas no processo de previsão do tempo. A tarefa de prever o tempo para o Brasil é executada diariamente por um grupo de meteorologistas, que analisa as condições da atmosfera, tanto de superfície como de altitude, e com base nesta análise efetua o prognóstico do tempo para 24 horas.

Como o domínio de conhecimento dos SEs é restrito, decidiu-se fazer inicialmente a previsão do tempo para o Estado do Rio Grande do Sul (RS). Tal região foi escolhida levando-se em consideração sua importância econômica e a qualidade e quantidade de dados meteorológicos oriundos desta região, recebidos diariamente pelo Instituto Nacional de Meteorologia (INEMET). As fontes de conhecimento para o desenvolvimento do sistema BOLETIM foram os meteorologistas do INEMET.

A ferramenta computacional adotada para construção do sistema BOLETIM foi o ESE/VM. O conhecimento é representado nesse sistema sob a forma de regras de

produção, suportanto tanto o encadeamento direto como o inverso para efetuar inferências sobre a base de conhecimento. Tal ferramenta foi escolhida devido à sua disponibilidade no momento, sendo possível adotar outra ferramenta baseada em regras de produção.

2.0 *SISTEMA ESE/VM*

O sistema ESE/VM (indicado a seguir por ESE) foi desenvolvido pela IBM, no seu Centro Científico de Palo Alto, na Califórnia (EUA), em 1985 [3]. ESE é formado por dois subsistemas: ESDE ("Expert System Development Environment") utilizado para construir bases de conhecimento e ESCE ("Expert System Consultation Environment") utilizado para processar bases de conhecimento em sessões de consulta.

ESE opera em arquiteturas IBM/370 e acima, sob os sistemas operacionais VM/CMS ou MVS/XA. O sistema suporta regras de produção como formalismo para representação de conhecimento. O engenheiro de conhecimento usa fatos do domínio, representados por parâmetros ESE e regras para construir uma base de conhecimento relativa a um dado domínio de conhecimento. O sistema ESE tem sido aplicado nas categorias de identificação, seleção, predição, diagnóstico, planejamento e tomada de decisão em áreas diversas de conhecimento.

Algumas das características mais importantes do ESE são:

- sintaxe simples das regras de produção

  As regras de produção são definidas usando uma sintaxe parecida com a da língua inglesa.

- facilidade de edição

  Editores especiais, com estrutura comum para todos os objetos ESE, facilitam a construção, remoção e alteração destes objetos (descritos a seguir) na base de conhecimento. Os editores são moldados para os objetos específicos do sistema e possuem capacidade de cheque automático de erro, notificando imediatamente o engenheiro de conhecimento.

- capacidade de explanação

  Durante uma sessão de consulta, o usuário pode fazer as seguintes indagações: WHAT, para solicitar uma explicação mais detalhada de uma questão efetuada pelo sistema; WHY, para indagar ao sistema porque uma certa questão está sendo efetuada e HOW, para solicitar ao sistema a linha de dedução que permitiu o mesmo atingir uma conclusão.

- suporte para depuração

  Durante testes de uma base de conhecimento, o engenheiro de conhecimento pode pedir um mapeamento exato do processo de inferência utilizado (comando TRACE).

- mecanismo de inferência

  O sistema possui dois mecanismos de inferência: encadeamento inverso e encadeamento direto.

- compilação incremental da base de conhecimento

  Os objetos incluídos na base de conhecimento são "compilados" automaticamente para uma forma que reduz a quantidade de espaço necessária para seu armazenamento.

- suporte para decomposição de problemas

  As características de controle do ESE permitem, quando viável, que um problema complexo seja decomposto em subproblemas menores e mais gerenciáveis. Esses subproblemas podem ser solucionados independentemente, implicando em maior produtividade e qualidade em termos da solução global do problema.

- facilidade para reexecutar e desfazer

  Consultas a uma base de conhecimento podem ser armazenadas e reexecutadas. Durante a reexecução, qualquer resposta dada anteriormente para uma questão efetuada pelo sistema pode ser mudada para estudar seu impacto na solução do problema. Também é possível durante uma consulta desfazer ou mudar uma resposta prévia específica.

- acesso a rotinas externas

  O sistema pode chamar rotinas codificadas preferencialmente em Pascal, para adquirir fatos em arquivos externos, para armazenar resultados em arquivos para processamentos subseqüentes ou para executar processamentos intermediários, modelos matemáticos ou de simulação.

- suporte para projeto de telas

  O sistema ESE trabalha com formatos predefinidos de telas em termos de diálogos e apresentação de resultados. Se necessário ou desejado, o engenheiro de conhecimento pode projetar novos formatos de telas que se tornam disponíveis em adição aos gerados automaticamente pelo sistema.

- apoio em tempo real

  Facilidades extensivas de apoio em tempo real estão disponíveis nos ambientes de desenvolvimento e de consulta, com explicações de como utilizar o sistema.

2.1 Objetos ESE/VM

Uma base de conhecimento contém uma representação codificada de conhecimento relativo a um determinado domínio. Esse conhecimento do domínio consiste basicamente em:

- fatos do domínio
- relacionamento entre os fatos do domínio
- especificações relativas a estrutura/controle do domínio

No sistema ESE estas entidades são representadas respectivamente por:

- PARÂMETROS
- REGRAS DE PRODUÇÃO
- BLOCOS DE CONTROLE DE FOCO (FCBs)

Também podem ser definidos GRUPOS de parâmetros, regras ou FCBs e "displays" especiais para o usário, que são chamados de TELAS. Em suma, o sistema ESE suporta cinco tipos de objetos: parâmetros, regras, FCBs, grupos e telas.

3.0 *METEOROLOGIA SINÓTICA*

Meteorologia sinótica é o ramo da meteorologia que trata das análises das observações meteorológicas feitas simultaneamente em vários lugares da atmosfera, sobre a Terra ou parte dela e suas aplicações nas análises e previsões de tempo e outros problemas.

Dentre as principais ferramentas utilizadas na previsão de tempo estão os **computadores**, que nos anos 40 impulsionaram a previsão numérica de tempo, as **telecomunicações**, que nos anos 50 possibilitaram a confecção dos mapas sinóticos de superfície por meio das informações colhidas, num certo horário (dito sinótico), em todas as estações meteorológicas e enviadas ao centro nacional de previsão, e os **satélites meteorológicos**, que nos anos 60 passaram a fornecer imagens de todo o planeta, com certa periodicidade, sendo importantes em áreas de difícil acesso, onde é problemática a instalação e manutenção de estações meteorológicas.

Uma nova ferramenta a ser considerada é o uso de **sistemas especialistas** [2] [4]. Neste caso, os SEs vêm ajudar a elaborar a previsão de forma mais rápida e eficiente, à medida em que o conhecimento do domínio já está armazenado, sem perigo de ser "esquecido". Em relação ao desenvolvimento de SEs aplicados à previsão de tempo, ao invés dos meteorologistas passarem parte do dia analisando os dados de superfície e de altitude para confecção do boletim meteorológico, bastarão alguns minutos de interação meteorologista/SE para que a previsão seja elaborada. Isso faria com que os meteorologistas pudessem dedicar mais tempo à análise dos fenômenos atmosféricos, como por exemplo, análise das situações onde a previsão de tempo diferiu do ocorrido.

3.1 Previsão de Tempo

Para a execução da tarefa de prever o tempo para 24 horas, em uma determinada região do país, os meteorologistas analisam as condições em superfície e em altitude, usando as informações vindas de radiossondas, das imagens de satélites, das cartas de altitude e de superfície.

As previsões de tempo chegam ao público por meio do boletim meteorológico, emitido diariamente pelo INEMET e divulgado pela imprensa e dos avisos meteorológicos especiais, que são emitidos esporadicamente, na iminência de um fenômeno mais grave, como por exemplo, ventos fortes, geadas, enchentes e incêndios de grande extensão.

As informações comumente contidas em um boletim meteorológico são:

- tempo: determina-se a ocorrência ou não de chuvas, geadas, trovoadas, etc, assim como a existência dos sistemas sinóticos atuantes na região, como frentes frias.

- nebulosidade: verifica-se a existência de nuvens no céu, não as denominando, mas informando a cobertura do céu (céu claro, nublado ou encoberto).

- visibilidade: caracteriza-se pela presença de partículas em suspensão na atmosfera, podendo incluir nevoeiro, chuva, poeira ou névoa seca. É considerada boa quando o observador reconhece um objeto a uma distância entre 10 e 20 km, fraca para uma distância menor que 4 km e moderada para distâncias entre 4 e 10 km.

- ventos: caracterizam-se pela sua direção e intensidade, que é determinada pela velocidade do vento. Para velocidades entre 5 e 10 nós, o vento é fraco, para velocidades entre 10 e 20 nós, o vento é moderado e quando a velocidade é maior que 20 nós, é dito forte.

- umidade relativa do ar: apresenta-se como uma variação da umidade relativa mínima e a umidade relativa máxima.

- temperatura: estima-se as temperaturas máxima e mínima para o período e a variação de temperatura (estável, em declínio ou em elevação).

Dentre as informações citadas, o sistema BOLETIM só efetua a previsão de tempo, informando a ocorrência ou não de chuvas e geadas.

4.0 *DESENVOLVIMENTO DO SISTEMA*

4.1 Aquisição de Conhecimento

O processo de aquisição do conhecimento de meteorologistas durou 10 meses e foi dividido em 5 fases principais:

1. familiarização com os termos técnicos em meteorologia, por meio da leitura de textos básicos;

2. acompanhamento de meteorologistas na tarefa de prever o tempo;

3. entrevistas com meteorologistas, com o intuito de estabelecer os sistemas que atuam no Rio Grande do Sul (Frente Fria, Ciclone, Sistema Convectivo do Chaco, Alta da Bolívia, Anticiclone do Atlântico, Anticiclone Polar e Cavado do Pacífico), os parâmetros de superfície e altitude que são analisados nesses sistemas para a previsão do tempo e os relacionamentos entre tais parâmetros;

4. formalização do conhecimento adquirido utilizando uma ferramenta computacional para construção de SEs (ESE/VM);

5. teste e refinamento das regras de produção que formam a base de conhecimento do BOLETIM.

4.2 Implementação Computacional

A fase 4 do processo de aquisição de conhecimento constitui-se na formalização do conhecimento adquirido durante as 3 fases anteriores.

A escolha do ESE/VM como ferramenta computacional baseou-se em uma experiência anterior feita no INPE, que culminou com o desenvolvimento de uma ferramenta para construção de SEs [5]. Essa experiência mostrou a aplicação de SEs baseados em regras de produção na Meteorologia. No caso, a aplicação era previsão de precipitação pluviométrica em São José dos Campos. Apesar do sistema possuir uma base de conhecimento reduzida (apenas 50 regras, dado que o escopo principal era a construção da ferramenta), notou-se que o uso de SEs na Meteorologia era viável e que poderia ser desenvolvido um tal sistema, baseado em regras de produção, para previsão de tempo. Essa viabilidade foi confirmada durante as fases 2 e 3 do processo de aquisição de conhecimento, onde foi observado que o conhecimento dos meteorologistas é empírico, apoiando-se em observações e experiências anteriores.

A fase de implementação computacional começou com a formalização dos fatos do domínio, relações entre os fatos e estruturas de controle que são representados por parâmetros, regras e FCBs (Blocos de Controle de Foco) no sistema ESE.

4.2.1 Parâmetros

O sistema BOLETIM possui atualmente 65 parâmetros, sendo que alguns deles podem ser analisados em mais de um sistema atuante no RS. Para melhor entendimento, os parâmetros podem ser divididos em 3 categorias:

- parâmetros cujos valores serão fornecidos pelo usuário (parâmetros de entrada), tais como, advecção de temperatura, advecção de vorticidade, temperatura do ar, etc.
- parâmetros cujos valores serão determinados pelo processamento das regras de produção e que serão usados para processar outras regras (parâmetros intermediários), tais como, convecção, umidade, intensidade da frente fria, etc.
- parâmetros cujos valores serão determinados pelo processamento das regras de produção e que serão mostrados ao usuário no final da consulta como resultado da solução do problema (parâmetros de saída), tais como, chuva fraca, geada, pancadas de chuva, etc.

4.2.2 Regras

Os relacionamentos entre os parâmetros, formalizados por meio de regras de produção, constituem a base de conhecimento do sistema BOLETIM. Ao todo são 885 regras (veja tabela 1, anexo 1), onde metaregras são regras que descrevem como outras regras devem ser usadas.

4.2.3 FCBs

O conceito e a disponibilidade de FCBs foi importante na estruturação do problema. Como os sistemas atuantes no RS são distintos, optou-se por utilizar um FCB para cada sistema, sendo que o FCB raiz (FCB_SIST_ATUA) determina qual o sistema atuante no RS. Isso possibilitou dirigir a busca, por meio de metaregras, fazendo com que somente as regras referentes ao sistema atuante sejam processadas.

A estrutura hierárquica dos FCBs pode ser vista na figura 1 (anexo 2) e a seguir tem-se uma definição sucinta dos mesmos.

4.2.3.1 FCB_FF

Como o sistema Frente Fria é o mais importante e o que possui mais regras, o FCB_FF foi subdividido com o objetivo de tornar o processo de busca da solução do problema mais eficiente. O FCB_FF possui a estrutura hierárquica da figura 2 (anexo 2).

FCB_INTENS, FCB_CONVEC, FCB_UMID e FCB_ENTR

Cada um destes FCBs resolve o subproblema particular de verificar a intensidade da frente fria, a existência ou não de convecção e umidade no RS e a possibilidade da frente fria entrar no RS, respectivamente. Portanto, todas as regras que concluem intensidade da frente fria, convecção e umidade no RS e entrada da frente fria no RS pertencerão a cada um destes FCBs e serão usadas pelo mecanismo de encadeamento direto.

FCB_TEMPO

Este FCB resolve o problema de prever o tempo no RS, utilizando os parâmetros que já foram determinados anteriormente (intensidade da Frente Fria, existência de convecção e de umidade no RS e entrada da Frente Fria no RS) pelos FCBs FCB_INTENS, FCB_CONVEC, FCB_UMID e FCB_ENTR. Para isso, o FCB_TEMPO foi dividido em 4 FCBs: FCB_BB, FCB_BP, FCB_CA e FCB_NU.

Dependendo da localização da Frente Fria (Baía Blanca, Bacia do Prata, Centro da Argentina ou Norte do Uruguai), o FCB_TEMPO irá determinar qual dos FCBs mencionados anteriormente é que vai ser processado. Por exemplo, se a Frente Fria estiver localizada no Centro da Argentina, então somente as regras contidas no FCB_NU serão utilizadas para solucionar o problema.

4.2.3.2 **FCB_CICL, FCB_SCC, FCB_AB, FCB_AA, FCB_AP e FCB_CP**

Cada um destes FCBs processa as suas regras, usando o mecanismo de encadeamento direto. Ao final do processamento das regras, o resultado (que é a previsão do tempo) é mostrado ao usuário.

5.0 *RESULTADOS OBTIDOS*

Após testes sucessivos e refinamentos da base de conhecimento, o sistema BOLETIM apresentou os resultados mostrados na tabela 2 (anexo 1).

Essa tabela foi elaborada com base em 42 casos reais, onde foi possível comparar as previsões dos meteorologistas (indicado por PREVISÃO) e do sistema BOLETIM (indicado por BOLETIM) com o que realmente aconteceu (indicado a seguir por OCORRIDO).

Analisando a tabela, observa-se que o sistema especialista BOLETIM teve um índice de erros de aproximadamente 16.66%, sendo que 9.52% dos casos, tanto os meteorologistas quanto o BOLETIM erraram. Já o índice de acertos pode ser considerado bom, atingindo aproximadamente 83.34%, sendo que 9.52% dos casos foram considerados ótimos. Esses casos ótimos são aqueles onde a previsão dos meteorologistas foi errada e a do BOLETIM foi correta, ou seja, sua previsão foi a mesma do OCORRIDO. Os 73.82% dos casos restantes foram de acertos gerais, onde as previsões do BOLETIM e dos meteorologistas foram as mesmas do OCORRIDO.

Observou-se que os erros cometidos estavam relacionados com situações consideradas anormais, tal como a ocorrência do Anticiclone Polar, com deslocamento continental, durante o verão. Esta situação é típica do inverno, onde este sistema provoca geadas no RS. Outros erros dizem respeito aos chamados "acasos", onde todos os dados levam a uma previsão e tal previsão é errada, sendo que às vezes nem os meteorologistas sabem explicar o motivo do erro.

Os casos em que ocorrem dois sistemas interagindo e atuando no RS também foram passíveis de erro, sendo algumas vezes um erro parcial, como por exemplo, o sistema BOLETIM concluindo chuva no S, SE e SW do RS e o que realmente ocorreu foi chuva só no SE.

AGRADECIMENTOS

Os autores agradecem ao Instituto Nacional de Meteorologia pela cooperação dos seus meteorologistas, especialmente Expedito Rebello, que foi o meteorologista coordenador e principal fornecedor de conhecimento, a Sandra Aparecida Sandri, do Instituto de Pesquisas Espaciais, e ao Dr. Celso de Renna e Souza, do Instituto Tecnológico de Aeronaútica, pelas suas valiosas observações.

REFERÊNCIAS BIBLIOGRÁFICAS

1. DUARTE,V. *BOLETIM - Um Sistema Especialista para Previsão de Tempo.* Dissertação de Mestrado em Computação Aplicada. São José dos Campos, INPE, 1987 (a ser publicada).

2. ELIO,R.; HANN,J. Knowledge representation in an expert storm forecasting system. In International Joint Conference on Artificial Intelligence, 9., Los Angeles, Aug., 1985. *Proceedings.* IJCAI, 1985. p. 400-406.

3. IBM. *Expert system consultation environment/VM and expert system development environment/VM*; reference manual. Irving, Texas, IBM, 1985. (GH20-9609)

4. REISE,C.; ZUBRICK,S. Using rule induction to combine declarative and procedural knowledge representations. Austin, Texas, Radian Corporation, 1985. (Technical Report RI-R-00299).

5. SANDRI,S. *Sistema diagnóstico*: uma ferramenta para construção de sistemas especialistas. Dissertação de Mestrado em Computação Aplicada. São José dos Campos, INPE, 1985. (INPE-3890-TDL/222).

6. WATERMAN,D. *A guide to expert systems.* Reading, MA, Addison-Wesley, 1985.

ANEXO I

Sistema Atuante	*Número de Regras*
Frente Fria	*484*
Ciclone	*134*
Sistema Convectivo do Chaco	*67*
Alta da Bolívia	*93*
Anticiclone do Atlântico	*32*
Anticiclone Polar	*26*
Cavado do Pacífico	*37*
Metaregras	*12*

Tabela 1: Conjunto de regras do sistema BOLETIM.

Erros		*Acertos*	
BOLETIM	*Previsão*	*BOLETIM*	*Previsão*
16.66%	*9.52%*	*83.34%*	*73.82%*

Tabela 2: Resultados obtidos.

ANEXO 2

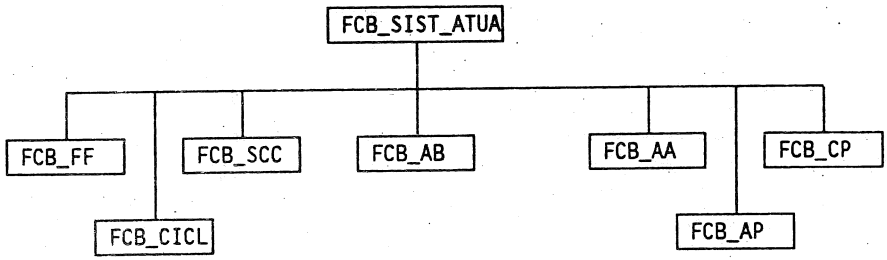


Figura 1: Hierarquia dos FCBs no BOLETIM.

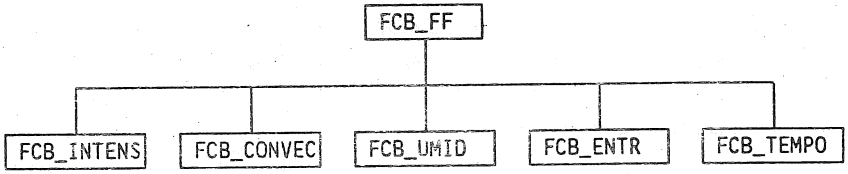


Figura 2: Estrutura hierárquica do FCB_FF.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

Knowledge Engineering: Prototyping a Small Knowledge Based System

by
Walter G. Rolandi
Knowledge Engineer
NCR Advanced Systems Development
Engineering and Manufacturing, Columbia, SC

Introduction

Although several sources have recently appeared on the subject of knowledge engineering, there is no clear consensus as to how the software developer should go about that task. In fact, with only a handful of exceptions, (for an excellent discussion, see Hoffman, 1987) most of that which has been published describes little more than a general theory or some skeletal, non-specific methodology. As more and more software engineers, programmers, and analysts, especially those coming from MIS or DP backgrounds, become involved in the development of knowledge based systems, the demand for specific descriptions of actual projects will grow.

This article seeks to address that demand. It is intended to briefly chronical the development of a knowledge based prototype at NCR Advanced Systems Development. Some of the details of the application are slightly fictionalized in order to avoid identifying particular individuals involved in the process and to protect proprietary interests. The overall scope and intent of the project is however clearly stated as are the knowledge engineering procedures undertaken to automate the process. Explanations of significant decisions and developments in the project life cycle are included.

The article picks up at a point in the project life cycle where the need for a knowledge based system has been identified, and the problem has been initially investigated. Basics as to management expectations, user requirements, and system development constraints have been determined. An "expert" has been identified and a feasibility prototype has been commissioned. The circumstances suggesting the need for a system are described.

Basic Background of the Problem

NCR customers are provided with toll-free telephone access to both hardware and software support. This support is provided on a nationwide bases. All calls, regardless of their origin in the U.S., are first channeled through a central communication office. Clerical telephone and CRT operators answer each call, obtain vital customer and product information, construct a description of the problem that the customer is having, and ultimately pass the "incident" on to an intermediary analyst. This intermediary analyst acts as an expediter and problem classifier. He scans the incident report to classify the problem as hardware related, OS related, application related, etc. and if appropriate, passes the incident on to more specialized analysts who re- establish contact with the customer and solve the problem. Part of this intermediary analyst's job is to recognize obvious hardware problems, confirm the problem by recontacting the customer, and dispatch a field engineer to repair whatever is wrong. This procedure allows a relatively fast turn around response to obvious hardware problems and also prevents the wasteful involvement of more specialized analysts.

Automation Task

The intermediary analyst and the abilities he possesses obviously represent a valuable asset. His skills help provide customers with timely resolutions to their problems while conserving highly specialized resources in order that their expertise can be more advantageously employed to solve more difficult or subtle hardware or software problems. The intermediary analyst can personally resolve as many as 11% of the total daily incidents. Because of his prowess for classifying problem descriptions and expediting problem resolutions, his "expertise" became the object of our analysis. Our goal was to build a system which would assist clerical personnel in classifying problem descriptions in a manner that was functionally equivalent to the expert. Upon completion of a successful prototype, the system would then be transparently interfaced into the existing database system used by the clerical telephone and CRT operators who answer the customers initial calls. Implementing the system as such, the customer would be better served. By virtue of the fact that the system would provide them with the knowledge to accurately assess the customer's problem, the clerical personnel could either fix it or immediately dispatch a field engineer if needed.

Getting Started

As with many software engineering projects and particularly with knowledge based systems, sometimes getting started can be a problem. Upon initial inspection, the problem domain might appear chaotic or seem overwhelmingly complicated. A large part of the knowledge engineer's job is to find order in such seemingly disordered decision making processes. The knowledge engineer cannot acquire nor functionally represent the knowledge of an expert without first obtaining a predictive understanding of what the expert does when they exercise their expertise. Behavior analytic methods are extremely effective to this end. (Rolandi, 1986).

"Predictive understanding" is meant to include answers to questions such as:

Exactly what decisions does the expert make?

What are the decision outcomes?

Which outcomes require greater reflection, exploration, or interaction than others?

What resources or inputs are required in order to make a decision?

What conditions are present when a particular outcome is decided?

At what point after exposure to decision influential inputs is a decision made?
How consistently do these conditions predict a given outcome?

and ultimately:

Given the particulars of a specific case, will the outcome predictions of the KE team be consistent with that of the expert?

Although basics obviously would precede more specifics, no other particular sequence is implied by the above listing. Answers to these questions need to be obtained as the questions themselves arise, regardless of when they arise in the knowledge engineering process. The specifics of how these questions were answered are described.

What decisions were being made by the expert?

In order to answer this question, the knowledge engineering team needed to define the scope of the overall decision making process. Initial interviews with the expert were arranged and conducted. Because behavioral analysis is an empirical process, descriptive quantitative data was sought. Since preliminary analysis of the process had identified the incident reports as a necessary component of the expert's decision making, a large sample set of incident reports was obtained for analysis. In order to identify the characteristics of a "typical" decision, an inexpensive tape recording system was put into place that allowed recordings of the conversations between customers and the expert. As mentioned earlier, these conversations occurred when the expert recontacted customers after having read their incident reports. The expert was asked to mention the incident number on the recording so that the team could later correlate that expert/customer interaction with a copy of the incident report. A series of taped interactions were transcribed and coordinated with their respective incident reports. Thus a wealth of data that realistically depicted the creation, classification, and resolution of a large set of cases was compiled.

What were the decision outcomes?

Perusing the body of data, the team was able to clearly identify decision outcomes. For the purposes of our automation task, there were basically three outcome categories: obvious hardware, obvious software or application, and other. "Other" essentially included everything (regardless of whether it was ultimately determined to be hardware or software related) that was not identified in or inferable from the incident reports. Given the team's objective to automate recognition of only obvious hardware cases, the focus of our analysis shifted to that data only.

Which outcomes required more interaction than others?

Using all the cases in our library that the expert identified as obvious hardware problems, the team compiled statistics on the number of conversational interactions between expert and customer. It was discovered that all of these interactions were characteristically short with a "typical" interaction involving only two exchanges of information. Upon inspection and classification of those interactions, it was determined that in 80% of the cases, the interactions served only to confirm the problem description as recorded on the incident report by the clerical telephone operator. This meant that whatever was necessary to make an immediate hardware decision was,

in most cases, contained in the problem description. Given that we had apparently located the decisive input for 80% of the obvious hardware cases, the problem descriptions became our next focus of analysis.

What resources or inputs were required to make a decision?

Although the team was convinced of the preeminent influence of the problem description on the decision making process, we needed to be certain that there were no other more subtle forces at work. The expert was observed in his work environment under his typical work conditions. He was asked to report the point at which he felt that he had made a decision for a number of cases. He was also asked to state what caused him to come to this decision. As expected, he reported information in the problem description alone. Additionally, he reported that his decision was often already made before he reestablished contact with the customer.

The expert was then asked to decide cases without access to the incident report. In other words, a knowledge engineer held the incident report reading aloud only the information the expert requested. In all cases, a decision was made based on the problem description alone. In retrospect, this may seem like overkill. The incident reports however contain several sources of information that could well have contributed to the expert's decision outcome. The team needed to isolate only those sources that were in fact decisive.

What conditions are present when a particular outcome is decided?

What information was extracted from the problem descriptions that accurately pointed to an obvious hardware classification? In order to address this question, the team began an in depth analysis of all of the problem descriptions that were identified as representative of obvious hardware decisions. Having previously been entered into a commercial database system, the problem descriptions were electronically available. A "C" program was written to parse the problem descriptions into a simple list of all words contained therein. The UNIX sort utility was employed to produce reports that alphabetized and listed all of the problem description words and additionally provided an indication of their relative frequencies. From these reports, the team was able to immediately identify a list of words that predicted obvious hardware decisions. The list that emerged was a list of peripheral hardware devices. Additionally, there emerged a list of words that when associated with a particular device, inevitably resulted in a hardware decision. Some of the descriptors had a "heuristic" effect. That is, the expert's decision could be predicted given the presence of certain descriptors regardless of the device that was implicated. This meant that there are attributes for all instances of computer hardware that if instantiated with a particular value, reliably predict hardware malfunction. These more global predictors emerged as a function of writing a number of problem descriptions on small pieces of paper and affixing them to a blackboard for analysis. Acting out a kind of "categorical imperative", the team arranged the problem descriptions into clusters by device type. The device type clusters were then subcategorized into problem descriptions with similarities. Not only did this exercise reveal patterns across hardware devices, it also discerned patterns of device specific problems.

How consistently did these conditions predict a given outcome?

These word lists so reliably predicted a hardware decision that the team decided to experiment with a very primitive natural language interface. This interface was designed to simply scan the problem descriptions first for words that identified a specific hardware device. Secondly, (if the first pass failed), it scanned the text for alternative words that reliably predicted or indicated the

involvement of a specific hardware device. And lastly, the program scanned the problem description for particular words that when associated with a known hardware device always indicated a field engineer was needed. It was hoped that this system would automatically flag all clear cut hardware cases and leave other possible hardware cases to be interactively examined by the system while the telephone operator extracted system prompted input requests from the customer by phone.

As it turned out, the immediate identification of "clear cut hardware cases", whether by expert or machine, was totally dependent upon a clearly stated and informative problem description. For problem descriptions that included or identified a hardware device or a part or component of a hardware device, and a predictive descriptor of that device or part, this system worked amazingly well. Initially we were worried about false positives where the problem descriptions contained the important keywords but not in the way that reliably implicated a known problem with a known device. As it turned out, this was not really a problem. This scheme's weakness was that it had to rely on the problem descriptions without benefit of subsequent interaction and unfortunately, only a relatively small number of the free form problem descriptions contained the information needed to make a decision. For the most part, they were too general ("system inop") or not actually descriptions of computer problems at all. For example, many problem descriptions were really assessments of the urgency of the solution ("checks won't print. need ASAP"), or statements referring to the emotional state of the customer ("problem with system, customer sounded frustrated"). In cases such as these, the expert would typically recontact the customer by phone to obtain a more specific description of their problem.

Although the simple parser scheme accurately identified all obvious hardware problems which had problem descriptions including a device, a device part, and a device or device part descriptor, we abandoned the scheme because it simply was too limited in its range. In other words, the percentage of the hardware cases that it would automatically flag was not significant although those cases so flagged were accurately identified. As an alternative, a menu system was implemented in a commercially available inference engine running on an NCR Tower 32.

Our experimentation with the parser was most productive however: by the time we had decided to abandon the interface, we had exhaustively identified all hardware devices and their component parts that if described by a member of a limited set of descriptors, reliably indicated that a field engineer was needed. We sought to obtain a measure of the consistency of these predictors with the decisions of the expert.

Using a set of incident reports that the expert had never before seen, the expert was asked to classify the problem as being related to hardware, software, or other. The incident reports were obtained from a history file that included a description of the ultimate solution to the problem. Thus, the team could identify which incidents involved actual hardware issues and which did not. By the way, with a large number of cases, this sort of exercise will typically reveal some degree of inconsistency in the performance of the expert. That is, cases with identical problem descriptions would occasionally be reacted to differently by the expert. This is important to note but is only problematic when the administration of the expert's expertise becomes unpredictable. Small inconsistencies are to be expected but true unpredictability would signify the absence of expertise.

The outcome of this exercise was to further reinforce our faith in the reliability of the keyword predictors. There was essentially a one-to-one correspondence between our predictions of what the expert would do given a certain problem description and what he actually would do. Needless to say, this was encouraging.

At this point in the knowledge engineering process, we had answered most of the questions one needs to answer to arrive at a predictive understanding of the expert's decision making behavior. We had discerned that the truly obvious hardware decisions would not be made by the expert unless he had first identified one or more of a limited number of devices or device parts, and one or more of a limited number of descriptors of those devices. Their presence essentially was an assertion of fact and as such, could be succinctly represented:

Device	Component or Part	Descriptor
printer	ribbon	tangled
terminal	key	jammed
tape drive	door	broken

The determination was driven in prototypical implementations by the backward chaining rule:

if	a hardware device is identified
or	a part, component, or attribute of a hardware device is identified
and	a significant descriptor of that device or component is present
then	there is an obvious hardware problem and a field engineer is called for.

The purpose of the menu system was to effectively construct an instance of some fact which was recognized as constituting an obvious hardware decision by our expert and represented in the knowledge base.

Of course not all hardware decisions were representable as constructable facts. As mentioned earlier, some problem descriptions do not contain the necessary information for an immediate classification. In such cases, the expert would telephone the customer and ask them to elaborate on their problem. This interaction would continue until such time as the expert had classified the problem as hardware or knew that it should be passed on to one of the analysts. His interaction with the customers not only provided him the means to formulate and test as many hypotheses as needed to classify the problem, it occasionally provided an opportunity to solve some of the customers' problems. This is because the expert responded to cues in the follow up customer interactions that could indicate any of a number of more subtle kinds of problems. The ability to make these sorts of discriminations was sought for the system as well.

This aspect of the KE process entailed a slightly different approach. We had already obtained an accurate assessment of the scope of the domain, that is, the range of things about which something had to be known. We had also discerned the critical components (as well as an effective way to functionally represent them) of the more common sense, obvious hardware problems. These types of problems essentially entailed a one-to-one mapping from something contained in a problem description to some actionable fact contained in the knowledge base. But what of less specific problem descriptions or subsequent conversational interactions? What was it about these "inputs" that served as a basis for action? Another way of asking this question is, "What does the expert have to *otherwise* know in order to respond to the cues he detects in conversation with the customers?"

What knowledge is subsumed by the assertion of given facts?

The expert was asked to role play a number of follow up scenarios as if he had been given incident reports containing very non-specific problem descriptions. It was soon discovered that, given a particular topic of discussion, (either a class of device or a class of problem indicators), he would seek to determine answers to only certain kinds of questions. For example, if some unspecified thing was wrong with the customer's printer, the expert would typically ask the model number of the printer. His subsequent conclusions or line of questioning were totally contingent upon the response of the customer. This is because knowing only one fact (the model number) actually implied knowing a great number of facts. It was almost as if upon learning a model number, the expert "loaded in to memory" as many as fifty other pieces of information that are associated with that fact. Regardless of the psychological mechanism actually entailed in this process, the analogy served the interest of the

project well.

General rules of inference that functionally conformed to the interactions of the expert were encoded in the knowledge base. External files were employed however to tabularly contain the large number of facts about printers, terminals, modems, etc.. These facts were then "loaded in" as a sort of frame of reference when called into play by the inference mechanism. What this netted was the ability to make very refined and specific decisions based upon relatively sparse inputs. Knowing what he knew about a wide range of hardware objects, the expert was capable of responding to a number of rather subtle cues that, to him, implicated certain kinds of problems or certain kinds of problem solutions. By determining a substantial subset of what that knowledge entailed, we were able to realistically emulate his problem solving behavior even with more obscure problems.

This ability has proven to be quite impressive. First of all, the representation scheme is straightforward, easy to maintain, and capable of providing logical branches to a vast number of problems. It also provides a means to appropriately react to and solve a number of problems that it discerns. It is this ability that seems to convey the impression of "intelligence" when the prototype is demonstrated. Reactions to this latest prototype have been quite favorable with viewers often being impressed at the range of what the system "knows".

Summary

The knowledge engineering process employed to construct a working prototype of a knowledge based classification system has been described. The approach taken was explorational and experimental in nature. Quantitative techniques borrowed from behavior analysis were advantageously employed. The process entailed the construction of several prototypical implementations before the current knowledge representation scheme was adopted. The events described represent about 7.5 man-months of data collection, data analysis, and prototype programming. It is hoped that the specific methods and procedures described will be helpful to other software engineers involved in the development of knowledge based systems.

References

Hoffman, R. R., (1987), *The Problem of Extracting the Knowledge of Experts from the Perspective of Experimental Psychology*, AI Magazine, Vol. 8, No. 2.

Rolandi, W.G., (1986) *Knowledge Engineering in Practice*, AI/Expert Magazine, Vol. 1, No. 4.

Bio-line

Walter G. Rolandi is a knowledge engineer with Advanced Systems Development, NCR-Columbia. Interested in machine learning and language acquisition, he is additionally pursuing graduate training in the departments of psychology and linguistics at the University of South Carolina.

Acknowledgements

Particular appreciation is extended to Diana Austin and Ken Steimer for both their technical expertise and their insights into the nature of the problem. Additionally, I would like to thank our management, Jim Davis and Wayne Miller for their support and encouragement. Lastly, I would like to thank our ultimate expert, Mr. Jerry Hodges.

Capítulo 10

Informática en la Universidad

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

Un Laboratorio de Estructuras de Datos

Jorge Villalobos
Mario J. Otero
Alejandro Quintero
Olga Mariño

Departamento de Sistemas y Computacion
Universidad de los Andes
Apartado Aereo 4976 - Telex 42343 Unand Co
Bogotá 1, D.E., COLOMBIA

Resumen: El propósito de este articulo es describir brevemente la metodología de enseñanza del curso Estructuras de Datos dictado en la Universidad de los Andes y presentar el ambiente de programación LED (Laboratorio de Estructuras de Datos) desarrollado en la Universidad como apoyo práctico del curso. Este laboratorio permite al estudiante "experimentar" con sus algoritmos y sus estructuras de datos, dandole facilidades gráficas para observar su comportamiento, medir su eficiencia y poder compararlo con otras posibles soluciones de un problema dado. Le permite ademas disponer de mecanismos para seguir en la practica el proceso de desarrollo y evaluacion que se sigue en la teoría. El laboratorio constituye el complemento práctico del libro "Estructuras de Datos: Un Enfoque desde Tipos Abstractos" [VQ088], publicado por el Departamento de Ingeniería de Sistemas y utilizado actualmente en varias universidades.

Palabras Clave: Estructuras de Datos, Tipos Abstractos de Datos, Ingeniería de Software, Ambientes de Programación, Lenguajes de Programación.

1. Introducción.

Una de las realizaciones importantes del Departamento de Ingeniería de Sistemas y Computación de la Universidad de Los Andes en los últimos tiempos, ha sido la de dar a la totalidad de su trabajo una sólida base teórica y formal. Es así como en algunos de los cursos ya se ha logrado obtener un material nuevo que refleja los últimos avances en esta área. Siguiendo esta línea de desarrollo, se ha publicado un texto especialmente diseñado para el curso de Estructuras de Datos [VQ088], en el cual se presenta el material que podríamos llamar "clásico" en la enseñanza de las estructuras de información, enfocado desde un punto de vista de tipos abstractos de datos (TADs). Este libro recoge las colaboraciones, el trabajo y la experiencia docente de varios profesores del Departamento de Sistemas de la Universidad, lo cual permite afirmar que el material que allí se presenta corresponde a una metodología bastante depurada, con alto grado de formalidad y a la vez práctica para definir tipos y manipular estructuras de datos de manera constructiva.

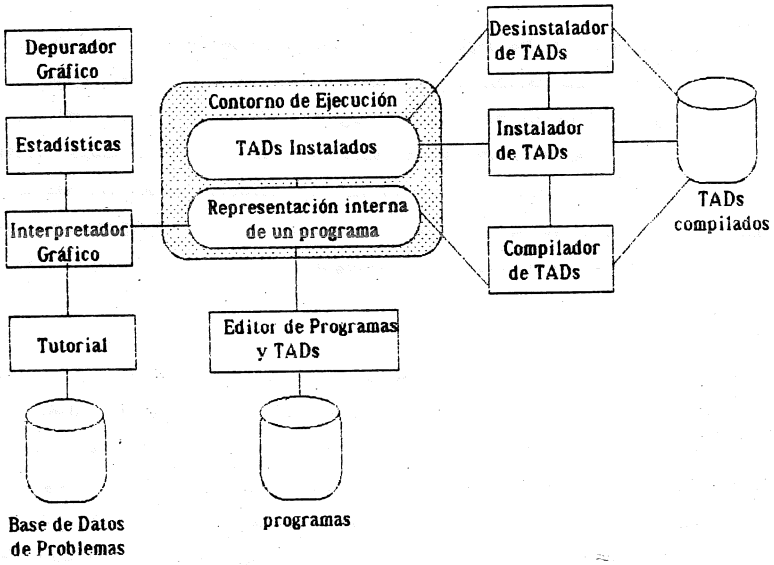
La metodología se basa en dos aspectos: una disciplina de diseño a través de Tipos Abstractos de Datos y una disciplina de desarrollo mediante la utilización de técnicas formales de programación, en las cuales el lenguaje gráfico juega un papel vital. Este nuevo enfoque ha sido utilizado desde hace dos años con resultados realmente alentadores (en cada semestre se dicta para unos 200 estudiantes de diversas áreas de Ingeniería), y aunque no se contaba inicialmente con un componente práctico satisfactorio se ha podido probar la metodología con un conjunto significativo de estudiantes.

Desde el mismo momento en que se comenzó a utilizar este nuevo enfoque en la enseñanza de Estructuras de Información, se sintió la necesidad de contar con un laboratorio en el cual el estudiante pudiera "experimentar" con sus algoritmos y sus estructuras de datos, así como disponer de herramientas suficientes para seguir en la práctica el proceso de desarrollo que se presenta en la teoría. Dicho laboratorio fue desarrollado en el último año, y cuenta con un editor guiado por la sintaxis (que se adapta a la metodología de desarrollo y de diseño), un interpretador gráfico, un depurador gráfico, un manejador de TADs (compilador, interpretador, generador de datos de prueba, etc.), un manejador de estadísticas (dibuja la función tiempo contra

número de datos procesados, calcula el tiempo de ejecución de cada rutina, etc.), un módulo tutorial con un resumen del curso, y otras herramientas para el estudiante (traductor del lenguaje del laboratorio a Pascal o a C, etc.). Los anteriores modulos se describen en los numerales 3 a 8 de este articulo.

El objetivo de este articulo es describir superficialmente el enfoque del curso (para una descripción más detallada consultar [VQ088]) y mostrar el laboratorio incluyendo algunos aspectos de su implementación. Se presenta además el lenguaje utilizado, que corresponde a una extensión de LCG (Lenguaje de Comandos Guardados) descrito en [MAR88] [BOH86] y que se utiliza en la Universidad como primer lenguaje algoritmico en la enseñanza de programación. El lenguaje y la metodología de desarrollo están basados en el trabajo de Dijkstra y Gries [DIJ76] [GRI81].

El siguiente es un esquema de la estructura del laboratorio, que muestra los diferentes módulos que lo componen y sus relaciones:



Es importante reconocer el trabajo de los estudiantes Adriana Bucheli, Henry Castellanos, John Díz, Ricardo Gómez, Luis Fernando Leal, Sandra Sanchez, y Jorge

Restrepo, quienes fueron los encargados de implementar el laboratorio, así como agradecer el apoyo del Departamento en este proyecto.

2. Metodología del curso.

En el curso se utiliza una metodología formal de desarrollo de programas, en la que se abandona el método tradicional y se pasa a una programación mucho más "disciplinada" y efectiva. En esta metodología guiada por objetivos, el estudiante utiliza refinamiento por pasos y aserciones obteniendo al final, además del programa, la prueba de su corrección.

Las aserciones se utilizan con tres propósitos. El primero es para especificar. Se debe colocar una aserción antes y después del problema y de cada uno de los subproblemas en los cuales ha sido dividido (pre y postcondición). Estas aserciones definen de manera no ambigua el problema y la interfaz entre cada una de sus divisiones. En segundo lugar las aserciones sirven como herramienta de desarrollo formal que hace posible "calcular" un programa en lugar de "improvisarlo". Esto permite por ejemplo encontrar la condición de salida más adecuada de un proceso iterativo mediante la manipulación y observación de las aserciones (el invariante del ciclo y su postcondición), en lugar de hacerlo por pura intuición. De esta manera existe garantía de obtener programas correctos [BOH86]. El tercer propósito es la documentación. Al final, las aserciones se pueden utilizar como una descripción del programa haciendo innecesario el uso de comentarios (es posible entender el funcionamiento de un programa viendo únicamente sus aserciones intermedias).

Esta metodología ha sido adaptada a las necesidades específicas del curso, y se ha definido una semántica para el lenguaje gráfico, vital en la construcción de aserciones que manejan estructuras de datos. La experiencia ha sido tan positiva, que se está comenzando a utilizar la misma metodología de desarrollo desde el primer curso de programación, para el cual ya se desarrolló el respectivo material escrito [MAR88].

Con respecto al diseño, se hace énfasis en la necesidad de separar la parte del algoritmo que maneja las estructuras de datos de la parte que resuelve el problema en sí. Esto con el fin de garantizar modularidad, facilidad en el mantenimiento y flexibilidad al

cambio de especificación. Esta separación nos lleva a la siguiente ecuación (parafraseando la celebre ecuación de Wirth):

Programa = Estructuras de Datos + Algoritmo de Datos + Algoritmo de Control

donde el Algoritmo de Control es totalmente independiente de las estructuras de datos utilizadas para representar la información. Así, es posible portar la solución a otra representación interna al costo de reescribir únicamente los algoritmos de manejo de datos lo mismo que resolver un problema sin necesidad de comprometer la solución a un tipo especifico de representación. Siguiendo esa misma linea, es posible ver las estructuras de datos como objetos que sólo pueden ser manipulados por un conjunto limitado de operaciones, que los algoritmos de datos se limitan a ejecutar Una clase de objetos, unida a su conjunto de operaciones, forma un Tipo Abstracto de Datos (TAD), que puede ser portado de un programa a otro sin ningún problema. De esta forma se llega a la ecuación

Programa = TADs + Algoritmo de Control

que resume el enfoque del curso desde el punto de vista de diseño. Este enfoque ofrece entre otras ventajas, la posibilidad de reutilizar software (un TAD es perfectamente portable), lo cual lo hace aún más interesante.

En el curso se enseña al estudiante a diseñar, especificar, implantar y utilizar Tipos Abstractos de Datos como parte de sus programas y de sus soluciones, estudiando en detalle los correspondientes a Listas, Pilas, Colas, Arboles y Grafos. Un proyecto tipico del curso es resolver el problema del transporte sobre el TAD Grafo (o sea en terminos de sus operaciones), luego desarrollar varias implementaciones posibles de dicho Tipo Abstracto, calcular la complejidad de cada operación en cada implementación y finalmente probar en el laboratorio los algoritmos diseñados para comparar los resultados teoricos con los practicos. Más información sobre Tipos Abstractos se puede encontrar en [CAR86] [CAR87] [GUT77] [GUT78]

3. El Lenguaje LCC.

El lenguaje en el cual se encuentra basado el laboratorio es una extension del utilizado en los dos primeros cursos de programación. Incluye el nucleo básico de todos los lenguajes imperativos: un comando de asignación, uno condicional (if-fi) y uno de iteración (do-od). En la sintaxis toma muchas construcciones de Pascal, ademas de sus tipos de datos básicos (integer, boolean, char, pointer, array y record). Tiene las siguientes diferencias con respecto a los lenguajes tradicionales:

- no maneja variables globales (es una manera de exigir modularidad y buen diseño)
- se elimina la necesidad del ";" para separar instrucciones
- no requiere la utilización de bloques begin-end
- incluye aserciones como parte del lenguaje (las aserciones son verificadas cada vez que el control del programa pasa sobre ellas)
- permite asignación entre variables estructuradas (asigna por ejemplo un arreglo a otro copiando todo su contenido)
- permite retorno funcional de valores de cualquier tipo, aún estructurado
- no existe programa principal sino una rutina por la que comienza la ejecución (como en C)
- es perfectamente ortogonal; ésto facilita su aprendizaje (no existen excepciones con respecto a las reglas de construcción de expresiones. Si por ejemplo una función retorna un valor de tipo apuntador, es válido escribir f(a)↑.info, mientras que si retorna un arreglo es posible utilizar la expresion f(a)[5], lo cual es incorrecto en muchos lenguajes)
- maneja Tipos Abstractos Genericos de Datos. Es posible por ejemplo definir el TAD Lista[Elemento] que recibe como parámetro el tipo de los elementos que va a contener cada uno de sus objetos abstractos. En ese caso es valida la declaracion:

  var lst1: Lista[**integer**];
  lst2: Lista[**boolean**];

  en la cual se generan dos objetos que se comportan de manera similar, pero cuyo contenido interno corresponde a elementos de diferente tipo.
- maneja funciones polimórficas. Una misma función puede retornar diferentes tipos de datos. Esto es indispensable para el manejo de las operaciones de un TAD, en las cuales el tipo de retorno depende del objeto en si que sea pasado como argumento. Por ejemplo en las siguientes asignaciones:

val2 := InfoLista(lst1,5)

val3 := InfoLista(lst2,5)

donde InfoLista(lst,i) retorna el i-ésimo elemento de la lista lst. corresponden en el primer caso a una asignación de enteros y en el segundo a valores lógicos.

4. El Editor.

El editor del Laboratorio de Estructuras de Datos (LED) permite la escritura y modificación de programas escritos en LCG. Dispone de una interfaz notablemente ágil con los demas módulos del laboratorio, lo cual facilita la depuracion y validacion igualmente se utiliza para generar y editar nuevos TADs

De acuerdo con la metodología expuesta, la escritura de programas debe ceñirse a dos aspectos: primero, obviamente, al lenguaje LCG y segundo, a la disciplina de programación guiada por objetivos. Por estas dos razones, se ha desarrollado un editor de programas y de TADs guiado por la sintaxis del lenguaje, que además de indicar la construcción correcta de cada comando que el programador desea utilizar y de dar a todo programa generado una estructura estándar (reglas fijas de indentación, pretty-printing, etc), garantiza su corrección sintáctica

Para dar soporte al aspecto de la disciplina de programación utilizada, el lenguaje se ha enriquecido, a nivel de su gramática, para permitir la inclusión de aserciones (en terminos de las variables de estado del programa, de ciertos formalismos gráficos básicos y de un conjunto razonable de predicados definidos sobre los TADs de base) y el uso de éstas a manera de comandos. Con respecto a ésto último, no sólo se han incluido las aserciones en la gramatica del lenguaje para permitir su escritura dentro de los programas, sino que también son validables en tiempo de ejecución

Siendo un editor guiado por la sintaxis, valida en cada momento el comando suministrado con respecto a la definición gramatical correspondiente. Cualquier tipo de error en este sentido debe corregirse, para que el editor permita continuar y eventualmente terminar la edición de un programa o rutina

Para escribir una instrucción el programador puede seleccionar de un menú el comando del lenguaje que desea utilizar, siendo la respuesta del sistema el despliegue de la estructura general del comando, ésto es, resaltando la palabra o simbolo clave, escribiendolo en el nivel de indentación que lleva el programa, mostrando los nombres de sus diferentes componentes según lo indican los simbolos no terminales de la producción gramatical correspondiente, y colocando todo en la pantalla de acuerdo con la estructura y reglas de indentación adoptadas (pretty-printing)

Para agilizar el trabajo de las personas familiarizadas con el lenguaje, el editor permite escribir los comandos directamente, sin restricción alguna. Al terminar cada linea el editor la toma, la analiza y de ella extrae, valida y reorganiza, como en el caso anterior el comando o los comandos que la conforman.

Cualquiera que que sea la alternativa utilizada para escribir un comando, el editor presenta los posibles errores cometidos y permite y sugiere formas de corregirlos
Para citar un ejemplo, es posible trabajar bajo la modalidad de declaracion automática, en cuyo caso se reporta todo simbolo no declarado que aparezca como parte de un comando y se sugiere la declaración más adecuada de acuerdo con el papel que juega el símbolo dentro de la expresión, indicando además el punto del programa donde podria incluirse tal declaración. Obviamente existe la alternativa de editar el simbolo, puesto que puede tratarse de un simple error de mecanografía.

Además de implementar las funciones habituales de un editor de texto, tales como insertar, borrar, moverse sobre el texto en cualquier sentido, imprimir, etc., (siempre que su acción resulte en situaciones sintácticamente correctas), el editor permite, entre otras operaciones, recuperar la última versión grabada del texto (i.e., ignorar los últimos cambios hechos), copiar y mover bloques completos de texto siempre y cuando sean correctos y tengan sentido en el sitio donde se copian.

Otro aspecto importante que influye en el funcionamiento del editor, es que la unidad básica de trabajo en el Laboratorio es el procedimiento o rutina, por lo cual sólo se puede estar editando una rutina a la vez y se dispone de operaciones a este nivel para ver, abrir, cerrar y crear rutinas.

Para el trabajo con tipos abstractos, además de la edición que tiene las mismas características descritas para el editor de programas, es posible "instalar" y "desinstalar" TADs en un programa. Estos términos se refieren respectivamente a poner y a retirar del alcance del programa o rutina en edición, implementaciones particulares de los tipos abstractos que maneja. De esta manera es posible, por ejemplo, probar un mismo programa de aplicación (cuyo texto permanece intacto) con diversas implementaciones, lo cual es uno de los objetivos del curso.

El producto de la edición de un texto es el árbol de sintaxis correspondiente al programa LCG escrito por el usuario. Su representación se hace mediante árboles binarios ordenados para almacenar la información de constantes, tipos y rutinas. Las constantes y tipos son globales para todas las rutinas que forman el programa (incluida "main"), las cuales contienen a su vez el árbol sintáctico que las describe.

5. El Manejador de TADs.

El manejador de tipos abstractos de datos es el módulo del sistema que se encarga de tomar las estructuras generadas en la edición de un TAD y de compilar esta descripción. El código que aquí se genera queda a disposición de los diferentes programas, listo para ser "instalado" y utilizado por ellos. Los componentes principales del manejador de TADs son el compilador y el instalador/desinstalador.

Antes de compilar sus definiciones de tipos abstractos, el programador puede depurarlas y probarlas, colocando el conjunto de operaciones como parte de un programa LCG que se encarga de llamarlas. Finalmente, cuando el usuario está satisfecho con el código que ha escrito, hace una llamada al compilador.

Para compilar la definición de un TAD, se ha definido una máquina intermedia a cuyo lenguaje se traducen las estructuras sintácticas que representan el texto del programa LCG: el compilador toma únicamente el conjunto de funciones de la declaración y genera el código correspondiente, que consiste en una secuencia de instrucciones en código intermedio, organizadas en bloques que se almacenan en disco. Esta será la definición del TAD que pueden tomar posteriormente los diferentes programas e instalar en su ambiente de trabajo.

El módulo instalador y desinstalador de TADs está disponible desde diferentes modulos del sistema para hacer más flexible el manejo de los diferentes tipos de datos en el ambiente de laboratorio. Así por ejemplo, es posible utilizarlo desde el editor para crear un programa LCG que utiliza una definición determinada de un TAD, o bien desde el interpretador para alternar dinámicamente diferentes implementaciones de dicho tipo abstracto y comparar su desempeño.

6. El Interpretador Gráfico.

Este módulo ejecuta el código escrito por el usuario presentando el desarrollo de la ejecución a través de una interfaz que permite conocer con mucho detalle diversos aspectos de la ejecución, así como afectarla de varias maneras.

El interpretador presenta de manera simultánea el programa fuente que se está ejecutando, el efecto de la ejecución de las instrucciones sobre el estado del programa y los resultados que produce.

Existe entonces una ventana en la cual se muestra el programa, indicando el comando que se ejecuta en cada momento. Esta ventana se va desplazando sobre el texto fuente a medida que progresa la ejecución, siendo afectado este desplazamiento según el modo de ejecución seleccionado. El "modo programa" permite rastrear el punto de ejecución sólo dentro del programa principal (o rutina main) señalando el comando que se ejecuta o la llamada a procedimiento cuya implementación se está interpretando.

Por defecto, el modo de ejecución es "procedimiento a procedimiento", el cual se desplaza sobre el cuerpo de cada una de las rutinas que se ejecutan. Esto se logra abriendo una nueva ventana de texto por cada nueva rutina, mostrando el desplazamiento del punto de ejecución sobre ella y, finalmente, cerrándola al terminar la ejecución: ésto tiene un ventaja adicional: la de mostrar con el abrir y cerrar de ventanas el cambio de contextos de ejecución que, visualizado de esta manera, ayuda más que muchas explicaciones a la comprensión del concepto.

Finalmente, es posible ver la ejecución instrucción por instrucción (sea avanzando por unidades de tiempo determinadas, sea avanzando a pedido del usuario).

Cualquiera que sea el modo de ejecución, es posible detenerla temporalmente en puntos de interés, mediante la colocación de marcas especiales (señales de "pare") sobre el texto. En estos puntos de detención es posible consultar el estado del programa y evaluar expresiones en el contexto vigente.

En la ventana de resultados es posible ver las modificaciones que van sufriendo los diversos elementos de información que conforman el estado del programa, así como, en conjunción con el depurador, detener la ejecución en ciertos puntos para realizar operaciones o evaluar expresiones en el contexto y estado vigentes. Es de esta misma forma que se puede hacer recuperación de errores sin necesidad de cancelar el ambiente y editar el programa: afectando el estado del programa y retornando el control al interpretador.

Las aserciones son validables durante la ejecución y es posible controlar, a varios niveles, la verificación que de éstas se hace: desde ignorarlas por completo, hasta hacer que la ejecución aborte si una aserción determinada no se satisface.

Es posible también activar un mecanismo de reloj, que permite medir en tiempo virtual la ejecución de los programas y producir estadísticas de la ejecución de varios procedimientos a la vez. La información que se tiene de cada uno corresponde, para cada llamado, a la hora de comienzo y terminación y al número de instrucciones simples ejecutadas.

7. El Depurador Gráfico.

Para facilitar la depuración y mostrar al estudiante la ejecución de su programa de manera gráfica, el laboratorio cuenta con un depurador gráfico. Mediante éste es posible apreciar la manera como se modifican o recorren las estructuras de datos de una aplicación, lo mismo que preguntar por el valor de cualquiera de las variables involucradas en la solución e incluso editarlas. Es posible, por ejemplo, ver en una

ventana la manera como se va balanceando un árbol AVL, o la forma como se va recorriendo un grafo para buscar un camino entre dos de sus vértices.

El depurador funciona en dos modos: interactivo o automático. En el primer modo, el usuario puede preguntar, durante la ejecución, por el valor de cualquier variable o por la representación gráfica de cualquiera de sus estructuras de datos, y puede interactuar con ellas. En el modo automático, el sistema va mostrando cómo afectan las estructuras de datos a medida que progresa la ejecución.

8. El Tutorial.

El carácter de aplicación educativa que tiene el Laboratorio de Estructuras de Datos hace imprescindible la existencia de este módulo, que tiende a satisfacer varios objetivos.

Primero, servir de repaso o "guía rápida" del material del curso. De esta manera un estudiante puede ver una presentación rápida del tema que le interesa y donde el sistema puede proponerle diferentes tipos de ejercicios que refuercen y/o evalúen su comprensión.

Por otra parte, le permite obtener explicación sobre errores cometidos en otras partes del sistema. Efectivamente, al ocurrir errores en módulos como el editor, el depurador o el interpretador, el estudiante tiene la alternativa de entrar al tutorial para obtener explicaciones más detalladas y precisas. Estas explicaciones, que se dan en el mismo contexto donde cometió el error, van acompañadas de ejemplos o de ejercicios que ayudan a aclarar las fallas ocurridas y muestran la forma correcta de expresarse en situaciones similares.

En una situación como la que se acaba de describir, el ambiente de ejecución no se ha perdodo y por lo tanto, una vez aclarada la fuente de los errores, el estudiante puede corregir los errores y retornar el control al interpretador.

Por último, el tutorial puede plantear retos de complejidad creciente que ayuden al estudiante a reforzar los conceptos del curso y a evaluar su propio dominio del tema.

9. Conclusiones

La ausencia de un laboratorio donde se pudieran manejar los conceptos con el nivel de abstracción apropiado, hacía que las sesiones prácticas previstas para el curso de Estructuras de Datos fueran apenas suficientes para la escritura en lenguaje Pascal (y validación por el método de prueba y error!!) de las operaciones básicas de los tipos de datos, logrando en muy pocas oportunidades llegar a la fase de probar programas de aplicación en términos de dichos TADs. Además, y casi más grave aún, no existía ningún tipo de herramienta para expresar (y verificar la correcta aplicación de) la disciplina de especificación y desarrollo descrita.

Tal como está implementado, el laboratorio presenta múltiples ventajas: permite que el estudiante ejecute sus soluciones a los problemas directamente en lenguaje algorítmico evitando la traducción a (y la dependencia de) un determinado lenguaje de programación; le permite utilizar y "ver" los conceptos en el mismo lenguaje gráfico al que está habituado en las sesiones de teoría, comparar el comportamiento de la misma solución implementada bajo diferentes formas y ver cómo su programa realmente satisface los objetivos (tanto los intermedios como el objetivo final) que su método de solución al problema había planteado.

Por último, el Laboratorio de Estructuras de Datos, cuya implementación en máquinas Apple Macintosh Plus comienza a probarse en agosto de 1988, corrige una de las principales deficiencias que tuvo el curso durante algún tiempo: la falta de un ambiente apropiado, en el cual se pudieran realizar prácticas de los temas presentados en las sesiones de clase. Por lo tanto, el Laboratorio de Estructuras de Datos constituye el complemento práctico al texto del curso, "Estructuras de Datos: Un Enfoque desde Tipos Abstractos", comformando así un paquete completo de material didáctico para la enseñanza del tema.

10. Referencias

[BOH86] Bohórquez, J., Cálculo de Programas: Una Metodología Precisa para el Diseño de Programas de Computador, Memo de Investigación #7, Universidad de los Andes, 1986.

[CAR86] Cardoso, R., Diseño e Implementación de Tipos Abstractos de Datos, Memo de Investigación #9, Universidad de los Andes, 1986.

[CAR87] Cardoso, R., Práctica en Tipos Abstractos de Datos, Memo de Investigación #17, Universidad de los Andes, 1987.

[DIJ76] Dijkstra, E.W., A Discipline of Programming, Prentice-Hall, 1976.

[GRI81] Gries, D., The Science of Programming, Springer-Verlag, 1981.

[GUT77] Guttag, J., Abstract Data Types and Software Validation, comm. ACM, vol. 21, num. 12, Diciembre/78.

[GUT77] Guttag, J., Abstract Data Types and the Development of Data Structures, comm. ACM, vol. 20, num. 6, Junio/1977

[MAR88] Mariño, O., Notas para el Curso Introducción a la Informática, Publicaciones de la Universidad de los Andes, Junio/88.

[VQ088] Villalobos, J., Quintero, A., Otero, M. Estructuras de Datos: Un Enfoque desde Tipos Abstractos, Publicaciones de la Universidad de los Andes, Enero/88.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

LUBO-1: UN MODELO DE MAQUINA ABSTRACTA PARA UN PRIMER CURSO UNIVERSITARIO DE PROGRAMACION.

Patricia Pesado. (*)
Maria C. Madoz. (*)
Lia H. Molinari. (*)
Armando E. De Giusti. (**)

(*) Profesora Adjunta del Dpto. de Informática de la Facultad de Ciencias Exactas de la UNLP.

(**) Prof. Titular de la Fac. de Cs. Exactas de la UNLP.
Investigador Independiente del CONICET.

Laboratorio de Investigación y Desarrollo en Informática (LIDI)
Facultad de Ciencias Exactas - Univ. Nac. de La Plata -
50 y 115 - 1er. Piso (1900) La Plata -

RESUMEN:

Se presenta un anàlisis del modelo de màquina abstracta adoptado en 1988 para el curso de Programaciòn de Computadoras (1er. Año de la Licenciatura en Informàtica de la UNLP), que ha sido una evoluciòn de experiencias anteriores realizadas en el Dpto. de Informàtica de la UNLP.-
El aporte màs novedoso es la combinaciòn de primitivas de movimiento (o graficaciòn) con las estructuras de control de un lenguaje procedural "Pascal-like", en un contexto sintàctico preciso.-

INTRODUCCION:

Los contenidos, la metodologia y las herramientas didàcticas utilizables en un primer curso de Programaciòn han sido motivo de numerosos trabajos y experiencias (Ref. 1,2,3,4,5).-
De todos modos una conclusiòn bastante generalizada es la utilidad que se puede obtener del empleo de un modelo abstracto de màquina con un conjunto de instrucciones reducido que pueda extenderse gradualmente. Conceptualmente interesa desprender al alumno de cualquier aspecto "de implementaciòn", tanto en la arquitectura como en el lenguaje, de modo de expresar operaciones abstractas sobre objetos simbòlicos en las que el aspecto primordial sea la correcciòn algoritmica. (Ref. 6,7).-

Enfoques clàsicos y muy conocidos en este sentido han sido los modelos del LOGO (Ref. 8, 9), del robot KAREL (Ref. 10) y de la calculadora evolutiva (Ref. 11). Universidades muy importantes en el mundo los han utilizado en cursos iniciales de programaciòn.-

Sin embargo, desde el punto de vista de la formaciòn esperada para un alumno de un curso de Programaciòn existe un importante "gap" entre el tipo de problemas soluble en el lenguaje reducido de estas màquinas abstractas y la implementaciòn posterior de soluciones en un lenguaje "profesional" tal como PASCAL, C o PROLOG.-

En la carrera de Informàtica de la UNLP, aùn antes de la creaciòn de la Licenciatura en Informàtica (1985), se han realizado una serie de experiencias basadas en diferentes modelos de aprendizaje para el alumno de un primer curso de Programaciòn que oscilaron entre dos extremos:

** El planteo convencional directo "Algoritmo - Lenguaje de Implementaciòn", que incluyò experiencias con FORTRAN, PL/1 y PASCAL.-

** La utilizaciòn de "Màquina abstracta + Seudocòdigo", que no incluìa instrucciones "ejecutables".-

Las mayores criticas al primer enfoque fueron, lògicamente, la distorsiòn en el aprendizaje de los alumnos por el esfuerzo en manejar los detalles del lenguaje involucrado, especialmente en el tratamiento de estructuras de datos.-

Recìprocamente el segundo enfoque, si bien correcto desde el punto de vista de la formaciòn algoritmica, era de difìcil coordinaciòn con los cursos posteriores y ademàs quitaba el aspecto de motivaciòn generado por la realizaciòn de programas "ejecutables" por los alumnos.-

Nuestro trabajo consistió en analizar críticamente las experiencias previas y proponer un modelo, aplicado a partir de 1988, que parte de una máquina abstracta con un lenguaje gradual que evoluciona hacia un PASCAL+ de modo que el alumno culmina el curso realizando programas "reales" en PASCAL sobre una arquitectura tipo PC.-

Un aspecto importante en este planteo fue "extender" los tipos de datos standard del Turbo Pascal de modo que el alumno en una primer etapa manejara naturalmente las estructuras de pilas y colas sin entrar en su implementación utilizando otro tipo primitivo. (Ref. 12).-

LOS OBJETIVOS DE UN PRIMER CURSO DE PROGRAMACION Y SU RELACION CON EL MODELO DE MAQUINA ABSTRACTA:

Una rápida caracterización de los objetivos de un primer curso de Programacion relaciona los mismos con dos contenidos: algorítmica y estructuras de datos.-

El alumno debe aprender las estructuras de control que caracterizan la programación estructurada y su utilización a fin de resolver algoritmos de diferente tipo y complejidad.-

Simultáneamente el alumno incorpora gradualmente las estructuras de datos: datos primitivos (numéricos y lógicos); conjuntos; registros; pilas; colas; arreglos; archivos; listas y por fin una generalización al concepto de tipo de dato abstracto. (Ref. 13).-

Un aspecto importantísimo dentro de los objetivos es el desarrollo de una metodología top-down en el planteo y análisis previo de los problemas que se expresaran algorítmicamente. Interesa especialmente tratar los problemas de modo jerárquico, modularizando en bloques funcionales y tratando de que el tratamiento algorítmico de cada módulo se haga por refinamientos sucesivos.-

Más allá de las justificaciones de simplicidad o fácil comprensión de "la máquina" que pueden dar ejemplos como la tortuga o el robot, este tipo de máquinas abstractas brinda dos características de especial interés en relación con los objetivos mencionados anteriormente:

** El "tipo" de problemas gráficos (dibujos - recorridos) son naturalmente modularizables y el alumno incorpora rápidamente una metodología de trabajo como la deseada.-

** Se puede introducir también naturalmente el concepto de "tipo de datos" asociado con "objeto": por ejemplo un robot como Karel recoge flores y con ellas puede realizar un conjunto de operaciones (depositar, recoger, guardar en su bolsa) y un conjunto de verificaciones (hay flor?, color=, bolsa llena?) que resultan asociadas sin esfuerzo al tipo de datos en cuestión.-

Con este análisis previo, es interesante hacer algunos comentarios sobre el contexto de trabajo y luego detallar las características de LUBO-1.-

EL CONTEXTO:

A- Los alumnos:

Los alumnos que comienzan un curso de programaciòn en la Universidad, constituyen un grupo heterogèneo en conocimientos sobre el tema. Podrìamos hacer una somera clasificaciòn en:

** los que no poseen siquiera conceptos mìnimos de programaciòn.

** los que han realizado un curso sobre algùn lenguaje, incluso algunos alumnos de este grupo poseen experiencia laboral en el tema.

** los que tienen experiencia como "operadores" (entrada de datos, manejo de menùes, juegos).

Los del primer grupo, son alumnos reticentes a la discusiòn de conceptos bàsicos de Informàtica. Habitualmente su visiòn de lo que significa una carrera en esta àrea està muy distorsionada.-
Sin embargo, superada una primer "adaptaciòn"aceptan màs fàcilmente los modelos de abstracciòn, sin impaciencia. Son màs proclives a la especificaciòn, al "porquè" màs que al "còmo".

El segundo grupo es, quizà, el màs conflictivo. Se les ha enseñado a codificar. Es muy difícil que hayan desarrolado un buen estilo de programaciòn con todo lo que ello implica: anàlisis, documentaciòn, uso de estructuras de control adecuadas al algoritmo a utilizar, criterios de correcciòn, etc.-
Y ademàs... estàn seguros de "saber suficiente" del tema.-

El tercer grupo logra una ràpida visualizaciòn en lo que respecta a la importancia en la presentaciòn de un sistema. Conoce de las incompatibilidades surgidas por una especificaciòn de requerimientos ambigua. La confianza con que el operador ejecuta un programa està estrechamente relacionada con una comunicaciòn clara y precisa entre el usuario y el sistema.-
Sin embargo, se trata de alumnos "ansiosos" que creen que toda la Informàtica termina en "un hombre y su terminal". Es muy difícil desarrollar en ellos una valorizaciòn de la abstracciòn y de los criterios generales de los que se derivan aplicaciones.-

Un complemento poco beneficioso es el escaso conocimiento general de los alumnos sobre Lògica y en general sobre Matemàticas: la mayoría de los alumnos (y lamentablemente algunos docentes de ambas disciplinas tambièn...) conciben dos mundos: "el de las matemàticas y el de la computaciòn"; incluso los enfrentan.-
Una de las tareas màs arduas en la enseñanza de programaciòn es la revalorizaciòn de los fundamentos lògicos y matemàticos buscando una formalizaciòn del anàlisis y resoluciòn de problemas.-

B- Los requerimientos de los cursos posteriores:

Una breve síntesis de las expectativas de los cursos posteriores al de Programación de Computadoras puede ser la siguiente:

** Que el alumno maneje adecuadamente las estructuras de control de un lenguaje tipo PASCAL.-

** Que conozca la noción de tipo de datos y tenga capacidad de abstracción de nuevos tipos, tratados como objetos y más allá de los detalles particulares de implementación.-

** Que haya desarrollado una metodología mínima de análisis de problemas top-down y que sepa aplicar refinamientos sucesivos ante un problema relativamente complejo.-

** Que tenga nociones conceptuales de corrección y que relacione claramente el proceso de especificación y documentación con el de prueba de corrección.-

** Que entienda los parámetros de medida de eficiencia de un algoritmo y pueda comparar alternativas algorítmicas.-

Otro aspecto importante a tener en cuenta es que un alto porcentaje de los alumnos opta por la salida intermedia (tres años), lo que significa que al menos debe superar primer año desarrollando aptitudes para el análisis y resolución de problemas "in the small" en un lenguaje ejecutable.-

Con este marco, las orientaciones lógicas fueron:

1- Adoptar una máquina abstracta con capacidad de recorrido o graficación, pero con un lenguaje orientado a PASCAL.-

2- Desarrollar rápidamente la evolución de la máquina hacia la resolución de problemas más abstractos que no requieran el empleo de las primitivas gráficas o de recorrido.-

3- Incorporar al Pascal a utilizar por los alumnos (Turbo Pascal sobre microcomputadora) una serie de bibliotecas desarrolladas por los auxiliares de la cátedra, a fin de reducir al mínimo los problemas "de implementación" respecto de estructuras de datos clásicas como pilas y colas.-

ANALISIS DE ALTERNATIVAS POSIBLES:

A- Tortuga/Lenguaje LOGO:

A nuestro juicio, las caracteristicas màs destacables del lenguaje LOGO estàn fundadas en los siguientes aspectos:

* Permite resolver problemas de distinta complejidad.
* Fomenta la modularidad.
* Permite estructuraciòn.
* Maneja naturalmente la recursividad.

Naturalmente el alumno de LOGO se inclina a escribir procedimientos asociados univocamente a las componentes funcionales del problema en cuestiòn. Esto implica evidentemente un mètodo de desarrollo top-down para el anàlisis y el diseño del programa. Por lo tanto la soluciòn de un problema se logra fàcilmente por la combinaciòn de los procedimientos desarrollados obteniendo como resultado un programa ampliamente modular.-

Por otro lado es inmediato advertir la posibilidad de resolver desde problemas muy simples hasta problemas muy complicados, introduciendo gradualmente herramientas que permiten atacar un primer nivel de manejo de la tortuga, luego el manejo de actores, siguiendo con la incorporaciòn de elementos adicionales (sonido, tiempo, etc.). Uno de los puntos de motivaciòn màs importantes para el uso de la tortuga es la ràpida visualizaciòn del resultado obtenido al correr los procedimientos desarrollados.-

Se ve claramente entonces que es posible comenzar con un subconjunto de posibilidades reducidas para iniciar los primeros pasos y que ese subconjunto es fàcilmente utilizable.-

Respecto a la construcciòn de los algoritmos es posible lograr una estructuraciòn màs o menos rudimentaria utilizando bloques del tipo IF THEN ELSE y DO WHILE.-

Quizàs en el tratamiento de la recursividad se encuentren las mayores bondades del lenguaje porque permite el llamado reiterado a un procedimiento con paso de parâmetros, evitando cualquier complicaciòn de implementaciòn para el usuario.-

Hasta aqui hemos marcado cuatro aspectos que representan las mayores cualidades del lenguaje, pero no debemos olvidar que existen algunos usos desaconsejados para resolver con LOGO:

- No permite manejar operaciones complejas con datos compuestos.
- No tiene un buen manejo de archivos.
- No facilita las operaciones de E/S masiva de datos.
- Los mensajes de error son demasiado genèricos y dificultan la autocorrecciòn del principiante.
- No permite una buena autodocumentaciòn porque no respeta la indentaciòn y los comentarios "en linea" estàn restringidos.
- La falta de indentaciòn provoca que en el còdigo, no sean legibles las diferencias de nivel por anidamientos sucesivos.

Por otra parte las implementaciones de LOGO son bastante dependientes de la màquina que lo soporta (sobre todo en el manejo gràfico de actores), por lo que es difícil tener un "standard" suficientemente general.-

El punto crítico en la decisiòn de utilizar el LOGO como lenguaje introductorio es el gran cambio al que debe someterse el alumno para pasar a un lenguaje de mayor potencialidad.-

Ya que obviamente el LOGO puede ser utilizado sòlo para que los alumnos hagan sus primeras armas y no de una manera profesional, es necesario crear la transiciòn y dada la estructura del LOGO (derivado del LISP), es sumamente difícil "enganchar" con cursos basados en lenguajes procedurales tipo PASCAL.-

Resumiendo, sin negar las cualidades didàcticas, pensamos que esmàs útil volcarse a una herramienta tipo màquina abstracta pero con màs puntos de conecciòn con un lenguaje "a la PASCAL".-

Robot/Seudocòdigo:

La alternativa natural en esta instancia es utilizar una màquina abstracta tipo ROBOT, con un lenguaje de expresiòn de algoritmos màs potente basado en un seudocòdigo o en un lenguaje de especificaciòn tipo PDL (Ref. 14).-

Las ventajas màs significativas de esta soluciòn es que, al no tener un compilador al que ajustar la sintaxis, puede hacerse un tratamiento muy general de las estructuras de control y tambièn de las de datos, haciendo coincidir la especificaciòn con la codificaciòn.-

Sin embargo estas mismas ventajas constituyeron dificultades recogidas de las experiencias en 1986 y 1987 en Programaciòn:

** Un curso de 600 alumnos tiene unos 40 docentes auxiliares. El grado de libertad que da la utilizaciòn de un seudocòdigo sin una sintaxis precisa que se valide a través de un compilador produjo numerosas "expresiones alternativas" generadas por los diferentes docentes.
Un caso especialmente conflictivo fue el tratamiento de problemas con tipos de datos como pilas, colas y listas.-

** El seudocòdigo permite la creaciòn un tanto libre de "tipos standard", para los que valen todas las operaciones pre-existentes, sin restricciones naturales. Esto nos aleja del concepto deseado de objeto, con un TIPO definido por el usuario y un conjunto de operaciones validas y condiciones verificables. Sin ser completa, el empleo del concepto de TIPO y el encapsulamiento que pueden dar las unidades en las versiones màs avanzadas del PASCAL, llevan al alumno a una mejor abstracciòn de datos.-

** La expresiòn algorítmica en seudocòdigo no ejecutable sobre una màquina real mantiene el "gap" que va desde el anàlisis a la puesta a punto de un programa y no colabora con el desarrollo de una idea unificada de lo que es el ciclo de vida del software. Màs aùn: se pierde el fuerte efecto motivador del trabajo interactivo, que incluye etapas muy interesantes como el manejo de un editor orientado a la sintaxis del lenguaje; la detecciòn de errores sintàcticos y semànticos e incluso la re-adaptaciòn de un mòdulo previamente declarado "correcto".-

Calculadora/ Calculadora Programable:

Un modelo muy interesante por su simplicidad y su inmediata comprensiòn por el alumno es el de "Calculadora" que gradualmente incrementa su capacidad desde las 4 operaciones y entrada/salida a la programaciòn de estructuras de control.-

Sin embargo, fue descartado por:

** Existe una contradicciòn entre el modelo propuesto, que incorpore estructuras de control "a la PASCAL" y las calculadoras programables realmente existentes, que en general incorporan primitivas "a la BASIC".-

** Es muy difícil salir de los ejemplos de cálculo matemàtico y especialmente numèrico, encontrándonos con la barrera de los conocimientos previos de los alumnos.-

** Los tipos de datos "no numèricos", y màs aùn los estructurados resultan poco naturales en el marco de las aplicaciones de una calculadora. Es así que superada la etapa de los datos primitivos, resulta muy difícil encontrar los ejemplos "de motivaciòn" que justifique la creaciòn o incorporaciòn de nuevos tipos y màs aùn de nuevas operaciones y verificaciones.-

Este anàlisis de alternativas nos llevò a la conclusiòn de que el camino correcto era profundizar las ideas del Robot + Seudocòdigo, acercando dicho seudocòdigo lo màs posible a PASCAL (si era necesario modificando aspectos del PASCAL a utilizar por los alumnos) y además llegamos a la convicciòn de que el proceso de generaciòn y ejecuciòn real de programas por los alumnos sobre màquinas convencionales (en lo posible interactivas) incrementaría su motivaciòn y por ende mejoraría los resultados generales del curso.-

Así nace la idea de especificar LUBO-1, un robot con dos niveles de instrucciones: las de movimiento y tratamiento de objetos del mundo real y las de procesamiento convencional "a la PASCAL".-

UN BREVE ANALISIS DE LUBO-1:

Basado en las ideas de varios docentes del Dto. de Informàtica, se dearrollò una especificaciòn de màquina abstracta, que hemos llamado LUBO-1.-

Conceptualmente, el alumno dispone de una calculadora + movimientos + comunicaciòn con el mundo que rodea al robot.-
Se dispone de un conjunto de instrucciones que permiten cinco capacidades bàsicas en una primer fase:

* moverse.
* orientarse.
* reconocer objetos, recogerlos, depositarlos.
* realizar càlculos elementales.
* leer e informar.

El alumno resolverà problemas:

- usando instrucciones simples.
- construyendo nuevas instrucciones.
- manejando señales de control.
- ordenando càlculos.

LUBO-1 se mueve en un contexto definido como una ciudad de 100 calles horizontales y 100 avenidas verticales.
Cada paso del LUBO-1 equivale a una cuadra.
Establecido en un punto de la ciudad LUBO-1 puede ejecutar sobre el tipo de objetos que reconoce funciones tales como recogerlos, depositarlos y contarlos.
LUBO-1 tambièn cuenta con un reloj que sirve como elemento de control temporal para las acciones.

Por lo tanto, existen varios niveles de problemas:
- recorridos.
- recorridos con obstàculos.
- càlculos simples.
- càlculos con informes.
- recorridos con càlculos e informes.

A su vez se propicia la tècnica de descomposiciòn del problema en problemas màs simples mediante el fàcil manejo de mòdulos bien encapsulables.
Si bien la definiciòn primitiva del lenguaje se ciñe a un conjunto reducido de instrucciones para lograr un uso adecuado de las primeras herramientas, se agregan gradualmente nuevas capacidades que se incorporan al conjunto de definiciòn, permitiendo aumentar el alcance de los problemas a resolver.-

En esta primer fase, constituida por el curso de Ingreso y las primeras 4 semanas de Programaciòn de Computadoras, el alumno aprende a utilizar las estructuras de control (decisiòn, selecciòn, repetitivas e iterativas) y simultàneamente desarrolla una tècnica de descomposiciòn funcional de los problemas planteados.-
En todo momento se respeta una sintaxis PASCAL, incluso con la estructura de Programa con su secciòn de encabezamiento la de declaraciones y la de implementaciòn de los algoritmos.-

Una característica importante de esta primer fase es el manejo de dos tipos de datos: numéricos enteros y booleanos, conjuntamente con "objetos" como claveles y caramelos que LUBO-1 puede transportar en su "bolsa" (almacenamiento auxiliar) y sobre los que hay un conjunto limitado y específico de primitivas y tests válidos.-

La ciudad constituye un interesante contexto pues obliga a desarrollar algoritmos de recorrido que tengan en cuenta "obstáculos" (nuevamente condiciones verificables mediante primitivas relacionadas con una variable boolean) y naturalmente introduce al alumno en los conceptos de corrección y de eficiencia.-

Nótese también que los modelos "de recorrido" pueden adquirir una complejidad igual o mayor que la obtenible con los actores del LOGO al introducirnos en juegos tales como los encierros entre obstáculos o los laberintos en los que se puede trabajar con el robot.-

También resulta natural la introducción de la recursividad, y sobre todo el concepto de "informe" que establece el sentido de las instrucciones de E/S que caracterizan a todos los lenguajes de programación. En este punto un "robot" que hace recorridos es más natural que lea,calcule e informe que un "lapiz que dibuja".-

Una vez superada la primer fase, la incorporación de más instrucciones y sobre todo nuevos tipos de datos estructurados va marcando una reducción en la necesidad del soporte de aprendizaje que constituye la máquina abstracta; sin embargo muchas explicaciones introductorias se hacen utilizando el modelo de LUBO-1:

** Por ejemplo la introducción al tipo de datos COLA, creando ejemplos con robots que tienen que acceder a un mismo recurso en un punto de la ciudad.-

** El planteo de los algoritmos clásicos de manejo de caracteres, en los que se convierte a LUBO-1 en una máquina de procesamiento de caracteres (y posteriormente de tratamiento de textos).-

** La asociación de los conceptos de posición en X y en Y (calle-avenida de la ciudad de LUBO-1) con los índices de arreglos y matrices al introducir estos tipos de datos.-

ALGUNOS RESULTADOS:

Si bien es prematuro dar conclusiones a la experiencia con LUBO-1 realizada en 1988, el criterio general es que los alumnos han respondido de un modo superior en cuanto a interés y motivación: el horizonte que tienen al trabajar en un lenguaje "profesional" ha colaborado para obtener una mayor participación en los primeros meses de su formación universitaria.-

Claramente los auxiliares docentes (por otra parte en general alumnos avanzados con muy buen dominio de PASCAL) han homogeneizado las respuestas sobre sintaxis y semántica de las instrucciones y esto también contribuye en la perfomance de los alumnos.-

El riesgo siempre latente es no caer en convertir un curso inicial de Programaciòn en otro "de lenguaje PASCAL"; por ello tambièn es importante trabajar con una màquina abstracta que brinde prestaciones adicionales, sobre las cuales se pueden plantear aspectos algoritmicos generales que excedan las prestaciones del lenguaje de referencia (por ejemplo modelizaciòn o simulaciòn de fenòmenos de tiempo real utilizando la variable "tiempo" de que dispone LUBO-1).-

CONCLUSIONES:

Se ha presentado una experiencia de definiciòn de màquina abstracta y lenguaje para un primer curso de Programaciòn, adecuado al contexto particulàr de la Licenciatura en Informàtica de la UNLP, que es la consecuencia natural de una serie de experiencias realizadas en el mismo àmbito en los ùltimos años.-

Creemos que el tema del enfoque y las herramientas para el aprendizaje por los alumnos en un curso de este nivel sigue siendo un tema abierto de discusiòn en nuestro paìs y en el mundo, con lo que este trabajo no intenta màs que hacer un pequeño aporte al tema.-

BIBLIOGRAFIA:

1 - SECYT.
"Formaciòn de recursos humanos en Informàtica"
Documento SID num. 8, 1985.-

2 - ACM recommended curricula for Computer Science and Information Processing programas in colleges and universities. 1968-1981.
ACM Press, 1982.-

3 - J. B. ROGERS.
" Teaching beginners to program: some cognitive considerations"
Proc. The Computer: Extension of the human mind.
ERIC 1982.

4 - E. TORRERO.
"Education:The challenges are classic"
IEEE Spectrum, Nov. 1984.-

5 - DE GIUSTI, ROSSI, DIAZ.
"Educaciòn Informàtica: lenguajes de programaciòn y docentes"
Anales del Simposio Internacional sobre Informàtica y Educaciòn.
Tucumàn, Mayo 1984.-

6 - R. HOLT y otros.
"SP/K: A system for teaching computer programming"
Communication of the ACM. May 1977.-

7 - M. MULDER, J. DALPHIN.
"Computer science program requirements and accreditation".
COMPUTER IEEE. Vol 17, num. 4, Apr 1984.-

8 - S. PAPERT.
"Desafìo a la mente".
Ed. Galàpago, Bs. As. 1981.-

9 - DE GIUSTI A.
"El LOGO como lenguaje en educaciòn. Anàlisis crìtico".
Anales de la X Conferencia Latinoamericana de Informàtica.
Valparaiso (Chile) 1984.-

10- R. E. PATTIS.
"Karel The Robot. A gentle introduction to the art of programming with Pascal". Wiley 1981.

11- S. J. GARLAND.
"Introduction to computer science with applications in Pascal".
Addison Wesley, 1986.

12- Cátedra Programación de Computadoras 1988.
"Extensiones de Turbo Pascal para el manejo de estructuras de datos dinámicas"
Informe Técnico interno. La Plata, 1988.-

13- Programa del curso de Programación de Computadoras.
Departamento de Informática. UNLP. 1988.-

14- D. GRIES.
"The science of programming".
Springer Verlag, 1981.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa.
Buenos Aires, Setiembre 1988.

ESTRATEGIAS PARA EL CONTROL DE SEÑALES DE TRAFICO URBANO

SERGIO NICOLAS DI GERONIMO
MARIBEL FERNANDEZ ROMERO

E.S.L.A.I

SUMARIO

Este trabajo examina el problema de optimización del flujo de vehículos en una red de tráfico urbano, aplicando control "off-line" y "on-line".

En la primera parte se estudian los problemas de la hora pico. Se presentan modelos matemáticos y se desarrollan estrategias de control, primero para cruces simples de calles y luego para redes urbanas. En la segunda parte se realiza un análisis similar para las horas de tráfico normal.

Por último se presenta una generalización de los modelos y estrategias propuestas, considerando cruces de avenidas y calles de varios carriles.

SERGIO NICOLAS DI GERONIMO _ 5º año de la Licenciatura en Informática - E.S.L.A.I. - Argentina, dirección : E.S.L.A.I. , casilla de correo 3193 - (1000) Correo Central, Bs. As. Argentina, teléfonos: (021) 870216 - (0223)3307

MARIBEL FERNANDEZ ROMERO _ 5º año de la Licenciatura en Informática - E.S.L.A.I. - Argentina, dirección : E.S.L.A.I. , casilla de correo 3193 - (1000) Correo Central, Bs. As. Argentina, teléfonos: (021) 870216 - (0223)3307

1_INTRODUCCION

Los automóviles son una gran fuente de contaminación en las ciudades. Por eso se intenta mejorar el flujo de vehículos, ya que esto decrementa el desgaste de los mismos y reduce la contaminación ambiental. Se pueden conseguir mejoras en el flujo, decrementando el tiempo de espera en los cruces con semáforos. Hasta el presente los esquemas utilizados para computar secuencias de señales de control, se basan en datos históricos ("off-line"), por lo cual no pueden manejar alteraciones imprevistas en el tránsito. Sin embargo, existen facilidades para monitorear en tiempo real la mayoría de las variables involucradas en el control de tráfico, lo que hace pensar que el control "on-line" podría disminuir las demoras.

Desde el punto de vista del control de tráfico, hay dos problemas diferentes:

* la hora "pico" (período sobresaturado): Aquella en que en la mayoría de las intersecciones con semáforos, las colas de vehículos en espera no desaparecen al finalizar la luz verde.
* la hora "no pico" (período no saturado): Aquella en que en la mayoría de los cruces con semáforos, no quedan vehículos al finalizar la luz verde.

El primero es más crítico, pues es cuando los servicios están sobrecargados y las demoras son muy grandes. En este sentido el problema del período no saturado es menos importante.

En este trabajo se analizan los dos problemas, considerando primero cruces simples y luego su generalización para casos más complicado. A partir de modelos matemáticos, que representan los cruces en los distintos períodos, se desarrollan estrategias de control que minimizan las demoras, tanto para el período sobresaturado como para el no saturado.

El siguiente cuadro esquematiza el análisis del problema hecho a lo largo de este trabajo:

* Periodo Sobresaturado _ Modelo
  _ Estrategias de control off-line y on-line

* Periodo No saturado _ Modelo
  _ Estrategias de control on-line

En todos los casos se analizó primero un cruce simple y luego una red.

Además de lo esquematizado en el cuadro anterior se abordó una generalización del modelo para avenidas y calles con varios carriles.

2 _ PERIODO SOBRESATURADO

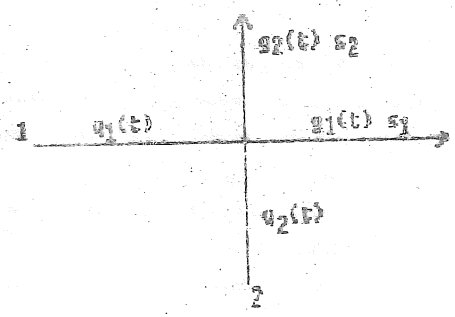
Se llama **período de tráfico sobresaturado** (o simplemente período sobresaturado) a las horas en que en la mayoría de los cruces señalizados, las colas de vehículos en espera de luz verde son tan grandes que el tiempo que dura la misma no es suficiente para que todos crucen (las colas de espera no desaparecen al finalizar la luz verde).

2# _ MODELOS

CRUCE SIMPLE SOBRESATURADO

Bajo la hipótesis de que las colas de espera no desaparecen al finalizar la luz verde, se propone un modelo matemático para este tipo de cruces. [SIN-73]

Por simplicidad se analizan primero cruces de calles de una sola mano, como el que muestra la figura:



$q_i(t)$ indica la tasa de arribo de vehículos (número de vehículos por minuto) en la dirección i

s_i flujo de saturación en la dirección i: máximo número de vehículos por ciclo que pueden pasar a través de la intersección en la dirección i, si se dispone del máximo de luz verde (todo el ciclo). Es una constante para todos los ciclos.

ciclo comprende desde que se prende la luz verde en una dirección, hasta que lo vuelve a hacer en la misma dirección.

C es la duración del ciclo. Se asume constante (Esta suposición se basa en el experimento de Webster [SIN-73] que mostró que variar la duración del ciclo no disminuye las demoras).

$C = G_1 + G_2 + L$

donde

G_i es el tiempo de luz verde en la dirección i

L es la suma del tiempo de luz amarilla en ambas direcciones

$g_i(t)$ es la tasa promedio de salida de vehículos por ciclo, en la dirección i (o sea: número de vehículos por C)

Las variables de control $u_i(t)$ están definidas como la fracción de luz verde en la dirección i.

$$u_i(t) = G_i / C \qquad i=1,2$$

Por ser el cruce sobresaturado las colas nunca desaparecen, siempre hay vehículos para pasar, por lo tanto es válida la relación:

$$u_i(t) = g_i(t) / s_i \qquad i=1,2$$

Por razones de seguridad es necesario tener un mínimo y un máximo de luz verde por ciclo, lo cual impone restricciones a las variables de control:

(1) $\quad U_i <= u_i(t) <= V_i \qquad i=1,2$

$0 < U_i, V_i < 1$ puesto que $V_i = (\text{Max } G_i) / C$

$U_i = (\text{Min } G_i) / C$

Un resultado que se obtiene a partir de la definición de las variables de control es el siguiente:

(2) $\quad u_1 + u_2 = (G_1 + G_2) / C = (C - L) / C = 1 - L / C = EG$ ("efective green")

De acuerdo a esto basta con tener una sola variable de control por intersección. (Si $u_1 = u$ entonces $u_2 = EG - u$)

Como las demoras están dadas por el tiempo pasado en cola se puede representar el estado del cruce en cada ciclo por las longitudes de las colas en cada dirección. El vector de estado X se define como: $X(k) = (X_1(k), X2(k))$

donde $X_i(k)$ es la longitud de la cola en la dirección i, en el ciclo k.

La evolución dinámica de las colas se describe por la ecuación:

$$X_i(k+1) = X_i(k) + q_i(k) - g_i(k) \qquad i=1,2$$

donde $g_i(k)$ está definida como antes (tasa de salidas en el ciclo k)

$q_i(k)$ es la cantidad de arribos en el ciclo k.

Como $g_i(k) = s_i * u_i(k)$ entonces:

(3) $\quad X_i(k+1) = X_i(k) + q_i(k) - (s_i * u_i(k))$

Además se imponen restricciones sobre la longitud de las colas:

(4) $\quad 0 < X_i(k) < X_i \qquad i=1,2.$

OPTIMIZACION EN UN CRUCE SIMPLE SOBRESATURADO

Como el objetivo de cualquier esquema de control en la hora pico es reducir las demoras, y éstas se miden por el tiempo de espera en cola, la suma de las longitudes de las colas es una medida apropiada de la performance de una secuencia de control. Por esto razón, se tratará de minimizar la sumatoria de longitudes de cola. Un objetivo secundario es el grado de utilización de los servicios. De acuerdo a esto la función objetivo es:

$$(5) \qquad I = \min_{u_1, u_2} \sum_{k=1}^{N} \sum_{i=1}^{2} K1 * X_i(k) + K2 * s_i * (u_i(k) - V_i)$$

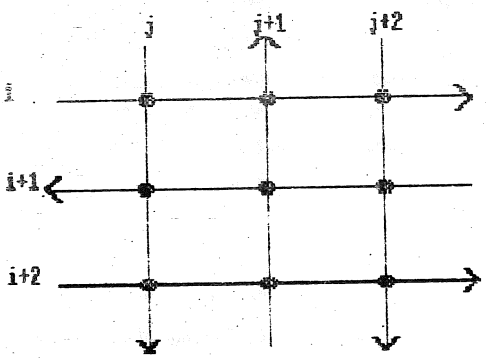
donde

- N es la cantidad de ciclos a optimizar
- K1,K2 son factores de ponderación (en este caso K2=0, puesto que para cruces sobresaturados el desperdicio es nulo)
- V_i es el máximo porcentaje de luz verde en la dirección i

El problema consiste en encontrar la secuencia de valores para las variables de control u_1 y u_2 que satisfagan (5), cumpliendo la ecuación (3) y teniendo en cuenta las restricciones dadas por las desigualdades (1), (2) y (4).

REDES SOBRESATURADAS

Las redes están formadas por intersecciones y calles que las interconectan, en una cascada de dos dimensiones. Se asume que los cruces son de calles de una mano y que los vehículos no doblan. Las calles que interconectan los cruces son modeladas como elementos de demora.



Como muestra la figura, la salida de la calle i del cruce (i, j) coincide con la entrada del cruce siguiente, luego de p períodos de demora (p >= 1).

$$g_{i\,j}(k) = s_{i\,j} * u_{i\,j}(k) \quad \text{luego} \quad q_{i\,j+1}(k+p) = s_{i\,j} * u_{i\,j}(k)$$

El estado de la red en cada ciclo estará representado por una matriz, en la cual la posición i j represente el estado del cruce (i, j) en dicho ciclo, por medio de las longitudes de las colas en cada dirección de la intersección. Dicha matriz tiene la siguiente forma:

$$X(k) = \begin{bmatrix} X_{1,1}(k) & \cdots & X_{1,n}(k) \\ \vdots & & \vdots \\ X_{n,1}(k) & \cdots & X_{n,n}(k) \end{bmatrix}$$

donde

$X_{i\,j}(k) = (X_{i\,j(i)}(k),\ X_{i\,j(j)}(k))$

siendo

$X_{i\,j(z)}(k)$ la longitud de la cola en la intersección (i, j) en la dirección z, en el ciclo k (z=i,,j).

La evolución dinámica de las colas está dada por la siguiente ecuación:

(6) $X(k+1) = (X(k) - (S \# U(k)) + C) [+] ((B_1 \# U(k-1)) + \ldots\ldots + (B_p \# U(k-p)))$ $k=0,1,\ldots N-1$

donde

B_j j=1, ... p son matrices de ponderación de control

S matriz en la cual cada posición indica el flujo de saturación de la calle de la intersección correspondiente

U(k) es la matriz de control (cada elemento es un par u_i,u_j de control para cada dirección de la intersección (i, j))

U(k-j) j=1, ... p son usados para simular las demoras por las calles que interconectan los cruces

C matriz de entradas desde fuera de la red

\+ es la suma común de matrices, componente a componente

\# es un producto de matrices componente a componente

[+] es una suma de matrices pero con la siguiente salvedad:

sea $A = B [+] C$ luego

$a_{i\,j} = b_{i\,j} + c_{i\,j-1}$ si hay una calle con sentido (i, j-1) (i, j) ó

$a_{i\,j} = b_{i\,j} + c_{i\,j+1}$ si hay una calle con sentido (i, j+1) (i, j)

Esto es teniendo en cuenta los cruces en la dirección i. Lo mismo vale para los cruces en la dirección j.

Las matrices de estado y de control están sujetas a las siguientes restricciones:

(7) $\underline{X} <= X(k) <= \overline{X}$

(8) $\underline{U} <= U(k) <= \overline{Y}$

OPTIMIZACION EN REDES SOBRESATURADAS

Al igual que en el caso de un único cruce, el costo está determinado por la suma de las longitudes de las colas en cada intersección. La función de costo es:

$$(9) \qquad I = \min_{U(0),\ldots,U(N-1)} \sum_{k=0}^{N-1} \| X(k) \|$$

donde N es la cantidad de ciclos que se van a optimizar.

Luego, el problema de optimización en una red sobresaturada consiste en resolver la ecuación (9) sujeta a las restricciones (6), (7) y (8).

Observación: Desarrollando la ecuación (9), es posible cancelar algunos términos, pero la función resultante no es más fácil de usar, por lo tanto se trabajará con la función objetivo (9).

2.2 _ ESTRATEGIAS DE CONTROL

2.2.1 _ CONTROL OFF-LINE

CRUCE SIMPLE SOBRESATURADO

El problema consiste en encontrar la secuencia óptima de tiempos de luz verde para N ciclos. El objetivo es:

$$\min_{u_1,u_2} \sum_{i=1}^{2} \sum_{k=1}^{N} X_i(k)$$

sujeto a las restricciones dadas por las ecuaciones (1), (3), (4).

Se verá si es posible representar el problema como búsqueda de caminos en un grafo. Primero es necesario analizar si es un problema discreto, para ello se han realizado medidas de

_duración del ciclo de un semáforo: 60 a 70 segundos.

_duración de la luz amarilla: 3 a 4 segundos.

_cotas del tiempo de luz verde: mínimo 20 segundos, máximo 40 seg.

_tiempo que tarda un auto en cruzar: 4 a 5 segundos.

De éstas se deduce que es posible representarlo con un grafo. La cantidad de sucesores de un nodo está dada por los posibles tiempos de luz verde en cada dirección, y debido a que un coche tarda 4 o 5 segundos en cruzar, los tiempos de luz verde pueden variar "discretamente" (de a 3 o 4 segundos). Entonces, la cantidad de sucesores es 6 o 7 (según se varíe de a 3 o de a 4 segundos).

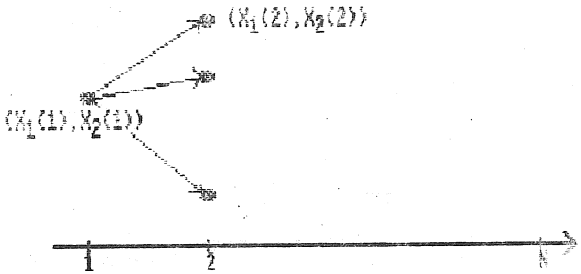
Por lo tanto el problema de encontrar la secuencia óptima de tiempos de luz verde puede representarse con un grafo en el que cada nodo tiene 6 sucesores, y los nodos están dispuestos en "camadas" (o etapas) correspondientes a los ciclos. El número de etapas es:

(duración del período de sobresaturación) / C

DESCRIPCION DEL GRAFO

Cada nodo representa un estado $(X_1(k),X_2(k))$. Cada arco representa un control $(u_1(k),u_2(k))$, o solamente $u_1(k)$, ya que $u_2(k)$ se calcula a partir de $u_1(k)$ con la ecuación (2).

Hay un arco por cada vector de control factible (que cumpla las restricciones). El estado al cual pase está dado por la ecuación (3).

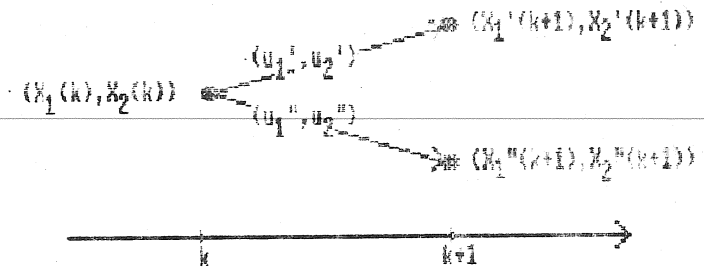


PROPIEDADES

1) No hay arcos múltiples.

Dem.

Se quiere probar que con dos controles distintos no se puede pasar al mismo estado.



Supongamos (u_1',u_2'), (u_1'',u_2'') distintos.

$q_i(k)$ = cantidad de arribos en el período k en la dirección i, por lo tanto es independiente de u' y u''

De acuerdo a la ecuación (3)

$$X_i'(k+1) = X_i(k) + q_i(k) - s_i * u_i'(k)$$

$$X_i''(k+1) = X_i(k) + q_i(k) - s_i * u_i''(k)$$

Por lo tanto u_i' *distinto de* u_i'' implica X_i' *distinto de* X_i'', lo cual demuestra que no hay arcos múltiples. LQQD.

2) El grafo no es un árbol.

Dem.

- Es posible llegar al mismo estado por dos caminos distintos (con dos secuencias de control distintas).

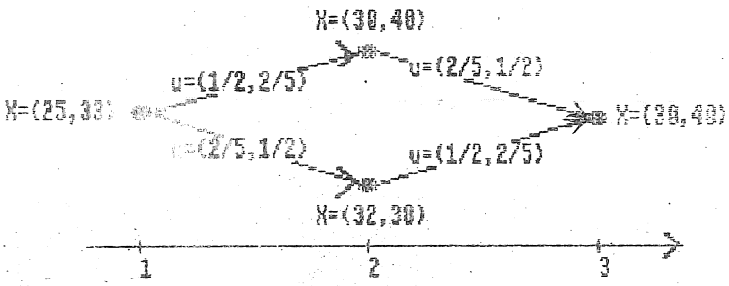
Ej.

C = 60 seg, L = 6 seg, $s_1 = s_2 = 20$, $20/60 < u_1, u_2 < 40/60$

q(1) = (15,15)

q(2) = (8,10)

EG = 1 - L/C = 9/10, $u_2 = 9/10 - u_1$



Los valores verifican $X_i(k+1) = X_i(k) + q_i(k) - s_i * u_i(k)$:

Pasaje de 1 a 2

(30,40) = (25,33) + (15,15) - (20,20)*(1/2,2/5)

(32,38) = (25,33) + (15,15) - (20,20)*(2/5,1/2)

Pasaje de 2 a 3

(30,40) = (30,40) + (8,10) - (20,20)*(2/5,1/2)

(30,40) = (32,38) + (8,10) - (20,20)*(1/2,2/5)

LQQD.

Algoritmo

El problema consiste en encontrar el camino mínimo en el grafo. Para esto se necesita la lista de arribos para cada ciclo (k=1,...,N), los cuales se suponen conocidos a priori (se usan datos históricos, o valores dados por alguna distribución de probabilidades).

Para hallar el óptimo como camino en el grafo, se aplican técnicas de programación heurística, las cuales se basan en el uso de información (heurística) que facilite la búsqueda. Concretamente se utiliza el algoritmo de búsqueda de camino mínimo en grafos A* [NIL-71]. La heurística h es un estimado del costo futuro del camino:

$$h(n_k) = \sum_{i=1}^{2} [(N-k) * X_i(k) + (\sum_{j=k}^{N-1} q_i(j) * (N-j)) - s_i * V_i * ((N-k)*(N-k+1)/2)]$$

donde

n_k es un nodo del grafo en la etapa k.

$q_i(k)$ son los arribos esperados en la dirección i para el ciclo k.

V_i es el máximo tiempo de luz verde que se le puede dar a la dirección i, de acuerdo a la restricción **(1)**.

El estimado de un nodo de la etapa k es la suma de los colas en las etapas restantes suponiendo que a dicho cruce se le da el máximo de luz verde en cada dirección.

Un estimado h se dice admisible si $h(n) <= h(n)$ para todo n, siendo h(n) el costo real del camino a partir del nodo n. Es conveniente elegir un estimado admisible puesto que en este caso el algoritmo A* termina encontrando un camino óptimo, si existe **[NIL-71]**.

El estimado h es admisible (es subestimado).

Dem.

$$h(n_k) = X_1(k+1) + \dots + X_1(N) + X_2(k+1) + \dots + X_2(N) =$$

$$= (N-k) * X_1(k) + \sum_{j=k}^{N-1} q_1(j) * (N-j) - \sum_{j=k}^{N-1} s_1 * u_1(j) * (N-j) + (N-k) * X_2(k) +$$

$$+ \sum_{j=k}^{N-1} q_2(j) * (N-j) - \sum_{j=k}^{N-1} s_2 * u_2(j) * (N-j)$$

$$h(n_k) - h(n_k) = s_1 * V_1 * \sum_{j=k}^{N-1} (N-j) + s_2 * V_2 * \sum_{j=k}^{N-1} (N-j) -$$

$$\sum_{j=k}^{N-1} s_1 * u_1(j) * (N-j) - \sum_{j=k}^{N-1} s_2 * u_2(j) * (N-j) =$$

$$= \sum_{j=k}^{N-1} (N-j) * s_1 * (V_1 - u_1(j)) + \sum_{j=k}^{N-1} (N-j) * s_2 * (V_2 - u_2(j))$$

Por la restricción **(1)**, $V_1 - u_1(j) >= 0$ para todo j

$V_2 - u_2(j) >= 0$ para todo j

por lo tanto

$h(n_k) - h(n_k) >= 0$, entonces $h(n_k) <= h(n_k)$.

LQCD.

Un estimado es consistente si es admisible y cumple $h(m) - h(n) <= k(m,n)$ para todo par de nodos m, n; siendo $k(m,n)$ el costo del camino entre m y n (*infinito* si no hay camino). Es conveniente que el estimado sea consistente pues en este caso se minimiza el número de operaciones realizadas por el algoritmo A*, ya que no se reabren nodos **[NIL-71]**.

El estimado h es consistente.

Dem.

Hay que probar: $h(n_k) - h(n_j) \leq K(n_k, n_j)$, k *distinto de* j puesto que entre dos nodos de la misma etapa no hay camino.

$K(n_k, n_j) = X(k+1) + \ldots + X(j)$

$h(n_k) = X'(k+1) + \ldots + X'(j) + \ldots + X'(N)$

$h(n_j) = X''(j+1) + \ldots + X''(N)$

La diferencia entre X' y X'' está en la forma de calcularlos:

$$X'(j+1) = X'(j) + q(j) - s * V$$

$$X''(j+1) = X(j) + q(j) - s * V$$

Como $X'(j) < X(j)$ (porque el valor estimado es menor que el real) entonces

$X'(j+1) <= X''(j+1)$; luego

$$h(n_k) - h(n_j) <= X'(k+1) + \ldots + X'(j)$$

Además

$$X(k+1) + \ldots + X(j) < X(k+1) + \ldots + X(j) = K(n_k, n_j)$$

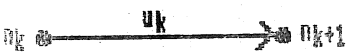
entonces

$$h(n_k) - h(n_j) <= K(n_k, n_j)$$

luego es consistente.LQQD.

Observación: El estimado h de un nodo se puede calcular fácilmente a partir del de su predecesor (lo cual es importante para la eficiencia del algoritmo):

Sean n_k y n_{k+1}



$h(n_k) = X'(k+1) + X'(k+2) + \ldots + X'(N)$

$h(n_{k+1}) = X''(k+2) + \ldots + X''(N)$

Entonces conociendo el estimado de n_k calculamos el de su sucesor de la siguiente forma:

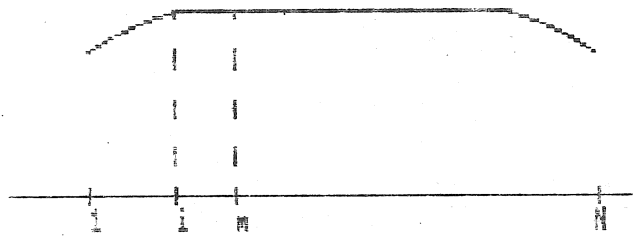
$$h(n_{k+1}) = h(n_k) - \sum_{i=1}^{2} (X_i(k) + q_i(k) - s_i * V_i) + \sum_{i=1}^{2} s_i * (V_i - u_i(k)) * (N-k-1)$$

EFICIENCIA DEL ALGORITMO

Es necesario almacenar los nodos abiertos. Si tenemos 60 etapas, en el peor caso el número de abiertos podría ser del orden de 8**60, por lo tanto hay que buscar alguna simplificación

Hipótesis sobre el comportamiento del tránsito:

En la hora saturada el comportamiento del tráfico varía al principio hasta alcanzar un equilibrio, que se pierde hacia el final de la hora saturada.



Entonces, de acuerdo al comportamiento anterior, se puede buscar el camino óptimo hasta la mitad de la hora saturada, y luego repetir la secuencia de controles óptima, en sentido inverso en la otra mitad del período. Hay dos opciones:

a) Buscar una solución óptima hasta la etapa m (m <= N/2). Repetir la secuencia de los últimos m-i controles hasta llegar a la etapa N-i. Finalmente poner los i primeros controles en orden inverso, desde la etapa N-i+1 hasta la N. (Se elige m tal que la cantidad de nodos abiertos sea manejable, según la capacidad de la máquina disponible).

b) Asignar todo el espacio de memoria disponible, y si en algún momento no es suficiente, se podan los nodos menos prometedores. Si el estimado es bueno, al eliminar los menos prometedores, la solución que se obtiene estará cercana al óptimo.

En ambos casos se está reduciendo el número de caminos a explorar, pero bajo los supuestos realizados, la solución obtenida estará muy cerca de la óptima.

RED SOBRESATURADA

En una red de n intersecciones, como hay 6 o 7 posibles valores de u para cada cruce, existen 6**n controles posibles para la red, lo que se traduce en esa cantidad de sucesores para cada nodo.

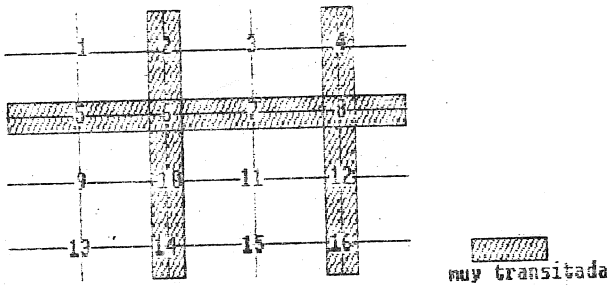
El problema en este caso también se puede representar por un grafo, pero antes de aplicar algoritmos de búsqueda de caminos es necesario reducir la cantidad de sucesores.

Una manera de lograr reducir el grafo es dividir la red en grupos de cruces, y asignarle a todos los cruces de un grupo el mismo vector de control. Hay varios criterios posibles para agrupar intersecciones, por ejemplo:

- igual número de cruces
- cruces con arribos y salidas similares
- cruces con restricciones similares (en cuanto a longitud de cola)

Como a todos los cruces del grupo se les va a asignar el mismo vector de control, el criterio de agrupación más conveniente es el segundo.

Ej:



Los grupos para esta red son

- (6, 8)
- (2, 4, 5, 7, 10, 12, 14, 16)
- (1, 3, 9, 11, 13, 15)

De 6**16 vectores posibles se redujo a 6**3.

Sabiendo que con grupos grandes se pierden muchas soluciones, podría ser conveniente subdividir los grupos grandes (que aparecen cuando hay muchos cruces con arribos y salidas similares).

Una vez realizada la división en grupos, hay distintas formas de resolver el problema:

a) Aplicar aproximaciones sucesivas **[BRO-87]** sobre los grupos: para la elección de una solución inicial hay varias opciones, una posibilidad sería dar a todos los grupos el vector de control que corresponde al valor mínimo de luz verde.

A partir de la solución inicial, se busca la secuencia de control (u (k)) óptima para un grupo, manteniendo constantes los controles de los otros grupos. (Para encontrar el óptimo de cada grupo se busca el camino mínimo en un grafo en el que cada nodo tiene 6 sucesores, debido a que se le asigna la misma secuencia de control a todos los cruces)

b) Basándonos en la Hipótesis de comportamiento del tránsito en la hora saturada, vista en la solución para un cruce simple sobresaturado se hace lo siguiente: Buscar una solución hasta la etapa m. Repetir la secuencia de m - l controles hasta llegar a la etapa N-l. Luego poner los l primeros controles en sentido inverso.

Buscar la solución hasta la etapa m, con h grupos, es buscar el camino óptimo en un grafo donde cada nodo representa el estado de los h grupos de la red, con 6**h sucesores por nodo, y m etapas.

El número de nodos abiertos en el algoritmo puede ser en el peor caso del orden de (6**h)**m. Por lo tanto hay que elegir h y m convenientes de acuerdo a la disponibilidad de memoria y tiempo de máquina, pero sin perder de vista que cuanto menor sea la cantidad de grupos, más posibilidades hay de perder el óptimo.

La solución para el caso **(b)** sería:

_ Elegir h y m.

_ Buscar el camino mínimo en el grafo, para m etapas (dándole a todos los cruces del grupo el mismo valor de u).

_ Repetir los últimos m - 1 vectores de control en las siguientes etapas (hasta la N-1)

_ Dar los controles hasta el ciclo N en sentido inverso a los encontrados para las etapas 1 a i

Se pueden combinar los dos métodos, aplicando (b) en **(a)**.

2.2.2 _ CONTROL ON-LINE

CRUCE SIMPLE SOBRESATURADO

Para resolver el problema en tiempo real se deben tratar las etapas en forma independiente, porque no se puede conocer los q_i futuros. Por esta razón se debe hallar la mejor solución para una etapa dada, optimizando el tiempo de espera en esa etapa (ciclo).

El modelo visto anteriormente describe la evolución de las colas en el cruce mediante la siguiente ecuación

$$X_i(k+1) = X_i(k) + q_i(k) - g_i(k)$$

$$\text{siendo } g_i(k) = s_i * u_i(k) .$$

El objetivo es

$$\textbf{(10)} \qquad I = \min_{u_1,u_2} \sum_{i=1}^{2} X_i(k+1)$$

Como en cada ciclo hay 6 o 7 posibles valores para G_i (ya que $u_i = G_i / C$), hallar los valores u_i que satisfacen la ecuación **(10)** consiste en buscar el mínimo para 6 o 7 valores.

Algoritmo

Para obtener el valor u_i (k) (i=1,2) óptimo, teniendo como dato el $q_i(k)$ y X(k), se calcula la sumatoria de la ecuación **(10)** para cada posible valor de u_i (k). El control óptimo es el que satisface **(10)**, y será aplicado en el ciclo k+1; durante éste se calcula el control óptimo para el k+2 y así sucesivamente.

Nota: No se puede calcular el control óptimo del ciclo k cuando se está en el ciclo k. Lo que se hace es calcular el óptimo para el ciclo k+1, utilizando el q_i (k-1) (debido a que q_i (k) recién se conoce al final del ciclo k). Este pequeño desfasaje no es significativo en el problema.

REDES SOBRESATURADAS

Al trabajar on-line, la optimización de todos los cruces se va a hacer etapa por etapa, debido a que como en el caso anterior, no se dispone de los q_i futuros. Por esta razón la función

objetivo es

(11) $$I = \min \sum_{i,j} [X_{1(i,j)}(k+1) + X_{2(i,j)}(k+1)]$$

Suponiendo una red de n cruces, la cantidad de vectores de control posibles es 6**n (6 por cada cruce). Luego, si n es grande, resulta ineficiente aplicar el algoritmo dado para un cruce simple on-line.

Por lo tanto la idea es disminuir la cantidad de vectores de control haciendo un análisis por grupo, y utilizar el método de aproximaciones sucesivas, asignando a todos los cruces del grupo el mismo vector de control.

Los criterios para dividir en grupos son los mismos que se dieron en la solución para red sobresaturada con control off-line.

Para aplicar aproximaciones sucesivas se utiliza como solución inicial, por ejemplo, los valores medios de las variables de control.

Algoritmo

Partiendo de la solución inicial, trabajar sobre cada grupo tomando de los vectores de control que cumplan las restricciones de ese grupo aquel que satisface (11). (Se mantienen constantes los otros grupos). Para esto hay que calcular la matriz de estado del ciclo k+1, (X (k+1)), para cada uno de los vectores de control, y luego sumar los elementos de la matriz.

Para calcular la matriz se necesita

q_{ij} (k) entradas en todos los cruces de la red

q_{ij} (k-1) salidas en todos los cruces de la red que es igual a $s * u_{ij}$ (k-1)

Se necesitan los últimos r vectores de control de cada cruce, siendo r la demora más grande en la red.

El algoritmo puede terminar por dos razones:

- porque se terminó el tiempo disponible para hallar la solución (no olvidemos que estamos trabajando en tiempo real)

- porque se llegó a un equilibrio (no hay diferencia entre dos iteraciones). Si todavía hay tiempo se puede mejorar la solución con grupos más pequeños.

Nota: Al buscar una solución por grupos se corre el riesgo de perder el óptimo, pero con aproximaciones sucesivas se obtiene rápidamente una solución aceptable.

Se utiliza el método de aproximaciones sucesivas porque es un problema en tiempo real y por lo tanto se necesita tener rápidamente una buena solución para entregar, y si hay tiempo se puede seguir mejorándola.

3 _ PERIODO NO SATURADO

3.1 _ MODELO

INTERSECCION SIMPLE NO SATURADA

Se entiende por intersección no saturada aquella en la cual al final de la luz verde no queda cola.

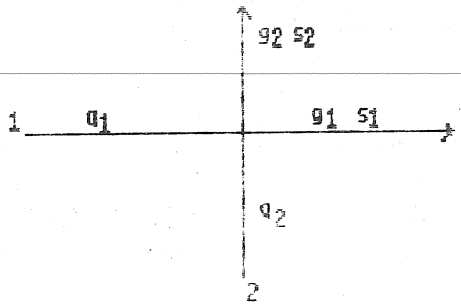
El comportamiento del tráfico en una intersección no saturada es bien conocido: cuando en una de las calles se prende la luz roja, se empiezan a formar colas. Cuando cambia a verde, las colas comienzan a disminuir hasta que desaparecen, y luego empiezan a fluir libremente los vehículos que van llegando a la intersección, hasta que se termina la luz verde, momento en que nuevamente comienzan a formarse colas.

Es el tiempo pasado en cola durante el ciclo lo que causa las demoras entonces será necesario medir en distintos momentos durante el ciclo, la longitud de las colas (por ejemplo medirlas cada segundo dentro del ciclo).

En este caso no es válido ver las llegadas y salidas como un promedio por ciclo, es necesario ver las variaciones segundo a segundo dentro del ciclo.

Debido a que en todos los ciclos al terminar la luz verde no hay cola, los ciclos se pueden analizar independientemente, por lo cual el problema que se plantea es la optimización de las longitudes de cola durante un ciclo.

Se utiliza el siguiente modelo [SIN-73] para una intersección simple no saturada:



Sean q_1, q_2 flujo de entrada en la dirección 1 y 2
g_1, g_2 flujo de salida en la dirección 1 y 2
X_1, X_2 longitud de la cola en la dirección 1 y 2
s_1, s_2 flujo de saturación en la dirección 1 y 2
} en cada segundo

Llamamos "períodos" a los intervalos que resultan de subdividir el ciclo en segundos.

La evolución de las colas este regida por la siguiente ecuación:

(12) $$X_i(k+1) = X_i(k) + q_i(k) - g_i(k) \qquad i=1,2$$

donde k indica el período dentro del ciclo.

Sea N_i el número de períodos durante la luz verde en la dirección i, o sea durante G_i. La salida se define como:

$$(13)\qquad \begin{cases} g_i(k) = s_i(k) & \text{si } X_i(k) > 0 \text{ y } k < N_i \\ g_i(k) = q_i(k-d) & \text{si } X_i(k) = 0 \text{ y } k < N_i \\ g_i(k) = 0 & \text{en otro caso} \end{cases} \qquad i = 1,2$$

donde $s_i(k)$ es la cantidad de vehículos que pasan en un período (flujo de saturación) en la dirección i

d es el retardo en el cruce

En otras palabras:
la salida es _ 0 si hay luz roja ($k > N_i$)

_ $s_i(k)$ si hay cola y luz verde

_ $q_i(k-d)$ si hay luz verde y no hay cola (salen a medida que llegan)

OPTIMIZACION PARA UN CRUCE SIMPLE NO SATURADO

Igual que en el caso sobresaturado, el costo esta determinado por la suma de las longitudes de las colas en el cruce. Pero en este caso se consideran solo las que se producen durante un ciclo.

La función objetivo es:

$$(14)\qquad \min_{u_i} \sum_{i=1}^{2} \sum_{k=2}^{N} X_i(k)$$

donde N es la cantidad de períodos del ciclo.

Minimizar tiempos de espera equivale a minimizar longitudes de cola, o sea que el problema es encontrar el valor de u_i (tiempo de luz verde durante el ciclo, en la dirección i), que minimice la suma de las longitudes de colas.

3.2 _ ESTRATEGIAS DE CONTROL ON-LINE

Por ser los ciclos independientes (como se explicó en el modelo), la secuencia de controles óptima esta formada por el óptimo en cada período. Por lo tanto sólo se hará el análisis de las soluciones en tiempo real puesto que no es necesario ver los datos de todos los períodos conjuntamente.

CRUCE SIMPLE NO SATURADO

El objetivo es:

$$\min_{u_i} \sum_{i=1}^{2} \sum_{k=1}^{N} X_i(k)$$

donde u_i es la fracción de luz verde en la dirección i.

Para calcular $S = \sum_{i=1}^{2} \sum_{k=1}^{N} X_i(k)$ se necesita conocer $q_i(k)$, ya que el flujo máximo por

segundo (s_j) es un valor conocido, y g_j (k) se calcula con la ecuación booleana (13). Como se está trabajando en tiempo real, el valor de q_j (k) se obtendrá a través de "sensores".

Hay dos formas de medir q_j (k):

a) antes de cada luz verde calcular el cociente (longitud de cola / tiempo de luz roja) y tomar q_j (k) constante para todo k del ciclo, igual al cociente anterior.

b) mediante sensores, medir en cada período la tasa de arribo de vehículos. De esta forma obtenemos un valor de q_j (k) para cada k.

Algoritmo

Para obtener el valor de u_j óptimo, calculamos S para cada valor posible de u_j (como se vió en el análisis que se realizó para el caso sobresaturado, hay 6 o 7 valores posibles), y tomando el u_j que minimice S. Estando en un ciclo j, con la sucesión q_j(k) medida en el ciclo anterior (j-1), calculamos el u_j que minimice S. Ese valor será el que se usará en el próximo ciclo (j+1).

Así, en cada ciclo se mide la sucesión q_j (k) que se utilizará en el siguiente, para obtener el u_j óptimo para el ciclo posterior. (Se asume que el tiempo de cálculo del u_j óptimo es inferior al tiempo del ciclo).

Nota: El mismo algoritmo se puede utilizar en el caso de no disponer de sensores,trabajando con datos históricos.

RED NO SATURADA

En el caso de red no saturada, es importante que los ciclos de los semáforos estén sincronizados de tal forma que cuando los vehículos que salen de un cruce llegan al siguiente, encuentren luz verde.

Debido a que el período de optimización en el caso no saturado es un ciclo, y estamos trabajando en tiempo real, una vez que están sincronizados los cruces, se puede tratar cada uno independientemente aplicando lo visto para un cruce simple. Entónces, en la red no saturada el problema se resuelve en dos etapas

1) Sincronización de los cruces

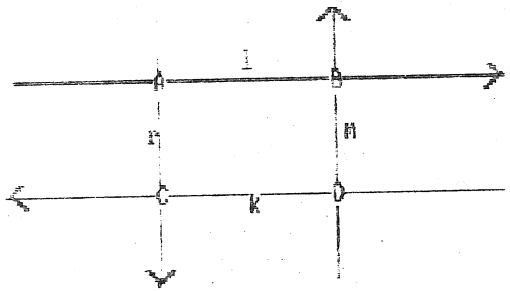
2) Optimización en cada cruce

Para el segundo punto, se puede aplicar el algoritmo de optimización dado para cruces simples.

Se analizará ahora el primer punto, o sea el problema de la sincronización de los cruces.

En calles de una única mano, la sincronización se logra desfasando los ciclos de los semáforos el tiempo suficiente para que los vehículos vayan de un cruce a otro a velocidad normal. Como el tiempo de luz verde es distinto en cada ciclo y en cada cruce, y además la distancia entre cruces no es la misma en toda la red, no es posible sincronizar todos los cruces.

Por ejemplo:

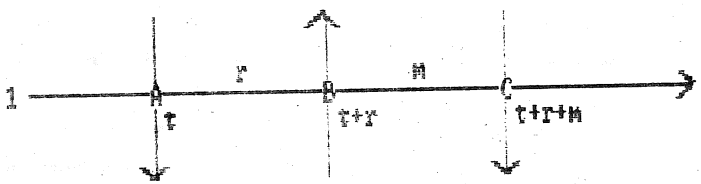


l = tiempo necesario para recorrer la distancia entre los cruces A y B a velocidad normal.

Análogamente se definen r, m y k.

Si l = k y m = r no será posible sincronizar estos cuatro cruces, y aún en el caso en que todos los cruces estén a la misma distancia, solo es posible sincronizarlos si el tiempo de luz verde por ciclo es constante para todos los cruces (no se cumple esto en nuestro caso). Luego, el objetivo no es sincronizar toda la red sino solo algunas calles, aquellas en las cuales la fluidez del tráfico sea importante.

En una red urbana generalmente se pueden distinguir calles muy transitadas y calles de poco tráfico. Nuestro problema consiste en sincronizar las primeras.



Para sincronizar la calle 1: Si el ciclo en A empieza en el instante t con luz verde en dirección 1, y el tiempo de demora en recorrer la distancia entre A y B es r, en el instante t+r debe haber luz verde en dirección 1 en el cruce B, o sea que los ciclos en B están desfasados r con respecto a los de A, y los de C están desfasados r+m, y así sucesivamente.

Un análisis más completo de las posibilidades de sincronización tendría que hacerse sobre la red particular que se esté analizando.

4 _ GENERALIZACION

4.1_ CRUCES DE AVENIDAS SOBRESATURADAS

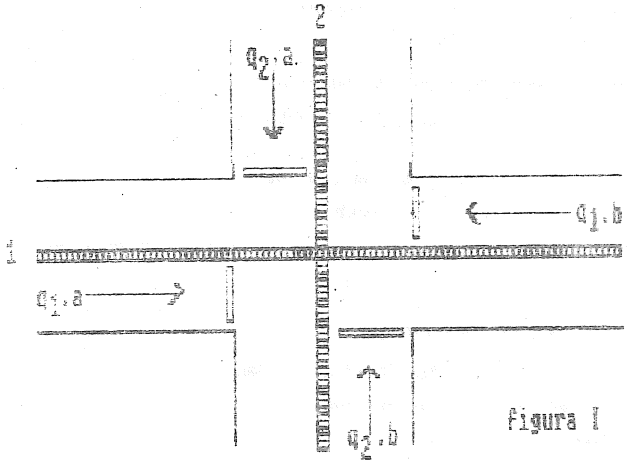


Figura 1

La generalización del modelo de cruce simple a un cruce de avenidas es sencillo. Consideraremos la intersección de dos avenidas de un único carril por mano (ver fig 1). Para cada avenida se define

u_i, g_i, C análogamente al modelo para cruce sobresaturado.

q_i.a, q_i.b . tasa de arribo de vehículos en cada sentido de la avenida i.

s_i flujo de saturación considerando una sola mano de la avenida i.

En cada mano de la avenida i se forman colas, cuya evolución dinámica se describe por ecuaciones similares a las del modelo para cruce sobresaturado:

$$X_i.a(k+1) = X_i.a(k) + q_i.a(k) - g_i.a(k)$$
$$X_i.b(k+1) = X_i.b(k) + q_i.b(k) - g_i.b(k) \qquad i=1,2$$

En una avenida pueden suceder dos cosas:

1) Las dos manos de la avenida son sobresaturadas.

2) Una de las manos es sobresaturada y la otra no.

En el caso (2) para representar el estado del cruce tomamos sólo la mano sobresaturada, y vemos a la avenida como una calle común.(Se aplican las mismas estrategias que en los casos estudiados antes para intersecciones de calles simples sobresaturadas).

En el caso (1), como en un cruce de avenidas el tiempo de luz verde para cada sentido de la avenida es el mismo

$$u_i.a(k) = u_i.b(k) \quad \text{para todo } k$$

y como $g_i = s_i * u_i$

entonces $g_i.a(k) = g_i.b(k)$ para todo k.

Representaremos el estado de las colas en la avenida por

$$X_i(k) = \max [X_i.a(k), X_i.b(k)]$$

con lo cual se podrá utilizar el modelo de cruces de calles sobresaturadas, con la única salvedad de que en el vector de estados tendremos (X1.a(k), X1.b(k), X2.a(k), X2.b(k)), pero para trabajar igual que en los casos anteriores se toma

$$(\max_{i=a,b} (X_1.i(k)), \max_{i=a,b} (X_2.i(k)))$$

Al generalizar de esta forma el modelo, los métodos a aplicar para encontrar la secuencia óptima de controles son iguales a los vistos anteriormente.

4.2_ CRUCES DE CALLES CON VARIOS CARRILES

En los modelos anteriores todas las variables están definidas sobre la hipótesis de que las calles son de una única vía. Veremos como extenderlas para el caso de calles con más de un carril. Supongamos calles de n carriles.

- Sea $q_i'(k)$ el número total de vehículos que arriban en un ciclo al cruce por los n carriles, en la dirección i. Bajo la hipótesis de que los vehículos que llegan al cruce se distribuyen homogéneamente en los n carriles, podemos decir que $q_i(k) = q_i'(k) / n$
- Si s_i' es la máxima cantidad de autos que pueden salir en la dirección i en un ciclo, definimos $s_i = s_i' / n$
- u_i es idéntica a la dada anteriormente en los modelos, o sea la fracción de luz verde en la dirección i.
- $g_i = u_i * s_i$

Luego, con esta nueva definición de las variables, a los cruces de calles con muchos carriles se les pueden aplicar los modelos dados anteriormente, y utilizar las soluciones ya vistas.

5_ CONCLUSIONES

A lo largo de este trabajo se abordaron algunos de los problemas que encierra el control de semáforos en una red de tráfico urbano. Se aplicaron conceptos y herramientas teóricas a problemas concretos obteniendo soluciones factibles, sin llegar a una implementación concreta.

En el trabajo se analizan sistemas de control de señales de tráfico off-line y on-line. Las soluciones presentadas para el primer caso son actualmente aplicables, en cambio no es común en la actualidad el uso de sistemas de control on-line (suponemos que se debe al costo, puesto que existe la tecnología adecuada para implementar tales sistemas).

Nota: Una versión anterior de este trabajo constituyó el proyecto final de la materia Algoritmos y Estructuras de Datos II [BRO-87], cursada por los autores en la Escuela Superior Latinoamericana de Informática en el año 1987.

6_ AGRADECIMIENTOS

A Edgardo Broner y Juan V. Echague por las valiosas ideas aportadas en la realización de la primera versión de este trabajo

7_ BIBLIOGRAFIA

[BRO-87] Apuntes del curso de Algoritmos y Estructuras de Datos II
E. Broner, E.S.L.A.I., 1987

[SIN-73] Hierarchical Strategies for the On-Line Control of Urban Road Traffic Signals
M. G. Singh, Lecture Notes in Computer Science, Vol 4, 1973

[NIL-71] Problem Solving Methods in Artificial Intelligence
N. Nilsson, Mc Graw-Hill, 1971

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

Nuevas Técnicas en el Análisis de Series de Tiempo Discretas: Aplicación en Modelos Económicos

Oscar Liberman
Fernando Tohmé
Departamento de Economía
Universidad Nacional del Sur
Bahía Blanca

Introducción

Los motivos que llevaron a la realización de este trabajo, pueden ser clasificados en motivos operativos y en motivos históricos. De estas dos categorias surge entonces un replanteo de la actual teoria de series de tiempo.

Desde el punto de vista de los motivos operativos, puede verse hoy en dia que la ciencia, y la economia en especial, no ha obtenido resultados satisfactoriamente operativos a través de las series de tiempo. Las situaciones por las cuales las series de tiempo discretas no manifestaron operatividad son principalmente tres:Requerimiento de excesiva cantidad de observaciones, Amplio margen de error en el corto y mediano plazo, Falta de relación con la causalidad de los fenómenos. Con respecto a los motivos históricos, ha existido a lo largo de la historia de la investigacion de las series de tiempo una fuerte tendencia a seguir las corrientes de lo investigado por la teoria del sonido. Claros ejemplos de esto son la existencia de términos como ruido blanco o filtros. Actualmente en aplicaciones de la teoria del sonido, los especialistas se han apartado en forma sustancial de sus métodos anteriores, lo que nos lleva a pensar si no puede ser necesario ,por analogia, replantear la forma de trabajar con las series de tiempo.

Estas dos justificaciones son tratadas con mayor detalle en la primer parte de este trabajo. La segunda parte del mismo contiene el desarrollo teórico del modelo y las posibles aplicaciones prácticas tanto en el análisis explicativo, como en lo que respecta a causalidad y predicción. La tercer parte del trabajo consta de un ejemplo práctico de como utilizar estos nuevos conceptos y de la forma en que aparecen las limitaciones del mismo.

PARTE I: "Por què un nuevo mètodo de Series de Tiempo ?"

Motivo Operativos: Tres problemas se manifestaron siempre en el trabajo con series de tiempo discretas :

1 - Requerimiento de gran cantidad de observaciones.
2 - Amplio margen de error en el mediano y corto plazo.
3 - Falta de relaciòn con la causalidad de los fenòmenos.

1 - Requerimiento de gran cantidad de observaciones:

La teoria de series de tiempo vigente tiene una gran restricciòn, presente tanto en los mètodos robustos como en los no robustos. [1] Esta limitacion reside en la necesidad de conseguir alrededor de 150 observaciones para poder realizar una aproximaciòn satisfactoria. Este hecho es de por si limitante, ya que es sòlo podremos conseguir, en general, datos de corto y mediano plazo (frecuencias diarias y semanales; eventualmente anuales con mayores restricciones de informaciòn). Como se verà en el pròximo punto, no es precisamente en este tipo de plazos donde los mètodos de series de tiempo evidencian mayor grado de certeza.

En este trabajo se mostrarà que para solucionar esta situaciòn se trabaja sobre la base de modelos ajustables dinàmicamente, donde la cantidad de observaciones no es una variable de trascendencia fundamental. Se verà en el ejemplo que los mòdulos de ajuste del modelo dinàmico utilizado constan entre 20 y 40 observaciones habièndose registrado experiencias con una cantidad inferior, obtenièndose muy buenos resultados explicativos.

---

[1] Por mètodos robustos se entiende aquellos que filtran los picos de las funciones, mientras que no robustos son aquellos mètodos que estiman una tendencia considerando todos sus puntos.

2 - Amplio margen de error en el corto y mediano plazo:

Los resultados tradicionalmente obtenidos mediante los mètodos de series de tiempo son en general sólo tendencias. Esto implica un margen de error en el corto y mediano plazo. Conviene recordar que esto se debe a la intenciòn de suavizar los picos de la funciòn, ya sea mediante la absorciòn de los saltos o la separaciòn de la componente aleatoria de la no aleatoria. Ademàs desde el punto de vista explicativo se han dado siempre en las aproximaciones clàsicas problemas que van desde la falta de significatividad hasta la aparición de multicolinealidad debido a la presencia de variables aleatorias.

Una de las corrientes màs recientes propone realizar un preblanqueo de las variables antes de ajustarla por minimos cuadrados. [2]

De esta forma se eliminaria la componente aleatoria de cada variable, pero surge el problema del amplio margen de error caracteristico de los mètodos estimadores de tendencias. En este trabajo se utiliza el mètodo de incorporar ecuaciones que expliquen "por si mismas" a las variables con fuerte presencia de componente aleatorio, pero sin necesidad de preblanquear la serie. [3]

---

[2] La idea del preblanqueo consiste en agregar al modelo ecuaciones de series de tiempo que separen la componente aleatoria de una variable convirtiendo a èsta la expresiòn de su tendencia. De esta forma se evitan posibles problemas en los resultados que pueden no haberse previsto en el modelo, como asi mismo reveer modelos que presentaron en el pasado problemas de ajuste. [Granger & Newbold- bib. 4]

[3] Con la frase "explicaciòn por si misma" de la variable quiere significarse que sin dejar de lado la incidencia que esta puede tener sobre otras variables, se reconoce la existencia de una componente aleatoria tan fuerte a travès del tiempo que produce perturbaciones sobre el ajuste final. Esto impide considerar otra causa del comportamiento de la variable que su propia conducta. Esta situaciòn se da frecuentemente en los modelos de retroalimantaciòn.

3 - Falta de relación causal entre los fenómenos:

Si nos remontamos a los orígenes del estudio de las series de tiempo, es posible ver que la justificación de utilizar sus métodos no radica precisamente en la posibilidad de dar cuenta de la causalidad de los fenómenos, sino por el contrario en la hipótesis de la ausencia de relaciones causales en los mismos. Con esto se quiere resaltar el hecho de que es tan grande la componente aleatoria de la variable, que si estudiamos su evolución a través del tiempo, se verá una presencia frecuente de ruido blanco. Así es que se sugiere entonces encontrar una estimación de la tendencia de la variable en el tiempo sin tener en cuenta sus relaciones causales.

Si bien esto no es falso ni erróneo, también conviene recordar que esta solución parece la única recomendable, ya que los métodos de ajuste conocidos y utilizados que detectan relaciones de causalidad entre fenómenos verifican problemas ante la presencia de aleatoriedad y en última instancia terminan realizando transformaciones de variables bastante similares a las estimaciones de tendencia. [4]

Lo que ahora se propone esconseguir aproximaciones más rigurosas y realistas a las series de tiempo. Si bien no se puede exigir estricta rigurosidad en modelos que, para estimar tendencias, filtran o absorben las observaciones que se alejan demasiado de la variación media, si es importante evitar la pérdida de significatividad de posibles observaciones. Aplicando los métodos tradicionales de series de tiempo se han evitado o absorbido saltos que no siempre eran de origen aleatorio sino que seguramente estaban inducidos por otra variable que, al presentar los mismos problemas de aleatoriedad , no permitían verificar una correlación entre ambas.

[4] Pulido San Román - [bib. 1]

Se puede decir que la componente aleatoria de una variable es fuerte si estudiando la evolución de la variable en el tiempo, se detectan oscilaciones bruscas que se aproximan al ruido blanco. Cuando se analizan variables de origen natural, como temperaturas a una misma hora todos los dias, es bastante lógico que en los comportamientos diarios se observen componentes aleatorias, y que los cambios de estación sólo eleven o disminuyan el nivel general de la tendencia. Pero si las variables obedecen a causas determinadas por factores humanos, como por ejemplo la cotización de un activo financiero, la situación es completamente diferente. En el enfoque tradicional se parte de aceptar que el comportamiento de corto plazo es aleatorio y que hay una tendencia distinguible y en cierto modo anticipable. Pero conviene estudiar el proceso más cuidadosamente en cada uno de los instantes y asi recorrer la oscilación de una variable en el tiempo desde el origen de la medición. Al llegar al primer salto importante se infiere que existió alguna causa que determinó ese cambio. Se ve que cada uno de éstos casos será absorbido como aleatorio por los métodos tradicionales quedando luego una tendencia que indique la participación ponderada de éstos saltos. De aqui podemos ver claramente que si se quiere obtener relaciones de causalidad entre variables, éstos métodos no representan la situación real sino que sólo dan una idea global del comportamiento de las mismas.

Como ejemplo conviene observar dos series en las que sabemos a priori que puede existir relación causal, y tomar dos observaciones consecutivas y desfasadas en cada una de las series. Es probable que a simple vista se pueda distinguir como una variación en una de las variables es seguida por una variación en la otra. Esta situación probablemente pase desapercibida en un análisis clásico.

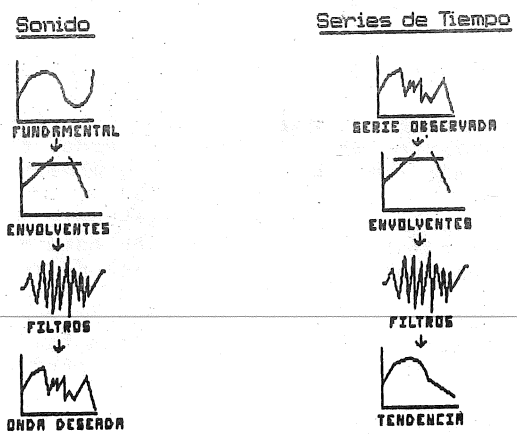
Motivos Històricos:

La circunstancia que llevò a plantear este trabajo, tiene que ver en cierto modo con la existencia de un paralelismo històrico entre la ingenieria de sonido y el anàlisis de series temporales. En un primer momento los diseñadores de los diferentes mètodos de series, buscaron sus fuentes en el anàlisis de ondas sonoras. Màs especificamente, existiò un paralelismo entre la programaciòn analògica de sintetizadores de sonido y la obtenciòn de tendencias en series de tiempo. Para obtener sonidos mediante un sintetizador analògico, el proceso consiste en generar una onda bàsica con un oscilador, y mediante el uso de filtros y envolventes conseguir onda final deseada. A la inversa los analistas de series de tiempo, parten de una onda final obtenida a travès de observaciones y mediante el uso de filtros y envolventes se llega a separar la onda fundamental de la onda de ruido. Hasta este punto, ambos mètodos llevaron caminos paralelos, pero los desarrollos posteriores en sintesis digital marcan un techo bastante rigido en el horizonte de la sintesis sonora. Hay que tener en cuenta que toda esta evoluciòn se debiò principalmente a la imposibilidad de los ingenieros de sonido de tomar la muestra de la onda y reproducirla. En cambio los analistas de series no mostraron interès en estimar la forma original en un sentido riguroso, sino que tratan de filtrar esa onda hasta obtener la fundamental.

Si comparamos las metodologias tradicionales en ambos campos, existe un punto de contacto : los dos enfoques se enfrentan a la onda, pero evitan represntarla en forma rigurosa. La diferencia reside en que los ingenieros de sonido tomaron èsta decisiòn por falta de medios mientras que los analistas de series lo hicieron por considerar necesario separar la componente aleatoria.

Todo este proceso histórico, se encuentra ahora en un periodo de transición debido a la aparición de los "SAMPLERS", que son generadores de sonido, que como su nombre lo indica (sampler significa en inglés muestreador), toman una muestra de un determinado sonido y tratan de imitarlo rigurosamente . Un sampler trabaja efectuando una transformada rápida de Fourier. Cabe preguntarse si en el trabajo de series de tiempo no ha llegado la hora de tomar también este camino de la reproducción rigurosa.

La siguiente figura esquematiza este paralelismo historico entre ambos procesos.



PARTE II - "Desarrollo del Modelo"

El modelo planteado está dirigido a obtener dos tipos de estimaciones: explicativas y predictivas.

En el primer caso, el proceso puede sintetizarse de la siguiente manera:

1 - Ingresar observaciones de una variable para un determinado periodo de tiempo .

2 - Estimar la función muestreada a través de la transformada de Fourier.

3 - Introducir en el modelo de ajuste la nueva función obtenida, evitando asi los tradicionales problemas producidos en los ajustes por las variables aleatorias y poder crear asi un modelo explicativo que incluye a estas variables.

La segunda aplicación del modelo requiere de los siguientes pasos:

1 - Estimación de posibles funciones causales, desfasadas tantos periodos como retardos existan, pero utilizando idéntica cantidad de observaciones.

2 - Determinar la relación existente entre las funciones estimadas, quedando de esta manera la variable tiempo absorbida como núcleo de relación.

3 - Ingresar una nueva observación para la función que anticipa, obteniendo asi una tercera función similar a la primera pero desfasada un periodo.

4 - Mediante la función de relación y la nueva serie estimada, inferir un nuevo valor par la segunda función.

Aplicación explicativa

Estimación de la serie observada: en ingenieria de sonido, las diferentes fábricantes de samplers utilizaron algoritmos que de una otra manera contienen la transformada rápida de Fourier. Las diferencias algoritmicas dependen de la capacidad de proceso del sampler y del tipo de sonido que se quiere realzar. En este modelo se utiliza la transformada de Fourier directamente. La fórmula que estima

la serie es:

$$F(t) = \frac{1}{2} a_0 + \sum_{j=1}^{k} a_j \cdot \cos\left(\frac{2.\pi.j.t}{T+1}\right) + b_j \cdot \operatorname{sen}\left(\frac{2.\pi.j.t}{T+1}\right) + \frac{\phi}{2} \cdot \cos\left(\frac{(k+1)2.\pi..t}{T+1}\right)$$

donde:

T : cantidad de observaciones en el tiempo.

K : es T/2 si T es par.
(T-1)/2 si T es impar.

ϕ : 0 si T es par.
1 si T es impar.

y los vectores aJ y bJ se obtienen de la siguiente forma:

$$aJ = \frac{2}{T+1} \sum_{i=0}^{T} X_i \cdot \cos\left(\frac{2.\pi.j.i}{T+1}\right)$$

$$bJ = \frac{2}{T+1} \sum_{i=0}^{T} X_i \cdot \sin\left(\frac{2.\pi.j.i}{T+1}\right)$$

Las observaciones abarcan el periodo 1 a T, con iteración 1. En la función F(t) se encuentran todos los puntos de la variable comprendidos entre 1 y T. Hipotéticamente si las observaciones son diarias, medidas a las 0.00 Hs. de cada dia, podemos inferir la observación al mediodia del tercer dia estimando F(3.5). Si bien de esta manera se puede explicar la variable para un periodo en forma continua, nos queda por resolver el problema de la periodicidad, ya que si estimamos F(T+1), el valor obtenido coincidirá con F(1), y asi sucesivamente lo que seguramente no representa una situación real.

Esta situación se resolverá en la segunda aplicación del modelo, o sea en el desarrollo de predicción De cualquier forma , se puede ver que si ahora introducimos esta función obtenida en cualquier método clásico de ajuste, se resolverán los problemas de aleatoriedad confiriéndole al modelo un grado más de multiecuacionalidad. Luego sólo resta aplicar algún método de resolución adecuado para modelos multiecuacionales.

Como se verá nuevamente en la aplicación siguiente, cada vez que ingresamos una nueva observación, eliminamos la primera. Esto significa que siempre tenemos T observaciones. Obteniendo la función de transformación entre cada una de éstas estimaciones, podemos derivar la tendencia de la serie.

Aplicación del modelo al análisis causal y a la predicción

Como se vió anteriormente, el principal problema que presentan las estimaciones obtenidas por el modelo es el de la periodicidad. Este supuesto debe ser solucionado debido a que suponer periodicidad en series de tiempo en general puede resultar una hipótesis alejada de la realidad.

Para corregir esta situación, conviene recordar el propósito inicial del trabajo y establecer un sistema de ajuste por etapas. El método a seguir consta de las siguientes etapas :

a) Estimación de dos funciones correspondientes a dos variables que observen causalidad, y de una tercera que relacione a ambas.

b) Ingresar un nuevo dato para la función "causa" y reestimarla.

c) Inferir el nuevo valor de la función "efecto", utilizando la relación entre ambas funciones.

Con respecto a la estimación, se utiliza el mismo método ya visto para la aplicación explicativa. En cambio en la segunda etapa, de reestimación se realiza eliminando siempre la primera observación de la serie y agregándole un nuevo valor. De esta forma se trabaja en módulos de T observaciones, con T-1 grados de superposición. Es visible que entre dos funciones consecutivas de una misma variable, no existirá gran diferencia. Por otra parte las diferencias se presentarán en los vectores aj y bj. , ya que en última instancia la forma de la función utilizada es la misma.

Precisamente aqui es donde se soluciona el obstáculo de la periodicidad, ya que cada nueva función absorbe el dato en T+1 que de otro modo significaria el comienzo de un nuevo ciclo.

Con respecto a la continuidad de la serie, de donde se inferiria la tendencia, si se relacionan todas las sucesivas estimaciones de una misma variable, podrá derivarse la tendencia.

Analiticamente, el proceso es el siguiente:

Supongamos dos variables X1 y X2 , tal que existe o al menos creemos que existe una ley causal tal que X1 anticipa en relación directa a X2 un periodo. De otro modo se pude decir que si se observa un aumento en X1 del periodo 1 al 2, cabe esperar que X2 también aumente en forma equiproporcional del periodo 2 al 3.

De esta forma tomamos en el instante t+1 dos muestras de t observaciones periódicas de X1 y X2. Los datos de la variable X1 van desde 1 a t, mientras que los de X2 de 2 a t+1. Matricialmente podemos representar a los datos de X1 y X2 como:

[X1]=[X11...X1t] [X2]=[X22...X2t+1]

Utilizando el método que denominamos "sampler", estimamos dos funciones F(t) y G(t), correspondiendo F(t) al vector [X1] y G(t) a [X2]. Obsérvese que la explicatividad de F(t) abarca el periodo 1..t, mientras que la de G(t) el periodo 2..t+1.

Ambas funciones son idénticas, variando sólo en el valor de los correspondientes vectores aj y bj. Definimos entonces como aj0 y bj0 a vectores correspondientes a F(t) y aj1 y bj1 a los vectores de la función G(t). En forma matricial tenemos ahora las matrices [aj0,bj0] y [aj1,bj1] de k+2 filas por dos columnas. [5]

Si relacionamos F(t) y G(t), podemos obtener una función H(t), que actúe como función de transferencia intertemporal entre X1 y X2.

Luego: H(t)=G(t)/F(t)

---

[5] Los vectores aj y bj abarcan de j=1 hasta j=k, pero como en general también se deben utilizar aj para j=0 y j=k+1, entonces la matriz [aj,bj] tiene k+2 filas, siendo [a0,b0] y [ak+1,bk+1] nulos en bj.

Al ingresar el dato de X1 para el periodo T+1, eliminamos la observaciòn X1 en el periodo 1 y estimamos una nueva funciòn F*(t) de T observaciones que abarcan del periodo 2 al T+1.

Podemos estimar ahora G*(t) abrcando el periodo 3..t+2 e inducir asi el valor esperado de X2.

La funciòn G*(t) se estimarà asi:

G*(t)=G(t)/F(t).F*(t)

Si reemplazamos las funciones anàliticas aqui vistas en la transmada de Fourier, las funciones estimadas para [X1] y [X2], F(t) y G(t) resultan:

$$F(t)=\frac{1}{2}a_{oo}+\sum_{j=1}^{k}a_{j_0}\cos\left(\frac{2.\pi.j.t}{T+1}\right)+b_{j_0}.\operatorname{sen}\left(\frac{2.\pi.j.t}{T+1}\right)+\frac{\phi}{2}.\cos\left(\frac{(k+1)2.\pi..t}{T+1}\right) \qquad \text{con } t: 1\ldots T$$

$$G(t)=\frac{1}{2}a_{o_1}+\sum_{j=1}^{k}a_{j_1}.\cos\left(\frac{2.\pi.j.t}{T+1}\right)+b_{j_1}.\operatorname{sen}\left(\frac{2.\pi.j.t}{T+1}\right)+\frac{\phi}{2}.\cos\left(\frac{(k+1)2.\pi..t}{T+1}\right) \qquad \text{con } t: 2\ldots T+1$$

Y los respectivos vectores aj, bj para cada función son :

$$a_{J_0}=\frac{2}{T+1}\sum_{i=0}^{t}X_{1i}.\cos\left(\frac{2.\pi.j.i}{T+1}\right) \qquad a_{J_1}=\frac{2}{T+1}\sum_{i=0}^{t}X_{2i}.\cos\left(\frac{2.\pi.j.i}{T+1}\right)$$

$$b_{J_0}=\frac{2}{T+1}\sum_{i=0}^{t}X_{1i}.\operatorname{sin}\left(\frac{2.\pi.j.i}{T+1}\right) \qquad b_{J_1}=\frac{2}{T+1}\sum_{i=0}^{t}X_{2i}.\operatorname{sin}\left(\frac{2.\pi.j.i}{T+1}\right)$$

para F(t) para G(t)

Al ingresar el nuevo valor de X1 , es decir para el periodo t+1, obtenemos $F^{*}(t)$:

$$F^{*}(t)=\frac{1}{2}a_{o_2}+\sum_{j=1}^{k}a_{j_2}.\cos\left(\frac{2.\pi.j.t}{T+1}\right)+b_{j_2}.\operatorname{sen}\left(\frac{2.\pi.j.t}{T+1}\right)+\frac{\phi}{2}.\cos\left(\frac{(k+1)2.\pi..t}{T+1}\right) \qquad \text{con } t: 2\ldots T+1$$

Finalmente la función de transferencia intertemporal H(t) resulta:

$$H(t)=\frac{G(t)}{F(t)} \qquad \text{es decir:}$$

$$H(t)=\frac{\frac{1}{2}a_{o_1}+\sum_{j=1}^{k}a_{j}.\cos\left(\frac{2.\pi.j.t}{T+1}\right)+b_{j}.\operatorname{sen}\left(\frac{2.\pi.j.t}{T+1}\right)+\frac{\phi}{2}.\cos\left(\frac{(k+1)2.\pi..t}{T+1}\right)}{\frac{1}{2}a_{oo}+\sum_{j=1}^{k}a_{j_0}.\cos\left(\frac{2.\pi.j.t}{T+1}\right)+b_{j_0}.\operatorname{sen}\left(\frac{2.\pi.j.t}{T+1}\right)+\frac{\phi}{2}.\cos\left(\frac{(k+1)2.\pi..t}{T+1}\right)}$$

$$G^{*}(t)=\frac{\frac{1}{2}a_{01}+\sum_{j=1}^{k}a_j.\cos(\frac{2.\pi.j.t}{T+1})+b_j.\operatorname{sen}(\frac{2.\pi.j.t}{T+1})+\frac{\phi}{2}.\cos(\frac{(k+1)2.\pi..t}{T+1})}{\frac{1}{2}a_{00}+\sum_{j=1}^{k}a_{j_0}\cos(\frac{2.\pi.j.t}{T+1})+b_{j_0}.\operatorname{sen}(\frac{2.\pi.j.t}{T+1})+\frac{\phi}{2}.\cos(\frac{(k+1)2.\pi..t}{T+1})}\cdot\ldots$$

$$\ldots\cdot\frac{\frac{1}{2}a_{02}+\sum_{j=1}^{k}a_{j_2}\cos(\frac{2.\pi.j.t}{T+1})+b_{j_2}.\operatorname{sen}(\frac{2.\pi.j.t}{T+1})+\frac{\phi}{2}.\cos(\frac{(k+1)2.\pi..t}{T+1})}{\ldots}$$

De esta manera, los valores obtenidos para G*(t) pueden contrastarse en el intervalo 3..t+1, y asi obtener un desvio entre la función G*(t) y G(t) con t:3...t+1.

El valor del desvio puede ser de utilidad para verificar causalidad. Debe destacarse que la relación intertemporal H(t), representa t-1 periodos y solo permite inferir uno. Es decir que es razonable ver la evolución del desvio entre G*(t) y G(t) y si éste se torna muy grande, podemos inferir que la causalidad supuesta no verifica, porque la estimación ha perdido trascendencia en el dominio de superposición temporal entre las variables.

Consideraciones especiales sobre la función de transferencia

En el modelo aqui planteado, se trabajó directamente sobre la transformada de Fourier. Cabe destacar en este punto que como se vió anteriormente, las funciones utilizadas F(t), G(t)y F*(t) son similares variando sólo los coeficientes aj y bj. Por lo tanto al estimar H(t) como cociente de funciones similares, se trata de abarcar un caso general. Esto no es excluyente, ya también se puede trabajar directamente sobre una matriz de relación [cj1 cj2], que establezca transferencia aj0 y aj1 y bj0 y bj1.

Por ejemplo en el caso de verificarse posibles transferencias lineales, se sugiere trabajar con una matriz [cj1 cj2] tal que tal que cj1=aj1/aj0 y cj2=bj1/bj0 de k+2 filas por dos columnas.

Estimando luego los [aj2,bj2] de F*(t), podemos inferir una matriz [aj3,bj3] correspondiente a G*(t), de manera tal que finalmente los vectores de G*(t) resultan:

$$[aj3,bj3]=[cj1\ cj2].[aj2\ bj2]$$

Para este caso la transferencia se supuso lineal y constante en los aj,bj. Pero esto no significa que sea la ùnica transformaciòn posible, sino que indica el camino a seguir por sucesivas investigaciones.

Una recomendaciòn que surge de las distintas experiencias pràcticas realizadas para este trabajo es el de distinguir un dominio de error màs restringido cuando se trabaje directamente sobre los aj y bj.

Los vectores aj y bj de cada funciòn estimada, pueden sugerir transformaciones lineales, pero dentro de los cuales cualquier margen de error puede transformar en algo completamente impredecible a la transferencia esperada.

En otros tèrminos, lo que quiere destacarse es la mayor amplitud posible de errores en las observaciones en los vectores aj y bj.

Si estimamos dos funciones de una misma serie con alguna modificaciòn en una o algunas observaciones, probablemente estas modificaciones no alteren en forma sustancial el resultado final. Mientras que si variamos algùn valor de los vectores aj,bj arbitrariamente, es posible que al reestimar la funciòn nos quede un resultado muy alejado de la funciòn original.

De cualquier forma, aùn teniendo en cuenta èstas restricciones, es recomendable continuar experimentando con estos vectores.

PARTE III - Aplicaciones prácticas del modelo

Un caso de predicción causal entre activos financieros

De todas las aplicaciones con que se ha experimentado al desarrollar este trabajo, puede resultar bastante ejemplificador este sencillo modelo de predicción entre activos financieros, por modelar un caso real además por abarcar todas las posibilidades del método (análisis causal, predicción, explicatividad).

Desde el punto de vista técnico, se prepararon tres programas de computación, realizados en una computadora IBM PC/AT. El primero de ellos es el programa de estimación general, que calcula la función de Fourier para una serie dada, grafica ambas (observada y estimada), y nos crea un archivo que contiene los vectores aj y bj de cada estimación. El segundo programa relaciona dos funciones desfasadas tantos periodos como periodos de retardo causal existan, calcula la función de transferencia intertemporal y una vez ingresada la nueva observación de la variable "causa", estima los valores máximo y mínimo esperado de la variable efecto [6]. El último programa es un actualizador de archivos que al ingresar la nueva observación de una variable de una variable, borra la primera y anexa esta última. De esta forma se mantienen T observaciones en el archivo.

Variables utilizadas en el modelo

El análisis causal que se explica en este ejemplo es la relación existente entre el dolar financiero y el bonex 82 bursátil contado. En economia es bastante conocido el hecho de que el precio de cierre del dólar influye sobre el precio de apertura del bonex al dia siguiente.

[6] Para estimar la función de transferencia intertemporal se utilizaron varios métodos. En aquellos que se trabajó directamente sobre los vectores aj y bj se obtuvieron resultados diferentes, dependiendo de la relación considerada. Los resultados aqui publicados no son los exactos pero si los más generales por utilizar Fourier directamente.

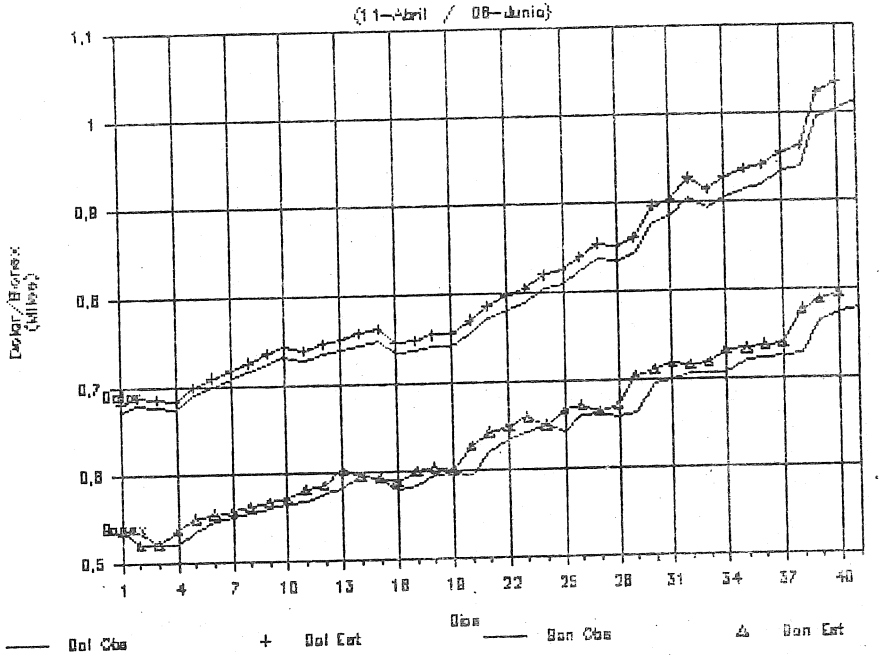
El inconveniente para estimar predicción en este caso, era la falta de un método adecuado y exacto. Suponiendo un dia de retardo en la causalidad, los datos observados y estimados de ambas variables fueron:

Fecha	Dolar	Bonex	Dol Est	Bon Est	Dvio Dol	Dvio Bon
11-Abr	671,00	538,00	680,00	-	0,013412	-
12-Abr	679,00	535,00	688,42	536,84	0,013874	0,003442
13-Abr	676,00	520,00	685,26	521,05	0,013701	0,002023
14-Abr	675,00	520,00	684,21	521,05	0,013647	0,002023
15-Abr	690,00	534,00	700,00	535,79	0,014491	0,003352
18-Abr	699,50	547,00	710,00	549,47	0,015012	0,004521
19-Abr	708,00	551,00	718,95	553,69	0,015461	0,004872
20-Abr	715,00	555,00	726,32	557,90	0,015826	0,005216
21-Abr	725,00	559,00	736,84	562,11	0,016333	0,005554
22-Abr	732,00	564,00	744,21	567,37	0,016680	0,005971
25-Abr	728,00	567,00	740,00	570,53	0,016483	0,006218
26-Abr	734,00	577,00	746,32	581,05	0,016777	0,007024
27-Abr	739,00	581,00	751,58	585,26	0,017023	0,007337
28-Abr	745,00	598,00	757,89	603,16	0,017306	0,008627
29-Abr	749,00	590,00	762,11	594,74	0,017496	0,008027
02-May	735,00	579,00	747,37	583,16	0,016827	0,007181
03-May	736,00	584,00	748,42	588,42	0,016876	0,007570
04-May	742,00	595,00	754,74	600,00	0,017167	0,008403
05-May	742,00	598,00	754,74	603,16	0,017164	0,008625
06-May	757,00	595,00	770,53	600,00	0,017867	0,008403
09-May	772,00	621,00	786,32	627,37	0,018544	0,010254
10-May	782,50	635,00	797,37	642,11	0,019000	0,011190
11-May	789,00	641,00	804,21	648,42	0,019277	0,011577
12-May	804,00	651,00	820,00	658,95	0,019899	0,012207
13-May	810,00	642,00	826,32	649,47	0,020143	0,011640
16-May	823,00	659,00	840,00	667,37	0,020657	0,012698
17-May	837,00	663,50	854,74	672,11	0,021191	0,012970
18-May	834,00	657,00	851,58	665,26	0,021077	0,012576
19-May	843,00	663,00	861,05	671,58	0,021415	0,012938
20-May	877,00	694,00	896,84	704,21	0,022624	0,014713
23-May	883,00	700,00	903,16	710,53	0,022828	0,015037
24-May	906,00	707,00	927,37	717,90	0,023584	0,015410
26-May	893,00	706,00	913,68	716,84	0,023162	0,015356
27-May	906,00	707,50	927,37	718,42	0,023586	0,015436
30-May	915,00	721,00	936,84	732,63	0,023871	0,016133
31-May	920,00	722,00	942,11	733,68	0,024027	0,016182
01-Jun	934,00	726,00	956,84	737,90	0,024457	0,016384
02-Jun	940,00	727,00	963,16	738,95	0,024635	0,016434
03-Jun	997,00	764,00	1023,16	777,89	0,026235	0,018185
06-Jun	1006,00	774,00	1035,16	788,42	0,028983	0,018631
07-Jun	1015,00	780,00	-	795,22	-	0,019511

Fuente: "Ambito Financiero"

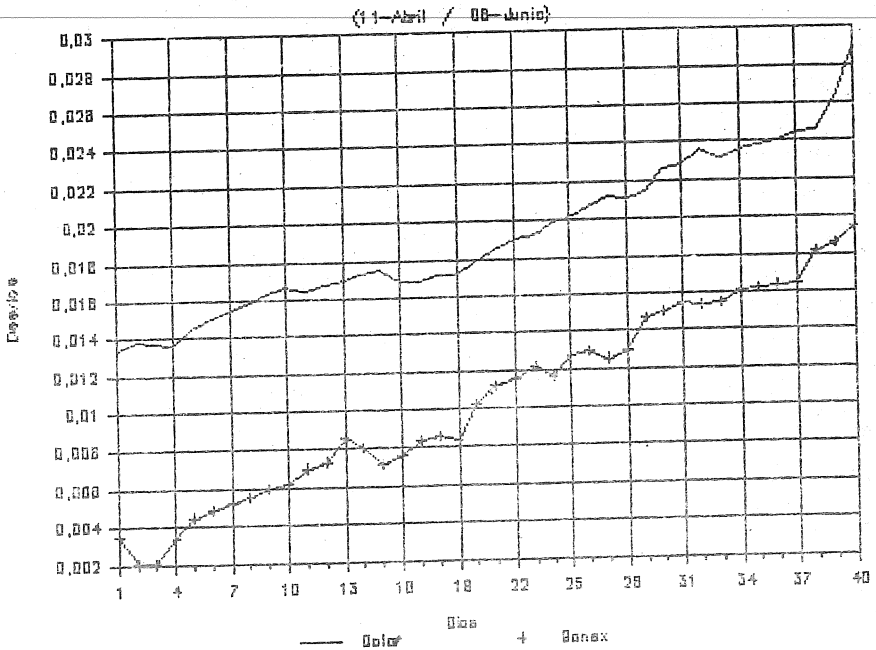
Evolucion Dolar – Bonex

(11–Abril / 08–Junio)



Desvios de los estimaciones

(11–Abril / 08–Junio)



Los vectores estimados para las funciones de Dòlar y Bònex son:

	Funcion Dolar		Funcion Bonex	
k	aj	bj	aj	bj
0	11343,00	0,000	5030,000	0,000
1	194,106	-1850,56	70,912	-1572,03
2	-422,849	-977,682	-164,133	-737,903
3	-220,806	-697,929	-98,422	-579,585
4	-248,381	-412,018	-56,129	-307,758
5	-199,897	-424,334	-95,463	-413,733
6	-264,272	-317,766	-96,831	-280,081
7	-347,627	-281,140	-99,378	-189,267
8	-284,832	-202,200	-110,798	-175,008
9	-274,930	-197,990	-113,530	-220,144
10	-345,998	-210,995	-199,499	-75,498
11	-339,529	-183,961	-167,816	-112,251
12	-331,112	-140,315	-157,868	-86,461
13	-326,493	-162,642	-215,322	-46,722
14	-347,690	-83,215	-139,907	-45,086
15	-320,103	-100,341	-172,537	=60,7368
16	-311,667	-109,851	-166,702	-4,987
17	-373,802	-69,097	-180,861	-24,400
18	-379,180	-32,305	-129,625	13,233
19	-360,901	8,747	-137,574	-5,183
20	-331,000	0,009	-167,000	0,004

Las series estimadas observaron un desvio standard de 0,00787 para la del dòlar, mientras que para la serie del bonex fue de 0,00860.

Cuando se ingresò en el programa de predicciòn el valor del dòlar para el 6 de Junio, correspondiendo 1006, la predicciòn nos entregò un valor esperado para el bonex mìnimo de 778,1668 y un màximo de 780,5550. Es interesante observar que la variaciòn entre los limites màximo y mìnimo estimados en T+1 es un intervalo que abarca el tres mil del valor del bonex en T (0,0030855).

Interpretaciòn de los resultados

Es evidente en este caso que para el 7 de Junio se espera un ascenso en el precio el bonex. Pero ha sucedido tambièn en anteriores experiencias que el precio esperado estè contenido en un entorno de la ùltima observaciòn, lo que significa que no se sabe si el precio

subirá o bajará. Por eso es fundamental la interpretación que se le dará al resultado obtenido.

Volviendo al problema especifico, la decisión a tomar se interpretó de la siguiente manera:

El bónex aumementará de precio para el 7 de Junio con una valor comprendido entre 778,1668 y 780,5550, pero dado que el dólar se muestra creciente , no sólo en este último paso sino en general en la serie completa, y que en la última semana se observa un salto atenuado, es muy probable que el precio se ubique muy cerca del limite superior.

Por lo tanto puede interpretarse el resultado con un valor esperado levemente inferior a los 780,5550.

El 7 de Junio el precio del bonex fue de 780 australes.

Es posible que pueda creerse este resultado un poco casual, o por lo menos que se dude de su continuidad predictiva, pero debido al caracter dinámico del modelo, al ingresar otro nuevo valor del dólar (7 de Junio) de 1015 y actualizar y reajustar el modelo, los resultados fueron los siguientes:

Valores esperados para el precio del bonex en el dia 8 de junio, minimo: 804,0223 y máximo: 806,4106.

Con respecto a la interpretación de resultados, en este tramo más atn se refuerza la idea de ubicarse cerca del limite superior, ya que el salto del dólar es significativo.

El 8 de junio, el bonex se cotizó a 803 australes.

Una consideración especial merece el nivel de retardo con que se trabajó. Si bien los resultados obtenidos realmente son confiables, en base a varias experiencias realizadas para diferentes momentos históricos, se encontró que existen periodos donde el dólar anticipa al

bonex en dos dias. Por lo tanto el consejo para este tipo de experiencias es estimar la serie con un retardo, luego con dos, y luego con dos retardos pero salteando un valor de dólar, es decir si el intervalo de dólar abarca de 1 a T, y el de bónex de 3 a T+2, ingresar el valor de dólar en T+2.

Comparando los resultados obtenidos con uno y con dos retardos, podemos detectar si el retardo actual es de dos periodos. El factor de comparación resulta de la tercer alternativa, ya que ésta nos da una tendencia de corto plazo, y podemos comparar que estimación se aproxima más a ella.

Restricciones y utilidades del modelo

Para el caso de éste ejemplo, existe una restricción fundamental, que está dada por el nivel de información del total de los operadores. Si todos los operadores o un número significativo de ellos conocen la confiabilidad y el resultado de este sistema, mañana seguramente elevarán o disminuirán el nivel de oferta o demanda de bonex de forma tal que el precio esperado caiga o se eleve. Por eso la restricción principal del modelo es que el monto total de operaciones o la cantidad de operadores que actúen según lo indica el modelo no alteren el nivel de equilibrio esperado. En caso de suceder asi una posible utilidad está dada en plantear modelos de retroalimentación de forma tal que cuando todos los operadores ubiquen el precio fuera del equilibrio, ésta situación altere el precio de otro activo, y que los poseedores de un grado más de información trabajen en realidad sobre este último activo. De cualquier forma el modelo queda abierto a nuevas experiencias, algunas de las cuales ya se están realizando, como puede ser el caso de modelos de causalidad simultánea entre más de dos activos financieros.

BIBLIOGRAFIA:

1- Pulido San Romàn, Antonio - Modelos Economètricos-, Ed. Pirà-mide, Madrid,1983.

2- Wallis, K. F. -Some Recent Developments in Applied Econometrics: Dynamic Models ans Simoultàneous Equation Systems - , Journal of Economic Literature. Vol. Vii No 3.

3- Moreno Otero, J. M. -Lògica y Limitaciones de la Econometria - ,Ediciones ICE.

4- Granger C.W.J. & Newbold - The Time Series Approach to Econometric Model Building - En Sims C. (ed) New Methods of Bussines Cycle Research Federal Reserva Bank of Minneapolis.

5- Fontao, Rafael - Apuntes sobre dinàmica de sistemas y simulación por computadora -, inèdito, sobre seminario dictado en la Universidad Nacional del Sur, 1987.

6- Sims, Cr - Money, Income and Causality - American Economic Review, 1972, Vol 62, pp 540-562

7- Zellner, A. - Causality and Econometrics .En A. Zellner: Basic Issues in Econometrics, Univ. of Chicago Press.

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa.
Buenos Aires, Setiembre 1988.

UN ALGORITMO PARA EL CALCULO DE LA PRUEBA ESPECTRAL EN EL CASO DE GENERADORES CONGRUENCIALES LINEALES GCL(A,C,M) CON PERIODO MAXIMO

Rafael A. Pastoriza

ESCUELA DE COMPUTACION, FACULTAD DE CIENCIAS,
UNIVERSIDAD CENTRAL DE VENEZUELA

Apartado 47002, Caracas, Venezuela.

RESUMEN

Se propone un nuevo algoritmo computacional - SP - para resolver la prueba espectral. Está basado en la sucesiva resolución de problemas simples de programación entera cuadrática sujeta a restricciones lineales. El algoritmo propuesto evita problemas numéricos y de terminación.

1. INTRODUCCION

La prueba espectral para generadores congruenciales lineales - **GCL** - es una prueba de carácter "teórico" que tiene por objeto evaluar a priori la perfomance relativa de un **GCL(A,C,M)** con respecto a las propiedades aleatorias teóricas esperadas, conociendo sólo el valor numérico de los parámetros **A, C, M** que especifican un determinado generador.

$$X_{n+1} = A * X_n + C \pmod{M}$$

Al decir de **Knuth [8, p.82]** la importancia de la prueba espectral radica no sólo en el hecho de que todos los buenos generadores conocidos la pasan, sino que para todas aquellas sucesiones, reconocidamente malas, los generadores de números aleatorios - **GNA** - que las producen no pasan dicha prueba.

Es bien conocido (ver **Marsaglia [10]**) que si se dibujan las sucesivas n-uplas $Y_i = (X_i, X_{i+1}, \ldots, X_{i+n-1})$, $Y_{i+1} = (X_{i+1}, X_{i+2}, \ldots, X_{i+n})$, $\ldots$, Y_k, $\ldots$ donde los valores que toman las coordenadas X_i provienen de un generador GCL(A,C,M) los puntos Y_i caen todos en un reticulado del espacio de n-dimensiones. En general ese reticulado es regular, y este es el caso para generadores GCL con período máximo.

La prueba espectral, originalmente propuesta por **Coveyou y Macpherson [1]**, proporciona un método para encontrar la distancia M/F_n entre la familia de hiperplanos que está más separada, en n-dimensiones, y calcula un valor G_n que mide esa distancia. La idea presentada por esos autores era desarrollar un método de análisis de los **GNA**, aplicable a todos los métodos prácticos de generación, y ellos propusieron con ese fin realizar el análisis de Fourier de las sucesiones resultantes de operar los **GNA**.

Se tiene (ver **Knuth [8, p.83]**) que para sucesiones "verdaderamente" al azar provenientes de la distribución uniforme, la transformada de Fourier de la función constante $1/M^n$ será siempre **uno** para los valores congruentes con cero (mod M), y **cero** en todo otro caso.

Cuando un **GCL(A,C,M,X0)** tiene período máximo su transformada de Fourier $F(S_1, \ldots, S_n)$ de las n-uplas es :

$$e^{(-2\pi C\, i/M) * (S(A)-S(i))/(A-1)} \quad \text{si } S(A) = 0 \pmod M$$

$F(S_1,\ldots,S_n) =$

$$0 \quad \text{si no}$$

donde

$$S(A) = S_1 + S_2 * A + \ldots + S_n * A^{n-1}$$

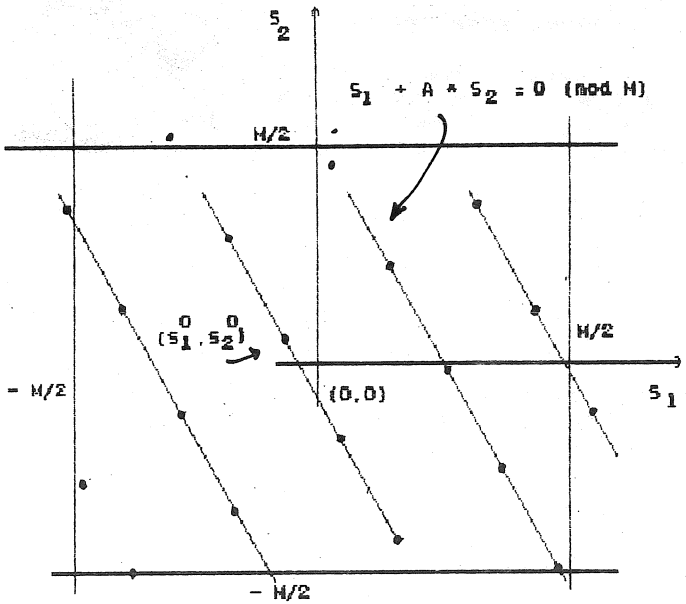
que es la expresión de lo que se denomina el espectro de un **GCL(A,C,M,X0)**.

En el texto de **Fishman [2, p.363]** se pueden encontrar también las expresiones de las transformadas de Fourier para los generadores **GNA** que no tienen período **máximo**, como son los generadores congruenciales lineales multiplicativos modulo primo - **GCLMMP** -.

Entonces una manera de medir la uniformidad de un **GCL(A,C,M)**, en **n** dimensiones, es medir para cuales puntos $(S_1,S_2, \ldots ,S_n)$ se aproxima $F(S_1,S_2, \ldots ,S_n)$, la transformada de Fourier de la sucesión a $f(S_1,S_2, \ldots ,S_n)$, ya que se puede ver que la transformada para las n-uplas es :

$$f(S1,S2,\ldots,Sn) = \begin{cases} 1 \text{ si } S1=0 \pmod M, \ldots, Sn=0 \pmod M \text{ y } S(A)=0 \pmod M \\ 0 \text{ si no} \end{cases}$$

Se puede entonces tratar de identificar el punto $(S_1,S_2, \ldots ,S_n)$ en **S(A) = (mod M)** más cercano al punto **(0,0, ... ,0)**. La figura siguiente proporciona una imagen del problema asociado para el caso de dos dimensiones.



En la prueba espectral se usa la norma euclídea para caracterizar el vector $(S_1, S_2, \ldots, S_n)$:

$$N_n(A) = (S * S^t)^{1/2} = (S_1^2 + \ldots + S_n^2)^{1/2}$$

Luego para seleccionar el peor caso posible, se debe encontrar el punto

$$S = (S_1^0, \ldots, S_n^0) \text{ con } S_i \text{ enteros}$$

que

minimiza $N_n(A)$

sujeto a

$$S(A) = 0 \pmod M$$

El análisis se facilita (ver **Knuth [1, p.104]**) si se cambia el dominio de los valores $0 <= S_i < M$ al $-M/2 <= S_i < M/2$, y entonces los S_i no deben ser todos cero.

2. LA PRUEBA ESPECTRAL

En resumen, dado un generador congruencial lineal GCL(A,C,M) con los coeficientes especificados de manera tal que tenga período máximo (ver **Hull y Dobell [7, p.233]**), la prueba espectral se reduce al siguiente procedimiento :

1. Dado **n**, valor de la n-uniformidad que se desea evaluar apriori, resolver el siguiente problema :

problema **PRESP** :

$$F_n = (S_1^2 + \ldots + S_n^2)^{1/2} \quad \text{-----> min}$$

sujeto a:

$$-M/2 <= S_i < M/2 \quad \text{con } i = 1,2, \ldots, n$$

$$(S_1,S_2, \ldots ,S_n) =/= (0,0, \ldots ,0)$$

$$S_i \text{ en } Z[-M/2,M/2], \text{ enteros, con } i = 1,2, \ldots ,n$$

$$\mathbf{S(A) = S_1 + A * S_2 + \ldots + A^{n-1} * S_n = 0 \pmod M}$$

2. Evaluar el valor C_n, propuesto por **Knuth [8, p.87]** donde :

$$\mathbf{C_n = (pi\ F_n)^n/((n/2)!\ M)}$$

donde el símbolo **(n/2)!** está definido por la expresión

$$\mathbf{(n/2)! = (n/2)\ (n/2-1)\ \ldots\ (1/2)\ (pi)^{1/2}} \text{ para } \mathbf{n} \text{ impar}$$

El valor de C_n es una cantidad que mide la efectividad de multiplicador **A** en una sucesión proveniente de un generador **GCL** con período máximo. Valores pequeños de C_n corresponden a generadores **GCL** no aleatorios.

El algoritmo S propuesto por **Knuth [8, p.93]** proporciona un procedimiento computacional para realizar la prueba espectral, y fué implantado en FORTRAN por **Golder [3, p.173]** con algunas incorrecciones. Tiempo después las condiciones correctas fueron dadas por **Hoaglin y King [5, p.376]**.

Posteriormente **Hopkins [6, p.328]** en 1983 publica otra version FORTRAN del algoritmo **S**, que se corresponde con la nueva versión corregida para evitar problemas de terminación, e incluída por **Knuth** en la segunda edición del volumen II de su texto The Art of Computer Programming **[9, p.98]**.

Cualquiera sea el caso dicho algoritmo requiere de un esfuerzo computacional muy grande, y en la práctica no se lo ha usado en equipos mainframe para evaluar la perfomance de los **GCL** más conocidos, más allá de n = 6. La búsqueda exhaustiva implícita en el algoritmo hace que el tiempo de ejecución crezca como 3^n, (ver **Knuth [9, p.100]**).

3. EL ALGORITMO SP

Aquí se propone un procedimiento computacional diferente y muy simple para resolver el problema **PRESP**.

Se trata de resolver de manera secuencial el problema asociado, definido por el funcional cuadrático, y como el término S(A) se puede escribir

$$S(A) = S_1 + A(S_2 + A(S_3 + \ldots + A(S_{n-1} + AS_n)) \ldots) = 0 \pmod M$$

emplear entonces el algoritmo SP siguiente :

ALGORITMO SP

Resolver sucesivamente los problemas auxiliares :

PAUX1 : $\min F_1(S_{n-1},S_n) = S_{n-1}^2 + S_n^2$

sujeto a:

$$-M/2 <= S_{n-1} < M/2$$

$$-M/2 <= S_n < M/2$$

S_i en $Z[-M/2,M/2]$, enteros, i = n-1,n

$$H_1(A) = S_{n-1} + A * A_n = 0 \pmod M$$

$$(S_{n-1},S_n) =/= (0,0)$$

Sea (S_{n-1}^*,S_n^*) una solución óptima de **PAUX1** (si existe), luego resolver el siguiente problema auxiliar :

PAUX2 : min $F_2(S_{n-2},H_1) = S_{n-2}^2 + F_1(S_{n-1},S_n)$

sujeto a:

$$-M/2 <= S_{n-2} < M/2$$

$$-M/2 <= H_1 < M/2$$

S_{n-2},H_1 en **Z[-M/2,M/2]**, enteros

$H_2(A) = S_{n-2} + A * H_1 = 0 \pmod M$

$(S_{n-2},H_1) =/= (0,0)$

Sea entonces (S_{n-2}^*,H_1^*) una solución óptima de **PAUX2** (si existe), lo cual implica resolver también el problema **PAUX1** con el valor $H_1 = S_n^*$, cuya solución, si existe, será (S_{n-1}^{**},S_n^{**}). Luego una solución óptima de **PAUX2** será $(S_{n-2}^*,S_{n-1}^{**},S_n^{**})$. Y así sucesivamente.

Es claro que este procedimiento se deberá repetir tantas veces como sea necesario para resolver el problema **PRESP**.

Luego para resolver el problema **PRESP** mediante el algoritmo **SP**, basta saber resolver el problema **PAUX** con un término genérico B cualquiera :

PAUX : min $F(x,y) = x^2 + y^2$

sujeto a :

$$-M/2 <= x < M/2$$

$$-M/2 <= y < M/2$$

x,y en **Z[-M/2,M/2]**, enteros

$H(A,B) = x + A * y = B \pmod M$

$(x,y) =/= (0,0)$

y construir las soluciones $(S_1^0,S_2^0, \ldots ,S_n^0)$ al problema original **PRESP** en función de las soluciones (x^0,y^0) a los sucesivos problemas auxiliares. Luego, conel valor de las soluciones se deberá, como antes, evaluar el valor de C_n .

4.REFERENCIAS

1. **Coveyou, R.R. y Macpherson, R.D.** :"Fourier Analysis of Uniform Random Number Generators", J.A.C.M., vol. 14, num. 1, p.100/109, (1967).

2. **Fishman, G.S.** :"Principles of Discrete Event Simulation", ed. J.Wiley, N.Y., (1970).

3. **Golder, E.R.** :"The Spectral Test for the Evaluation of Congruential Pseudo-random Generators", Appl. Stat., vol. 25, num. 2, p.173/180, (1976).

4. **Golder, E.R.** :"Remark ASR18 : The Spectral Test for the Evaluation of Congruential Pseudo-random Generators", Appl. Stat., vol. 25, num. 3, p.324, (1976).

5. **Hoaglin, D.C. y King, M.L.** :"Remark ASR24: A Remark on Algorithm AS 98 : The Spectral Test for the Evaluation of Conguential Pseudo-random Generators", Appl. Stat., vol. 27, num. 3, p.375/377, (1978).

6. **Hopkins, T.R.** :"A Revised Algorithm for the Spectral Test", Appl. Stat., vol. 32, num.3, p.328/334, (1983).

7. **Hull, T.E. y Dobell, A.R.** :"Random Number Generators", SIAM Rev., vol. 4, p.229/254, (1962).

8. **Knuth, D.E.** :"The Art of Computer Programming", vol. 2, ed. Addison-Wesley, Reading, Massachusetts, p.82, (1969).

9. **Knuth, D.E.** :"The Art of Computer Programming", vol. 2, second edition, ed. Addison-Wesley, Reading, Massachusetts, p.89, (1969).

10. **Marsaglia, G.** :"Random numbers fall mainly in the planes", Proc. Mat. Acad. Sci. U.S.A., vol. 61, p.25/28, (1968).

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

TECNICAS EFICIENTES PARA GENERAR VALORES NORMALES Y LOGNORMALES POR EL METODO DEL COCIENTE DE UNIFORMES

R.LAVIERI, R.A.PASTORIZA

Escuela de Computación, Facultad de Ciencias,
Universidad Central de Venezuela.
Apartado 47002. Caracas, Venezuela.

RESUMEN

Se presentan algoritmos de composición basados en el cociente de uniformes para generar valores de variables normales y lognormales. Las pruebas realizadas permiten asegurar una alta eficiencia en relación a la velocidad de ejecución y almacenamiento de estos algoritmos.

INTRODUCCION

El método del cociente de uniformes fue introducido en 1977 por **Kinderman y Monahan [1]**, y resultó particularmente eficiente para generar valores de variables aleatorias normales, aún cuando se trata de un método general para generar valores de ciertas variables aleatorias mediante el muestreo uniforme en una región R del plano en lugar de generarlos empleando la función de distribución $F_x(x)$.

La idea central de éste método para generar variables aleatorias con una distribución deseada $F_x(x)$ consiste en generar un punto uniformemente en una región adecuada R del plano. El cociente de los valores de las coordenadas de un punto interior a R proporciona un valor de la variable aleatoria que se reparte según la función de distribución especificada. En la implantación casi siempre se usa la técnica del rechazo ya que en general la región de muestreo R suele ser irregular; es conocido que la eficiencia de un algoritmo de éste tipo depende muy fuertemente de la aproximación que se haga mediante la región de acotación C.

Los puntos se generan uniformemente en C, suponiendo que el muestreo en esa región se puede realizar fácilmente; aquellos ensayos cuyos puntos caen fuera de la región R se rechazan, y se vuelve a probar. La eficiencia global del algoritmo depende del valor que tiene e = [R]/[C], cociente de las respectivas áreas.

En general un algoritmo de este tipo se puede mejorar más aún, dividiendo la región R en subregiones disjuntas R_i cada una de las cuales, a su vez, es encerrada en su propia región de acotación C_i.

En síntesis, se estructura entonces un algoritmo de composición donde se emplean los valores de probabilidad $p_i=[R_i]/[C_i]$ para seleccionar la subregion R_i, y luego, al igual que antes, se utiliza la técnica del rechazo para generar los valores deseados.

El uso de pruebas previas que definen regiones de aceptación-rechazo automáticas, poco costosas de realizar computacionalmente, suele mejorar notablemente aún más la eficiencia global del algoritmo.

El método para generar una variable aleatoria X, con densidad $f_X(x)$ es emplear la región R definida mediante la expresión

$$R = \{ (u,v) / 0 <= u <= (f(v/u))^{1/2} \}$$

En estas circunstancias, si el par de variables aleatorias (u,v) está distribuido uniformemente en esa región R, el valor del cociente **X = v/u** tiene la densidad deseada $f_X(x)$.

NORMAL

Para la generación de valores normales $N(mu,sigma^2)$ se ha propuesto generar valores **X** normales standard y efectuar la transformación

Z = sigma * X + mu

Así tomando la función $g(t) = e^{-t^2/2}$

la región definida por

$$u^2 <= g(v/u)$$

queda dada por la expresión

$$(v/u)^2 <= -4 * \ln u$$

En 1980 **Robertson y Walls [2]** describen el uso de ésta técnica para en la generación de valores aleatorios normal y gamma.

Estos autores encierran la región de muestreo en un trapecio compuesto a su vez por dos triángulos y un rectángulo, con zonas de aceptación-rechazo automáticas.

Posteriormente **Knuth [3]** propone también zonas de aceptación-rechazo automáticas, pero su método se basa en encerrar la región de muestreo mediante figuras curvas.

Aquí propondremos algunas pequeñas mejoras a estas ideas, modificaciones, en algunos aspectos, que permiten acelerar ligeramente el tiempo esperado de ejecución del algoritmo.

La primera modificación está basada en la siguiente observación :

Sea una variable aleatoria X == $N(0,sigma^2)$, entonces el valor transformado siguiente se distribuye también como una normal

$$\{(sigma1/sigma) * X + mu \} == N(mu,sigma1^2)$$

Tomando como función la

$g(t) = e^{-t^2/2*sigma^2}$

entonces la región de aceptación es

$(v1/u)^2 <= -4 * sigma^2 * \ln u$ con $u > 0$

Así se propone generar valores $N(0,sigma^2)$ con varianza $sigma^2 = e/8$, de manera que los valores aceptados queden enteramente contenidos en la región

$[0,1] * [-.5,.5]$

y utilizar posteriormente la transformación

$z = (sigma1/sigma) * X + mu$

para generar valores $N(0,sigma1^2)$.

Algoritmo normal (mu,sigma1)

inicialización

$A = (e/8)^{1/2}$; $B = e/8$; $C = sigma1/A$;

$D = 5 * B$; $E = 0.5 * e^{1.25}$; $F = 0.5 * e^{-.35}$;

$G = 1.4 * B$; $H = -0.5 * e$; $P = 27/16$

paso 1.

generar (u,v) == unif[0,1]*[0,1],
poner X = (v- 0.5)/u

paso 2.

poner $T = X^2$

si $T <= D - E * u$, poner $z = C * X + mu$,
devolver z y finalizar

en caso contrario, continuar

paso 3.

si $T >= F / u + G$, ir al paso 1

en caso contrario, continuar

paso 4.

si T <= H * ln u, poner z = C * X + mu,
devolver z y finalizar

en caso contrario, ir al paso 1

Los pasos 2 y 3 dependen muy fuertemente de la velocidad con la cual se realiza la operación de logaritmación. Si ésta operación es relativamente lenta respecto a las operaciones aritméticas simples, se propone otra zona de rechazo automática.

La nueva zona se puede describir de manera muy simple mediante la expresión

$(v/u)^2 >= P * (1-u)/8$, con $u > 2/3$

cuya implantación es el paso 3.5

paso 3.5

si { u > 2/3 } y { T >= P * (1-u) } ir al paso 1

Pruebas comparativas realizadas con equipos mainframe y microcomputadores - TABLA 1 - permiten asegurar que se obtiene una mejora del orden de un 15% en el tiempo de ejecución, utilizando FORTRAN, respecto a las otras implantaciones comparadas [2] y [3].

LOGNORMAL

Para la generación de valores lognormales $LN(0,sigma^2)$ tomamos la función

$g(t) = 1/t * e^{-logt^2/2*sigma^2}$

En este caso la región queda definida por :

$2 * sigma^2 * (logu + logv) <= -(logv - logu)^2$

Es fácil ver que la región R queda enteramente contenida en el cuadrado

$[0, e^{sigma^2/4}] * [0, e^{sigma^2/4}]$

y que es simétrica respecto a la recta u = v.

De todas las posibles regiones R(sigma) se seleccionó la definida por

$$sigma = 2^{1/2}$$

ya que la misma queda enteramente por debajo de la recta

$$u + v = 2$$

lo cual permite gran simplicidad en la zona de aceptación-rechazo automática.

Sea (u0,v0) un par uniforme en el cuadrado

$$[0,e^{1/4}] * [0,e^{1/4}]$$

Podemos entonces definir una zona de rechazo automática

$$R1 = \{ (u0,v0) / u0 + v0 > 2 \}$$

y una zona de aceptación automática

$$R2 = \{ (u0,v0) / (v0,u0) <= e^2 \text{ y } (u0,v0) >= e^{-2}$$

$$\text{y } (u0 + v0 <= e^{-3/2} + e^{1/2}) \}$$

Ahora bien si u == unif[0,1] y v == unif[0,1] entonces

$$u0 = e^{1/2} * u$$

y

$$v0 = e^{1/2} * v$$

por lo que v0/u0 = v/u, y podemos además describir las regiones como sigue

$$R((2)^{1/2}) = \{ (u,v) / 1 + \log(u) + \log(v) <=$$

$$- 1/4 * (\log(u) - \log(v)^2 \}$$

$$R1 = \{ (u,v) / u + v > 2 * e^{-1/2} \}$$

$$R2 = \{ (u,v) / (v/u) <= e^2 \text{ y}$$

$$(v/u) >= e^{-2} \text{ y } (u + v <= e^{-2} + 1 \}$$

Ahora bien, si X == lognormal(0,2) entonces

$$e^{mu} * X^{sigma/sqrt(2)} == \text{lognormal } (mu,sigma^2)$$

El algoritmo propuesto es el siguiente :

Algoritmo lognor(0,sigma)

inicialización

$A = 2 * e^{-1/2}$; $B = e^{2}$; $C = e^{-2}$;

$D = 1 + C$; $E = e^{mu}$; F = sigma / sqrt(2)

paso 1.

generar u,v == unif(0,1)

paso 2.

hacer G = u + v

si G > A, ir al paso 1

en caso contrario, continuar

paso 3.

hacer R = v/u ;

si (R <= B) y (R >= C) y (G <= D), ir al paso 5

en caso contrario, continuar

paso 4.

hacer Z = ln(u) y W = ln(v) ;

si (1 + Z + W <= -.25 * $(Z-W)^2$), ir al paso 5

en caso contrario, ir al paso 1

paso 5.

hacer L = E * R^{F} ; devolver L y finalizar

Referencias

1. **Kinderman, A. J. y Monahan,J. F.** :"Computer Generation of Randon Variables Using the Ratio of Uniform Deviates" ACMTOMS, vol. 3, num. 3, p 257/260, 1977.

2. **Robertson,I. y Walls, L. A.** :"Random Number Generators for the Normal and Gamma Distributions Using the Ratio of Uniforms Method", Harwell AERE - R 10032, 1980.

3. **Knuth, D.E.**:"THE ART OF COMPUTER PROGRAMMING",vol. 2, Reading,Mass., second ed., 1981.

TABLA 1

Tiempos promedio obtenidos generando 100.000 valores tras 20 corridas en un equipo Burroughs B-6700, en FORTRAN

Polar	101.3 seg.
Teorema Central del Limite	68.2 seg.
Robertson-Walls	36.9 seg.
Knuth	34.2 seg.
Lavieri-Pastoriza	29.7 seg.

Apèndice

```
C  ******  LISTADO          ********************************
C                                          ARCHIVO NORMAL.FOR
C  ***************************************************************

      REAL FUNCTION NORMAL (MU,SIGMA)
C
C     OBJETIVO :
C          GENERAR UN VALOR DE VARIABLE ALEATORIA QUE SE
C          DISTRIBUYE SEGUN UNA NORMAL N(MU,SIGMA)
C
C     PROCEDIMIENTO :
C          SE EMPLEA EL METODO DEL COCIENTE DE UNIFORMES,
C          EN BASE A UN ALGORITMO DE COMPOSICION Y LA
C          TECNICA DEL RECHAZO
C
C     VARIABLES DE ENTRADA :
C        MU : MEDIA DE LA FUNCION DE DENSIDAD NORMAL
C     SIGMA : DESVIACION STANDARD DE LA FUNCION DE DENSIDAD
C             NORMAL
C
C     VARIABLES DE SALIDA :
C         Z : VALOR DE LA VARIABLE ALEATORIA
C             NORMAL(MU,SIGMA)
C
C     SUBRUTINAS LLAMADAS :
C         NINGUNA
C
C     REFERENCIAS :
C           KNUTH,D.E.: THE ART OF COMPUTER PROGRAMMING,
C           VOL.2 2ND ED., ADDISON-WESLEY, READING, MASS.,
C           P 121/125, (1981)
C
C           LAVIERI,R.y PASTORIZA,R.A: TECNICAS EFICIENTES
C           PARA GENERAR VALORES NORMALES Y LOGNORMALES
C           POR EL METODO DE COCIENTES DE UNIFORMES.
C
C     PROGRAMADOR :                  U.C.V. FAC.CIENCIAS
C           R.A.P.(1986)             ESCUELA DE COMPUTACION
C
C  ***************************************************************
C          A /(E/8)**1/2/ , B / e/8/    , D / 5 * B/,
C          E /0.5 * E**1.25 /, P / 27/16/ ,F / 0.5 * e**-.35 /,
C          G / 1.4 * (e/8)/ , H / -e/2 / , W / 2/3 / ,
C          FK / 0.45 /

       DATA A /.5829109/, B /.3397852/, D /1.6989261/,
      1 E /1.7451714/ , P  /1.68750/, F /.3523440/,
      2 G /.4756993/ , H /-1.3591409/, W /.6666667/ ,
      3 FK / 0.45 /
```

```
      REAL MU
      C = SIGMA/A
   30 CONTINUE
      U = GNA(S)
      V = GNA(S)
      X = (V-.5)/U
      T = X * X
C
C        PRUEBA PREVIA DE ACEPTACION
C
      IF ( T .LE.(D- E * U) ) GO TO 200
C
C         PRUEBAS PREVIAS DE RECHAZO
C
      IF (( U .LT. FK ) .AND. (T .GE. G + F/U )) GO TO 30
      IF (( T .GE. W ) .AND. (T .GE. (P * (1-U)/B))) GO TO
30
C
C           REGION CURVA PARA LA NORMAL
C
      IF ( T .GT. H * ALOG(U) ) GO TO 30
  200 Z = C * X + MU
      NORMAL = Z
      RETURN
      END
C
C  ***********************************************************
C
```

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

IMPLEMENTACION DE UN MODULO OPTIMIZADOR EN UN SIMULADOR DE PROCESOS QUIMICOS

Dr. Jorge M. Montagna
Dr. Oscar A. Iribarren

INGAR - Instituto de Desarrollo y Diseño
Avellaneda 3657 - (3000) Santa Fe
TE (042) 34451 - 45136

RESUMEN

Hasta hace pocos años, la optimización de un proceso químico utilizando un simulador de procesos modular secuencial era una tarea que demandaba un excesivo tiempo de computación, por lo que prácticamente no se usaba. Con la aplicación del método de Programación Cuadrática Sucesiva de Powel se logra reducir sustancialmente el tiempo de CPU empleado. Este método se ha implementado en el simulador SIMBAD, lo cual ha requerido el análisis de una serie de elementos. Se hizo hincapié en el manejo de la información mediante un Administrador de Base de Datos, pues el módulo optimizador interactúa con el simulador a través de la base de datos. Esto posibilita la aplicación de distintas estrategias para la optimización. Además, la estructura del SIMBAD, que maneja todos los elementos involucrados como programas ejecutables, simplifica la resolución de la optimización. Se ha generado un esqueleto que posibilitará el análisis e incorporación de los distintos elementos de este método: cálculo de gradientes, optimización univariable, resolución del problema cuadrático, etc. Dada la modularidad y el enfoque centrado en los datos con que fue desarrollado el SIMBAD, no ha sido necesario alterar sus elementos para realizar esta información.

1. Introducción

La simulación de procesos por computadora es una herramienta muy utilizada en el desarrollo de un proceso químico. Existen para ello un gran número de simuladores con avanzado desarrollo que calculan los balances de materia y energía de la planta. Casi todos los de uso industrial utilizan para la resolución el enfoque modular secuencial en el cual se resuelven las ecuaciones que modelan el funcionamiento de cada equipo en una subrutina, teniendo como dato las corrientes de entrada y los parámetros del mismo, y calculando sus corrientes de salida. Según aparezcan los equipos que componen la planta en la secuencia de resolución, son llamadas las distintas subrutinas. Las ventajas de este enfoque son, fundamentalmente, su robustez y confiabilidad, a partir del avance en los métodos y algoritmos para resolver cada equipo, y la posibilidad de seguir fácilmente el desarrollo de la simulación al coincidir el modelo computacional con la estructura de la planta.

Uno de los mayores inconvenientes del enfoque modular secuencial está en que el problema de la optimización demanda tiempos de computación tan elevados que imposibilitan su resolución. Normalmente la planta se maneja como si fuera una "caja negra". El módulo optimizador no tiene ninguna relación directa con la ejecución de la simulación y por lo tanto la evaluación de cada uno de los puntos de la sucesión del algoritmo requiere una simulación completa de la planta para poder calcular la función objetivo y las restricciones. Se utilizan métodos de busqueda directa.

Se da un cambio significativo con los trabajos de Biegler y Hughes (1981, 1982, 1985). Por un lado estos autores utilizan el método de Programación Cuadrática Sucesiva (PCS) de Powell (1977), pero además se abandona el esquema en el cual el simulador es independiente del optimizador y por el contrario hay ahora una relación entre ambos. Una serie de trabajos continúan hasta el presente en esta línea, buscando hacer cada vez más confiable, eficiente y rápida la resolución de la optimización de una planta química (Kisala (1985), Biegler (1985), Lang y Biegler (1986), etc).

En este trabajo se describe la implementación de un módulo optimizador para el simulador de procesos SIMBAD (Montagna y otros

(1987), Vecchietti y otros (1987) y Leone y otros (1987)). Tomando como base los desarrollos mencionados se ha utilizado el método PCS. Teniendo en cuenta que el SIMBAD utiliza para el manejo de la información un Administrador de Base de Datos (DBMS) se han aprovechado también en este caso las ventajas derivadas de ello, sobre todo en el sentido de mantener un enfoque centrado en los datos en su desarrollo. Además la estructura modular y flexible del SIMBAD, permite la fácil implementación y evaluación de distintas alternativas para los diferentes pasos de este algoritmo: cálculo de gradientes, optimización univariable, etc. y de las diferentes estrategias de resolución.

2. Optimización de procesos químicos

En principio, la simulación de un proceso químico es la resolución de un gran sistema de ecuaciones que modela la planta. Trabajando con el enfoque modular secuencial se reduce a una serie de subproblemas ordenados de acuerdo a la secuencia de resolcuión. Dado que cada subrutina que calcula un equipo requiere como dato las corrientes de entrada y como en las plantas de procesos químicos son comunes los reciclos, es necesario recurrir en casi todos los casos a métodos iterativos para la resolución. Estos parten de rasgar un conjunto de corrientes, proponer valores para las mismas e iterar repetidamente hasta alcanzar la convergencia. La figura 1 muestra un ejemplo muy simple de un trabajo de Biegler y Hughes (1982), en el cual se ha elegido como corriente de corte la 2, pues no existe ningún equipo que pueda ser resuelto por conocerse todas sus entradas. Proponiendo valores para esta corriente, y recogiendo los valores calculados para la misma del equipo IV, se converge la planta luego de algunas iteraciones.

La simulación puede resumirse entonces como:

$$h(y) = 0 \qquad (1)$$

donde h son las ecuaciones que exigen la convergencia de las corrientes de corte, e y las variables de estas últimas. El resto de las variables se obtiene como resultado intermedio del cálculo de los módulos.

Hasta el presente el problema de optimización se planteaba como:

PROCESO FLASH SIMPLE

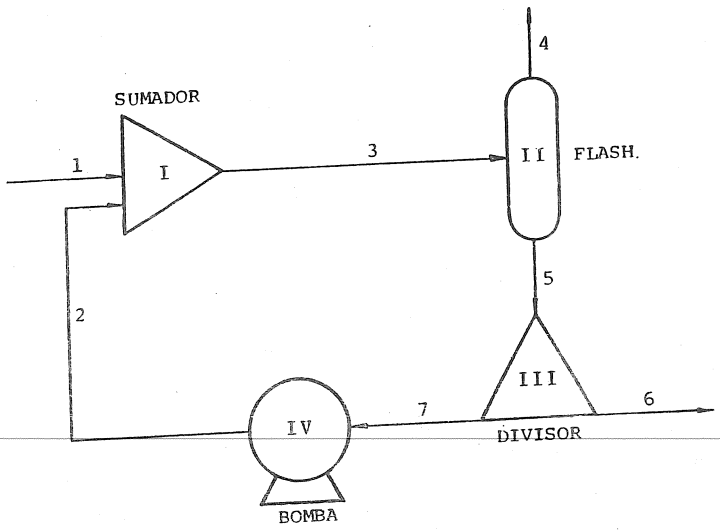


FIGURA 1

$$\text{Min } F(x)$$

sujeto a: (2)

$$c(x) = 0$$
$$g(x) \geq 0$$

donde c y g son las restricciones impuestas a las variables de diseño $\bar{x}$. Este planteo corresponde a la utilización del simulador como una caja negra, pues para cada x se efectúa una simulación completa de la planta, esto es, se resuelve (1).

En el nuevo enfoque se reunen la simulación y la optimización pues se trabaja con el siguiente problema:

$$\text{Min } F(x,y)$$

sujeto a:

$$h(x,y) = 0$$
$$c(x,y) = 0 \qquad (3)$$
$$g(x,y) \geq 0$$

o sea que se optimiza la función objetivo F sujeta a las restricciones h, c y g, las cuales se cumplen en el óptimo, en particular h que exige la convergencia de las corrientes de corte o sea la resolución de los balances de materia y energía del proceso.

De acuerdo a la forma en que se resuelve (3) surgen distintas estrategias: camino no factible (Biegler y Hughes (1982)), camino factible (Biegler y Hughes (1985)) e híbridos entre ambos (Kisala (1985)). Se diferencian en las iteraciones que para un dado x se realizan con distintos valores de y con (1). En el caso de los métodos de camino factible en cada iteración de la optimización se resuelve la simulación. Para camino no factible, se convergen al mismo tiempo ambos problemas. La ventaja del camino factible es que, si la optimización fracasa, se cuenta con una solución de la planta que, si bien no es óptima, permite realizar distintos análisis. Sin embargo, si la optimización concluye, se ha gastado tiempo de CPU resolviendo la simulación en puntos alejados del óptimo. Conclusiones opuestas se pueden obtener para camino no factible. Los métodos híbridos, si bien no exigen la convergencia de la simulación como en camino factible, permiten la aproximación de la solución realizando un determinado número

de iteraciones sobre la secuencia de resolución de la planta.

3. Método de Programación Cuadrática Sucesiva

Teniendo en cuenta el problema (3) el método de Programación Cuadrática Sucesiva (PCS) determina en cada iteración i una dirección de búsqueda d a partir del problema cuadrático:

$$\text{Min } \nabla F(x_i,y_i)^T d + \frac{1}{2} d^T B d$$

sujeto a:

$$\begin{aligned} h(x_i,y_i) + \nabla h(x_i,y_i)^T d &= 0 \\ c(x_i,y_i) + \nabla c(x_i,y_i)^T d &= 0 \\ g(x_i,y_i) + \nabla g(x_i,y_i)^T d &\geq 0 \end{aligned} \qquad (4)$$

La matriz hessiana B se obtiene aplicando la fórmula BFGS (Dennis y Moré (1977)). Sobre la dirección d obtenida de (4) se realiza una optimización univariable para determinar la longitud del paso en esta dirección. Se utiliza una función de penalidad que incluye un término de la función objetivo del problema original más una combinación no lineal de las restricciones violadas, siendo los factores de cada una de ellas función de los multiplicadores de Kuhn-Tucker de las dos últimas iteraciones.

4. Estructura de datos

SIMBAD utiliza una base de datos jerárquica para almacenar la información relativa a la simulación. Pese a ser un sistema network, dado el tipo de relaciones que se precisa manejar, con una estructura jerárquica ha sido posible diseñar el esquema de la base de datos (Vecchietti y otros (1987)). Básicamente éste tiene al tope información general de la planta (nombre, número de equipos, compuestos y corrientes, tipo de resolución, etc) y, en un segundo nivel colgada de la misma grupos de datos de corrientes, equipos y compuestos.

La implementación de la optimización requirió una mínima

modificación del esquema anterior. En la información general se agregaron items relativos a la optimización: tope y contador de iteraciones para la optimización global y la univariable, error máximo permitido, factor de perturbación, etc, y en el segundo nivel, información de las variables de optimización (figura 2).

El esquema logrado no ha implicado ningún cambio en ningún programa al variar la estructura de datos, pues las modificaciones son, desde el punto de vista resolutivo, independientes de lo existente. El único gasto ha sido el compilado y encadenado de todos los programas del simulador para generar la nueva área de trabajo de la base de datos.

Más allá de detallar las ventajas generales de la base de datos o del diseño efectuado en este caso, conviene puntualizar que se realizó un desarrollo centrado en los datos, tanto para el simulador SIMBAD original, como para el optimizador. En el primer caso permitió que la incorporación del módulo de optimización no implicara ningún cambio en el resto de los programas a partir de la flexibilidad con que fue diseñado. En el segundo caso, posibilitará la incorporación y evaluación de distintos elementos para cada uno de los pasos del método PCS y distintas estrategias de optimización.

5. Ejecución de la optimización

SIMBAD utiliza para la simulación dos archivos de comandos anidados. El primero, denominado Archivo Principal de Ejecución (APE) se encarga, principalmente, de generar el Archivo de Operación (AO) teniendo en cuenta la información almacenada en la base de datos en la etapa de carga y durante el preprocesamiento. El AO incluye los comando de ejecución de los programas de los equipos de la planta y otros auxiliares.

La figura 3 es el AO para la simulación de la planta de la figura 1 denominada FLASH1. Al comienzo se abre el fichero FLASH1.TRB que en ciertos casos incluye una cadena POOj de modo de ir a la ejecución del equipo ubicado en la posición j en la secuencia de resolución. Este dispositivo permite saltear la corrida de ciertos equipos, evitando un gasto innecesario en aquellos casos en que se ha realizado una corrida parcial y han surgido inconvenientes que obligaron a detener la

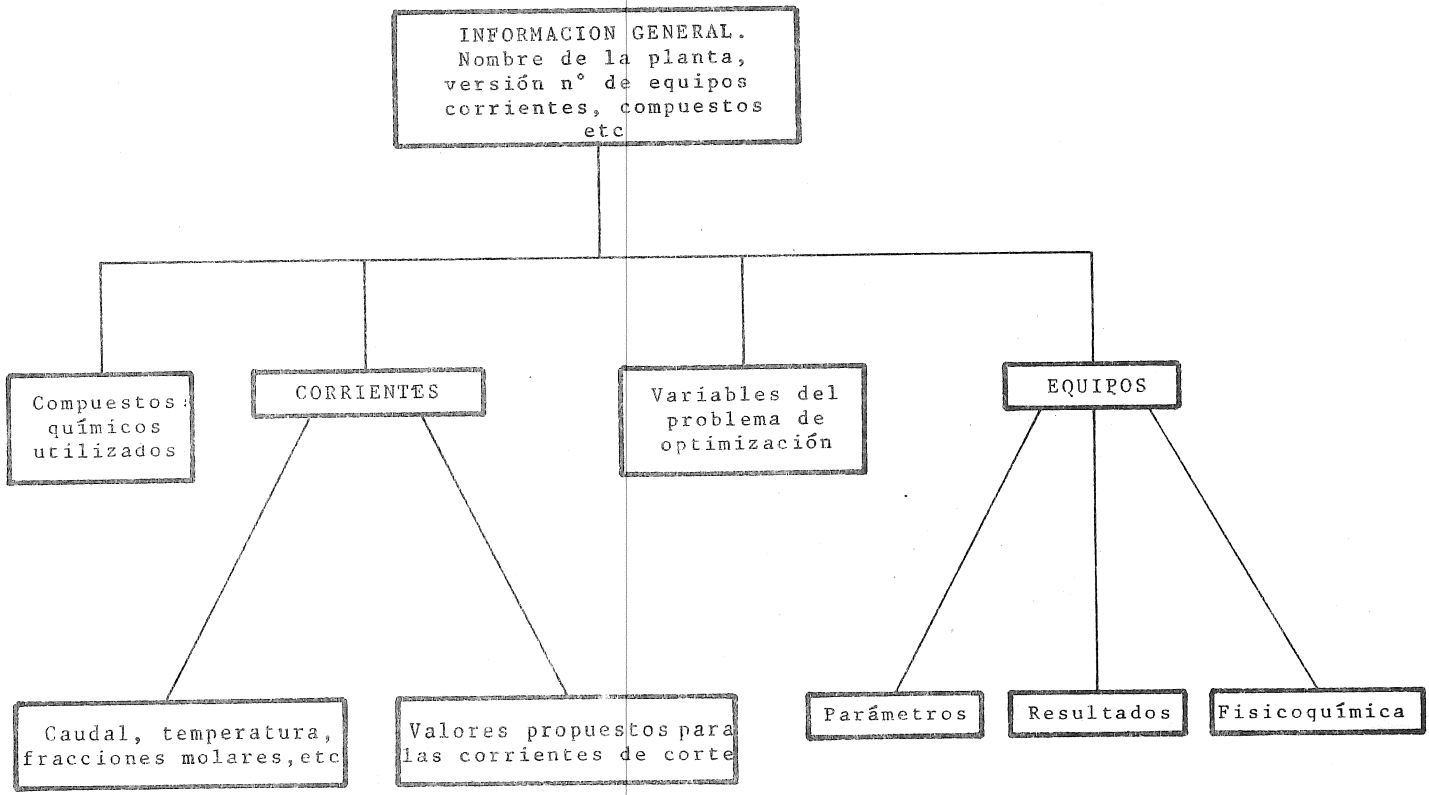


FIGURA 2.- Esquema General de la Base de Datos.

```
$ SET NOVERIFY
$ ON CONTROL_Y THEN GOTO JOUR
$ OPEN/READ/ERROR= P001 TRABAJO FLASH1.TRB
$             READ TRABAJO VAYA
$             CLOSE TRABAJO
$             DELE FLASH1.TRB;*
$             GOTO 'VAYA'
$ CORRE100:
$             CONTINUE
$ P001:
$         R [GSIMBAD.FINAL]NS1
$ P002:
$         R [GSIMBAD.FINAL]FV1N2
$ P003:
$         R [GSIMBAD.FINAL]ND1
$ P004:
$         R [GSIMBAD.FINAL]BV1N1
$ R [GSIMBAD.FINAL]CONVER
$ OPEN/READ ENTRADA FLASH1.CNV
$             READ ENTRADA KO
$             CLOSE ENTRADA
$ IF KO.EQS."A" THEN GOTO  CORRE100
$ DELETE FLASH1.CNV;*
$ R [GSIMBAD.FINAL]MASA
$ R [GSIMBAD.FINAL]ID
$ R [GSIMBAD.FINAL]IC
$ R [GSIMBAD.FINAL]IR
$ R [GSIMBAD.FINAL]DOC
$ EXIT
$ JOUR:
$ @SEGURO
$ EXIT
```

FIGURA 3.- Archivo de Operación para la simulación de la planta FLASH1 con el SIMBAD.

simulación. Luego, el AO tiene los comandos de ejecución de los distintos equipos ordenados según la secuencia de resolución. En este ejemplo se simulan un sumador, un flash, un divisor y una bomba con los programas NS1, FV1N2, ND1 y BV1N1, opciones seleccionadas en la carga de datos. Luego el programa CONVER, utilizando el método de sustitución directa o el de aceleración de Wegstein, efectúa la convergencia de la corriente de corte 2. Si no se ha alcanzado la convergencia, se retoma la ejecución de los equipos desde CORRE100 y, en caso contrario, se imprimen los datos y resultados con los programas ID, IC e IR.

Las ventajas de una estructura como la descripta son, principalmente, un eficiente uso de memoria central y de los recursos del sistema, y una estructura sumamente modular y flexible que permite la implementación de distintas estrategias de resolución sin modificar los elementos del simulador. Todos éstos interactúan a través de la base de datos, de la cual retiran la información para el cálculo, y donde depositan los resultados para que sean utilizados por las siguientes aplicaciones.

La introducción de la opción de optimización no ha implicado cambios en la estructura del SIMBAD, teniendo en cuenta las ventajas ya mencionadas. El único programa que se altera es el incluido en el APE, encargado de generar el AO. Este debe contemplar la participación del módulo optimizador y otros cambios menores. La figura 4 es el AO para la optimización de la planta FLASH1. En primer lugar se ha reemplazado el programa CONVER por el OPTIMIZA que se encarga de la aplicación del método PCS. Es preciso reiterar que, de acuerdo a (3), al mismo tiempo que optimiza, se convergen las corrientes de corte al incluir $h(x,y)=0$ entre las restricciones. Además se ha cambiado la posición del rótulo CORRE100 pues antes, para converger las corrientes de corte, siempre se calculaban todos los equipos desde el comienzo de la secuencia de resolución. Para optimizar muchas veces es necesario simular una parte de dicha secuencia a partir de un equipo intermedio, por lo que el dispositivo con el fichero de extensión. TRB es muy útil. Por ejemplo, si se desea calcular gradientes con respecto al coeficiente de partición del divisor III, luego de perturbar este valor en la base de datos, el programa OPTIMIZA carga en FLASH1.TRB la cadena P003, y sólo se corren para calcular dichos gradientes los equipos III y IV.

La figura 5 muestra esquemáticamente la operación del optimizador.

```
$ SET NOVERIFY
$ ON CONTROL_Y THEN GOTO JOUR
$ CORRE100:
$ OPEN/READ/ERROR= P001 TRABAJO FLASH1.TRB
$             READ TRABAJO VAYA
$             CLOSE TRABAJO
$             DELE FLASH1.TRB;*
$             GOTO 'VAYA'
$
$ P001:
$         R [GSIMBAD.FINAL]N61
$ P002:
$         R [GSIMBAD.FINAL]FV1N2
$ P003:
$         R [GSIMBAD.FINAL]ND1
$ P004:
$         R [GSIMBAD.FINAL]BV1N1
$ R [GSIMBAD.FINAL]OPTIMIZA
$ OPEN/READ ENTRADA FLASH1.OPT
$             READ ENTRADA KO
$             CLOSE ENTRADA
$ IF KO.EQS."A" THEN GOTO  CORRE100
$ DELETE FLASH1.OPT;*
$ R [GSIMBAD.FINAL]MASA
$ R [GSIMBAD.FINAL]ID
$ R [GSIMBAD.FINAL]IC
$ R [GSIMBAD.FINAL]IR
$ R [GSIMBAD.FINAL]DOC
$ EXIT
$ JOUR:
$ @SEGURO
$ EXIT
```

FIGURA 4.- Archivo de Operación para la optimización de la planta FLASH1 con el SIMBAD.

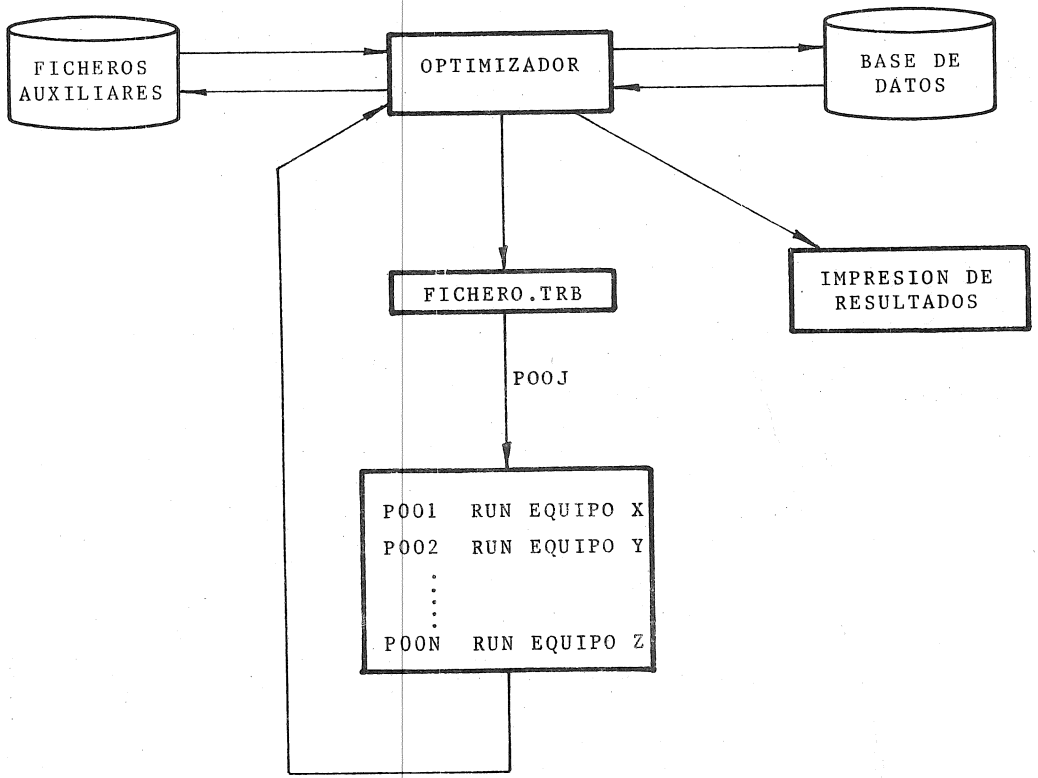


FIGURA 5.- Ejecución de la optimización.

Este interactúa con la base de datos, pero también cierta información la maneja a través de ficheros auxiliares por la volatilidad de esos valores. La relación con la secuencia de resolución se realiza en todos los casos (cálculo de gradientes, optimización univariable, etc.) a través del fichero de extensión .TRB.

La estructura implementada conserva la modularidad del SIMBAD. Además permite cambiar con facilidad distintas estrategias: camino factible, no factible o híbridos intermedios, incorporar por ejemplo módulos simplificados para el cálculo de los gradientes, etc.

6. Futuros desarrollos y conclusiones

El trabajo presentado describe la estructura básica del módulo optimizador del SIMBAD. Se trata de un esqueleto sobre el cual se irán incorporando distintas opciones en el futuro. Muchas de estas corresponden a variantes del método PCS y a su aplicación al caso concreto de procesos químicos. Existen un número de parámetros y factores que afectan la performance y eficiencia del método que es preciso determinar con la mayor exactitud posible, más si se tiene en cuenta que para este tipo de problemas una evaluación de la función objetivo y las restricciones en cierto punto demandan un considerable tiempo de CPU. Los elementos a analizar son:

- Cálculo de los gradientes: por perturbación directa (única opción disponible al momento), regla de la cadena, jacobiano analítico, módulos simplificados, etc.
- Optimización univariable: selección de la función de penalidad y los factores para cada restricción violada (al momento se usa lo planteado por Powell (1977)), criterio de terminación, etc.
- Inicialización del problema
- Escalado del problema
- Factor de perturbación
- Selección de las corrientes de corte

Además, cada uno de los puntos anteriores está incluido en una estrategia de resolución: camino factible, no factible e híbridos

intermedios. Todas las anteriores trabajan con el flujo de información para la resolución coincidiendo con el flujo de materia y energía en la planta. Si se altera para la resolución el flujo de información buscando aquel que favorezca la resolución, utilizando el planteo de Montagna e Iribarren (1988) se puede obtener una reducción en el tiempo de CPU requerido.

Se cuenta por lo tanto con la estructura del módulo optimizador que permitirá la evaluación de distintas alternativas de modo de obtener un eficiente algoritmo para la optimización de procesos químicos. Este es un tema en el cual se está trabajando activamente en distintos centros y que ha llevado a que ya hayan aparecido las primeras versiones de simuladores comerciales incluyendo versiones iniciales de optimización general de procesos.

REFERENCIAS

- L. Biegler, Improved infeasible path optimization for sequential modular simulators - I. The interface, Comp. and Chem. Engng., 9, 3, 245 (1985).

- L. Biegler y R. Hughes, Approximation programming of chemical processes with Q/LAP, Chem. Engng. Prog., 77, 4, 76 (1981).

- L. Biegler y R. Hughes, Feasible path optimization with sequential modular simulator, Comp. and Chem. Engng., 9, 4, 379 (1985).

- L. Biegler y R. Hughes, Infeasible path algorithms for sequential modular optimization, AIChE J., 28, 6, 994 (1982).

- J. Dennis y J. Morê, Quasi-Newton methods, motivation and theory, SIAM Rev., 19 (1), 46 (1977).

- T. Kesala, Successive quadratic programming in sequential modular process flowsheet simulation and optimization, Doctoral Thesis Digest, M.I.T., Cambridge, E.E.U.U. (1985).

- Y. Lang y L. Biegler, A Unified Algorithm for Flowsheet Optimization, EDRC-06-18-86, Engineering Design Research Center, Carnegie Mellon University, E.E.U.U. (1986).

- H. Leone, T. Melli, J. Montagna, A. Vecchietti y R. Cerro, SIMBAD: A steady state process simulator associated to a DBMS. III. The physico-chemical properties package, Comp. and Chem. Engng., 11, 3, 217 (1987).

- J. Montagna, H. Leone, T. Melli, A. Vecchietti y R. Cerro, SIMBAD: A steady state process simulator associated to a DBMS. I. The executive system, Comp. and Chem. Engng., 11, 63 (1987).

- J. Montagna y O. Iribarren, Optimal computation sequence in the simulation of chemical plants, Comp. and Chem. Engng., 12, 1, 71 (1988).

- M. Powell, A fast algorithm for nonlinearly constrained optimization calculations, trabajo presentado en la Dundee Conference on Numerical Analysis, Escocia (1977).

- A. Vecchietti, H. Leone, T. Melli, J. Montagna y R. Cerro, SIMBAD: A steady state process simulator associated to a DBMS. II. The structure of memory, Comp. and Chem. Engng., 11, 5, 519 (1987).

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

OPTIMIZACION DE SISTEMAS DINAMICOS VIA PERTUBACION SINGULAR DE LA ECUACION DE HAMILTON-JACOBI-BELLMAN

Laura Aragone

PROMAR- Instituto de Matemática Beppo Levi
Avenida Pellegrini 250
(T.E: 217998-TELEX: 41817 CIROSAR)
Universidad Nacional Rosario
(2000) Rosario - Argentina

Resumen: En este trabajo se presenta una metodología especial para resolver la ecuación de Hamilton-Jacobi-Bellman asociada a la función de costo óptimo de problemas de control de sistemas determinísticos en el caso estacionario. La misma consiste en utilizar una perturbación singular del operador diferencial interviniente en la ecuación de Hamilton-Jacobi-Bellman y esquemas de discretización especiales para el gradiente de la función de costo óptimo, basados en el método de las diferencias finitas.

Estos esquemas satisfacen un Principio de Máxima Discreto, lo que implica la estabilidad y convergencia del método, así como la posibilidad de utilizar algoritmos iterativos para resolver el problema discretizado.

I-Introducción

Se considera en este trabajo un problema de control óptimo de sistemas dinámicos descriptos por ecuaciones diferenciales ordinarias. Se estudia el caso estacionario, realizando discretizaciones orientadas a preservar para el problema aproximado, las características de linealidad en los controles del problema original; de esta forma las optimizaciones puntuales que aparecen en la ecuación de Hamilton-Jacobi-Bellman (H-J-B) resultan ser programaciones lineales, que al estar definidas sobre restricciones de tipo hipercubos pueden ser solucionadas directamente por inspección de los coeficientes que multiplican a los controles.

El sistema tratado tiene una evolución dinámica descripta por el sistema de ecuaciones diferenciales ordinarias:

$$\frac{dx}{dt} = f(x(t),y(t)) \qquad (1)$$

$$x(0) = x_0$$

siendo

$$f(x,y) = f_0(x) + \sum_{i=1}^{m} f_i(x)\ y_i$$

donde $x_0 \in \Omega$, el dominio $\Omega \subset \Re^n$ es acotado y se supone que para todo $t \geq 0$ el estado del sistema $x(t) \in \Omega$ (la trayectoria del sistema no sale nunca de Ω)

Las políticas de control:

$y(\cdot) : [0,\infty] \rightarrow Y$

siendo

$$Y = [\ y_{1,min}\ ,\ y_{1,max}\ ] \ldots [\ y_{m,min}\ ,\ y_{m,max}\ ]$$

son funciones medibles y se supone que:

$f_i(x)$ es una función continua y acotada de $x \in \Omega$ y que además se satisface:

$$\| f_i(x) - f_i(\tilde{x}) \| \leq C_{f_i} \| x - \tilde{x} \| \qquad (2)$$

$\forall\ x, \tilde{x} \in \Omega$

Bajo estas condiciones se demuestra (para cualquier control $y(\cdot)$ arbitrario) la existencia y unicidad de la solución de (1) [5].

A este sistema dinámico se le asocia el funcional de costo $J(x_0,y(\cdot))$. Si $y(\cdot)$ es una

política arbitraria de control, J toma la forma:

$$J(x_0,y(\cdot))=\int_0^{\infty} \left(l_0(x(t)) + \sum_{i=1}^{m} l_i(x(t))\, y_i(t)\right) e^{-\alpha t}\, dt \qquad (3)$$

El objetivo del problema es encontrar la política de control $\bar{y}(\cdot)$ que minimiza el funcional J, es decir tal que

$$J(x_0, \bar{y}(\cdot)) = \inf_{y(\cdot)} J(x_0,y(\cdot))$$

Los coeficientes $l_i(x) : \Omega \to \Re$, que intervienen en la función de costo instantáneo

$$l(x,y) = l_0(x(t)) + \sum_{i=1}^{m} l_i(x(t))\, y_i(t)$$

son funciones continuas, acotadas y satisfacen:

$$| l_i(x) - l_i(\tilde{x}) | \leq C_1 \| x - \tilde{x} \| \qquad (4)$$

para una constante $C_1 > 0$, $x, \tilde{x} \in \Omega$

α es el coeficiente de actualización y se supone

$$\alpha > \sum_{i=1}^{m} C_{f_i} \max\left(|y_{i,min}|, |y_{i,max}| \right)$$

II-Definición de la función de costo óptimo

La función de costo óptimo se define

$$V(x_0) = \inf_{y(\cdot)} J(x_0, y(\cdot)) \qquad (5)$$

(ver [5] pág 9).

Bajo las condiciones dadas para f y l (ver [5] pág. 38) se demuestra la existencia de la función costo óptimo y la lipschitzianidad de la misma. Mas aún, debido a la forma especial (lineal en y) de la dinámica (1) y del costo J, existe una política óptima ya que:

a) J es continuo en la topología $*$-débil de $L^{\infty}((0,\infty); \Re^m)$

b) el conjunto de las políticas admisibles es compacto bajo esta topología en virtud de la convexidad de Y [7].

Asímismo se prueba que V verifica la ecuación de H-J-B asociada:

$$\min_{y \in Y} \; LV=0 \qquad (6)$$

siendo

$$LV = \nabla V \, f + 1 - \alpha \, V \geq 0$$

V es llamada la función de costo óptimo y en base a su conocimiento pueden determinarse las políticas óptimas de control buscadas, por lo que la solución del problema original se reduce esencialmente al cálculo de esta función V.
Para encontrar numéricamente la solución V de (6), se resuelve esta ecuación de H-J-B en forma aproximada por discretización, hallando una función V^h que converge hacia V cuando h (parámetro de discretización) tiende a cero.

III-Perturbación singular de la ecuación de H-J-B

Con vistas a obtener esquemas de discretización convenientes se procede a considerar una perturbación singular del operador diferencial interviniente en (6). La ecuación perturbada tomará la forma:

$$\min_{y \in Y} \; L_\sigma(V^\sigma)=0 \qquad (7)$$

donde

$$L_\sigma(V^\sigma) = A_\sigma V^\sigma + \nabla V^\sigma + 1 - \alpha V^\sigma$$

con

$$A_\sigma = \sum_{k=1}^{n} \sigma_k \frac{\partial^2}{\partial x_k^2}$$

Este operador resulta de analizar, en lugar del sistema determinístico asociado a la ecuación (1), el sistema con perturbaciones aleatorias descripto por la ecuación estocástica de Itô.

$$dx = f(t) \, dt + \sum_{k=1}^{n} \sqrt{2\sigma_k} \, dw_k \qquad (8)$$

donde los w_k son movimientos brownianos unidimensionales independientes [6].
Se considera que la difusión descripta por (8) se comporta como una difusión

reflejada en la frontera de Ω; en consecuencia la ecuación (7) tendrá como condición de frontera natural la siguiente restricción

$$\sum_{k=1}^{n} \sigma_k \frac{\partial v^\sigma}{\partial x_k} n_k = 0 \qquad (9)$$

n_k componente k-ésima del versor normal a $\partial\Omega$

se tiene así que

$$v^\sigma \in W = \left\{ w \in W^{2,\infty} \ / \text{ verifica (9)} \right\}$$

Puede probarse que cuando $\sigma \to 0$ $v^\sigma \to v$ por lo que en los siguientes párrafos será desarrollada una metodología para aproximar v^σ.

La ecuación perturbada de H-J-B puede ser obtenida también directamente aplicando la teoría de perturbaciones por regularización de ecuaciones e inecuaciones variacionales (ver [11]), sin necesidad de recurrir a interpretaciones probabilísticas de las mismas.

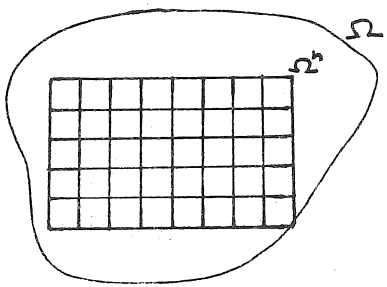
Asimismo, los esquemas obtenidos a través de la perturbación estocástica del sistema pueden ser logrados también como caso límite (pasando al caso estacionario) de los esquemas obtenidos para problemas determinísticos evolutivos en el trabajo [9].

IV-Formulación del problema discretizado

a)Discretización de Ω:

El conjunto Ω es reemplazado por un cubrimiento Ω^h formado por nodos $x^h_{p=x_p}$ (ver figura 1) que determinan un enrejado uniforme.

$h = \max(\Delta x_1, \Delta x_2)$, es llamada la norma de la partición.



b)Discretización del conjunto de controles Y:

El conjunto Y es reemplazado por el conjunto aproximado $Y^h = I_1^h \ldots I_n^h$ siendo

$$I_j^h = \left\{ \xi_j^k \;/\; \xi_j^k = y_{j,min} + \left(y_{j,max} - y_{j,min} \right) \frac{k}{n_j} \quad , \quad k = 0,1,\ldots, n_j \right\}$$

c)Discretización de W:

Las funciones de W son reemplazadas por funciones w^h definidas en los nodos del enrejado Ω^h

d) Discretización de las condiciones de frontera de la ecuación (7).

La condición de frontera $\frac{\partial w}{\partial n}=0$ se discretiza en forma natural: (para el caso de Ω rectángulo)

$$\begin{aligned}
&w^h(0,x_2^h) = w^h(\Delta x_1, x_2^h) && \forall\ x_2^h \\
&w^h(L_1,x_2^h) = w^h(L_1-\Delta x_1, x_2^h) && \forall\ x_2^h \\
&w^h(x_1^h, 0\ ) = w^h(x_1^h, \Delta x_2) && \forall\ x_1^h \\
&w^h(x_1^h, L_2) = w^h(x_1^h, L_2-\Delta x_2) && \forall\ x_1^h
\end{aligned}$$

y en forma similar para el caso general

La ecuación (7) se discretiza de la siguiente forma

$$\min_{y \in Y^h} L_0^h\, \overline{V}^h = 0 \qquad (10)$$

$$L_0^h\, w^h = A_0^h\ w^h + \nabla^h w^h \cdot f + 1 - \alpha w^h$$

donde las discretizaciones Δ^h y ∇^h del laplaciano y del gradiente están definidas respectivamente por:

$$A_0^h\ w^h = \sum_{k=1}^{n} \sigma_k \frac{w^h(x+e_k\Delta x_k) + w^h(x-e_k\Delta x_k) - 2w^h(x)}{(\Delta x_k)^2}$$

$$\nabla w^h \cdot f = \nabla w^h \cdot f_0 + \sum_{i=1}^{m} \nabla w^h \cdot f_i\ y_i$$

$$\nabla w^h \cdot f_i = \sum_{k=1}^{n} \frac{w^h(x+e_k\Delta x_k) - w^h(x-e_k\Delta x_k)}{2\,\Delta x_k} f_{ik}(x)$$

donde

$f_i = (f_{i1}, \ldots, f_{in})$ $\quad i=0,1,\ldots,m$

y

$e_k = (0,\ldots,1,\ldots,0)$

es el k-ésimo versor unitario.

Esta forma de discretizar el operador L_σ permite reescribir la ecuación (10) en la forma equivalente:

$$\bar{V}^h = P_h \bar{V}^h$$

siendo:

$$P_h w^h(x_1,x_2) = \frac{1}{\left(2\sum_{k=1}^{n}\sigma_{k1}(\Delta x_k)^{-2} + \alpha\right)} \left\{ \sum_{k=1}^{n} \sigma_k \frac{w^h(x+e_k\Delta x_k) + w^h(x-e_k\Delta x_k)}{(\Delta x_k)^2} + \right.$$

$$+ \sum_{k=0}^{n} \frac{w^h(x+e_k\Delta x_k) - w^h(x-e_k\Delta x_k)}{2\,\Delta x_k} f_{ok}(x) + l_o(x) +$$

$$\left. + \min_{y\in V^h} \left(\sum_{k=1}^{n} \left\{ \frac{w^h(x+e_k\Delta x_k) - w^h(x-e_k\Delta x_k)}{2\,\Delta x_k} f_{ik}(x) + l_k(x) \right\} y_k \right) \right\} \qquad (11)$$

Cuando

$$\sigma_k \geq \left(\| f_{ok} \| + \sum_{i=1}^{m} \| f_{i,k} \| \right) \frac{\Delta x_k}{2} \qquad (12)$$

se puede demostrar fácilmente que L_σ^h verifica un principio de máximo discreto (ver [8]) y que en consecuencia el operador P_h es contractivo. Por lo tanto existe un único punto fijo $\bar{V}^h$del operador P_h , que es la única solución de la ecuación discretizada (10).

V) Caracterización de la solución del problema discretizado

Podemos entonces (bajo la condición (12)) aplicar el teorema del punto fijo [9] y obtenemos:

Existe un único punto fijo $\bar{V}^h$ del operador P_h que es la única solución del sistema (10).

Queda así caracterizada la solución del problema discretizado y al mismo tiempo el

operador P_h permite definir un algoritmo para calcular $\overline{V}^h$.

VI) Solución del poblema discretizado

La demostración del teorema del punto fijo nos sugiere el siguiente algoritmo para computar $\overline{V}^h$.

ALGORITMO

Paso 0 : Tomar $V_0 \in W^h$, hacer $\nu=0$

Paso 1 : Hacer $V_{\nu+1} = P_h V_\nu$

Paso 2 : Si es $V_\nu = V_{\nu+1}$ ir a 3, sino, hacer $\nu=\nu+1$ e ir a 1.

Paso 3 : Hacer $\overline{V}^h = V_\nu$ y parar.

Este algoritmo termina en un número finito de pasos o genera una sucesión de elementos V_ν que converge a $\overline{V}^h$, verificándose la siguiente desigualdad:

$$\| V_\nu - \overline{V}^h \| \leq (\tilde{\beta})^\nu \| V_0 - \overline{V}^h \| , \quad \tilde{\beta} < 1$$

VII) Convergencia de las soluciones aproximadas

La solución $\overline{V}^h$ del problema discretizado converge uniformemente hacia la función V (costo óptimo), solución de (6).

En efecto por deformación de dominio, utilizando las técnicas de regularización y discretización empleadas en [8] se obtiene que $\overline{V}^h \underset{h\to 0}{\to} V$. Pudiéndose demostrar además que

$$\| \overline{V}^h - V \| \leq C\sqrt{h}$$

Los detalles completos de estos resultados están contenidos en [2].

VIII) Ejemplo de aplicación

$\Omega = [0, 1] \times [0, 1]$

$x(0)= x_0 \in \Omega$

Ecuación dinámica del sistema

$f_{0k}= 0 \qquad k=1,2$

$$f_{ik}= -\beta_i \text{ sat } \left(\frac{\sqrt{x_1^2+x_2^2}- 0.5}{\lambda} \right) \frac{x_k}{\sqrt{x_1^2+x_2^2}} \qquad k=1,2 \qquad i=1,2$$

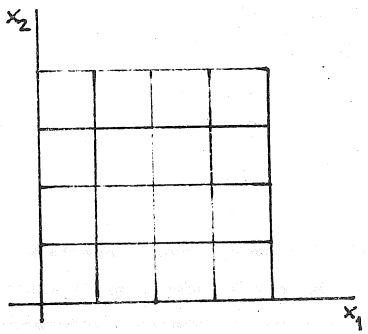
$g_i(\cdot) : [0,\infty) \to [0, 1] \qquad i=1,2$

Costo instantáneo:

$$l_C(t) = \gamma \left(\sqrt{x_1^2 + x_2^2} - 0.5 \right)^2$$

$$l_i = c_i \, y_i$$

Gráfico de discretización de Ω:



Analíticamente se obtiene que las políticas de control pueden calcularse exactamente en función de los valores de γ, α, β_1, β_2, c_1 y c_2, resultando los controles óptimos constantes en las siguientes zonas

$y_1=0$ si $| \rho - 0.5 | < \omega_1$

$y_2=0$ si $| \rho - 0.5 | < \omega_2$

$y_1=1$ si $| \rho - 0.5 | \geq \omega_1$

$y_2=1$ si $| \rho - 0.5 | \geq \omega_2$

donde

$$\rho = \sqrt{x_1^2 + x_2^2}$$

$$\omega_1 = \frac{c_1 \alpha}{2\beta_1 \gamma} \quad , \quad \omega_2 = \frac{c_2 \alpha}{2\beta_2 \gamma} + \frac{\beta_1}{\alpha}$$

En las siguientes gráficas se muestran algunos resultados numéricos obtenidos con esta metodología, dando las zonas de aplicación (y=1 , z=1) de los controles óptimos y la correspondiente evolución del sistema bajo estas estrategias.

Las constantes empleadas son:

$\lambda = 0.01$

$c_1 = 0.03$

$c_2 = 0.045$

$\beta_1 = 0.05$

$\beta_2 = 0.05$

$\alpha = 10$

$\gamma = 20$

Gráficas 1 a 4.

Conclusiones:

En este trabajo se ha logrado una extensión de la metodología presentada en [8] al caso de control óptimo continuo. Lo esencial del procedimiento es la introducción de una perturbación singular que permite discretizar el operador diferencial L en la forma indicada en (11). Esta forma especial de discretización implica que el operador P_h puede escribirse en forma tal que la operación de búsqueda puntual de mínimo en la ecuación (10) resulta una programación lineal sobre un hipercubo por lo que este último problema puede ser solucionado directamente por inspección de los coeficientes que multiplican (en (6)) a los controles y_i.

Pueden darse sin ninguna dificultad condiciones suficientes sobre los coeficientes de perturbación para que sea válida la característica de contracción de P_h.

En base a esta propiedad se obtiene no sólo que el problema discretizado tiene solución única, sino también la estructura iterativa del algoritmo de cómputo y su convergencia. El procedimiento tiene la propiedad de convergencia uniforme de las aproximaciones y los ejemplos numéricos (testeados sobre un problema con solución exacta conocida) muestran la factibilidad y eficiencia del mismo.

Referencias

[1] Aragone, L. S. & González, R. L. & Tidball, M. M. "Teoría de Juegos Diferenciales con Controles Monótonos". Aceptado para ser presentado en el Séptimo Congreso Brasilero de Automática. 15- 19 Agosto 1988 São José Dos Campos- Brasil

[2] Aragone, L. S. & González, R. L. & Tidball, M. M. "Solución numérica de Juegos Diferenciales" Rapport de Recherche INRIA. En preparación.

[3] Barron, E. N. & Jensen, R., (1984) "A nonlinear evolution system with two subdifferentials and monotone differential games", Journal of Differential Equations. Vol 97 N^0 1.

[4] Ciarlet, P.G. & Raviart, P.A. "Maximum principle and uniform convergence for the finite element method", Computer Methods in Applied Mechanics and Engineering, 2 (1973), 17-31

[5] Friedman, A., (1971). "Differential Games", Wiley- Interscience, New York.

[6] Friedman, A., (1975). "Stochastic Differential Equations and Applications", Vol. 1, Academic Press, New York. Vol 2, 1976.

[7] Ekeland, I & Teman, R, (1974) " Analyse convexe et problèmes variationnels" , Dunod, Paris.

[8] González, R. L. & Rofman, E. (1985) "On deterministic control problem: an aproximation procedure for the optimal cost", Parts I and II, SIAM J. Control and Opt., 23, 2 (1985) 242-266 and 267-285.

[9] González, R. L. & Rofman, E. "Computational methods in scheduling optimization", Fermat Days 85: Mathematics for Optimization 135-155

[10] Protter, M. H., (1977) "A first course in real analysis", Springer Verlag, New York.

[11]Lions, J.L. & Stampacchia, G. (1967) "Variational inequalities" , Comm. Pure Applied Math., 20, 493-519.

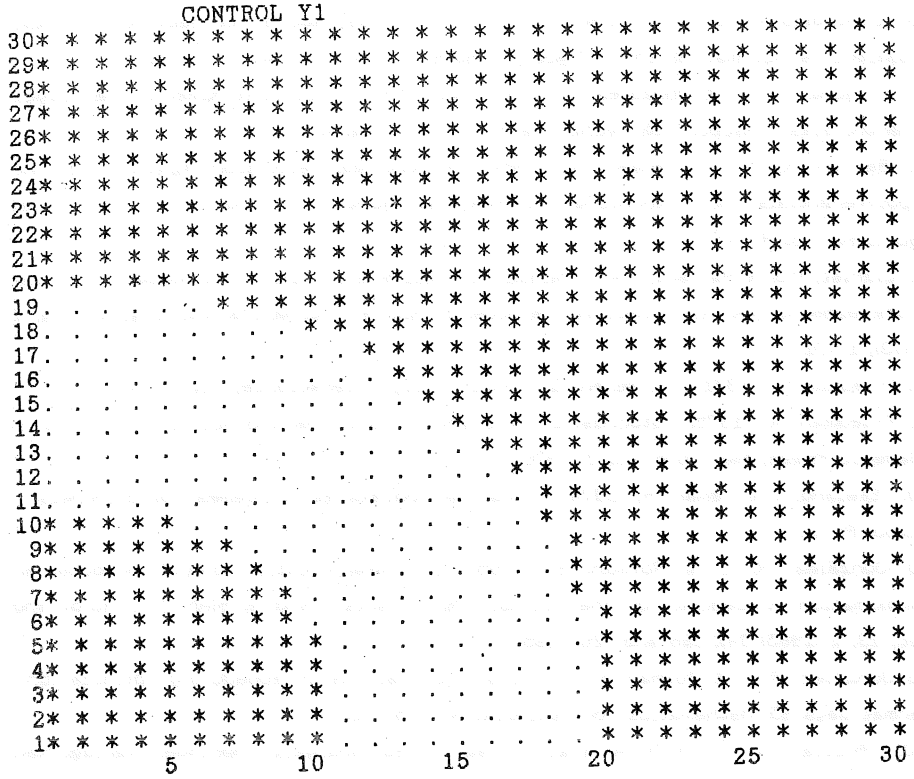

CONTROL Y1
30*
29*
28*
27*
26*
25*
24*
23*
22*
21*
20*
19.
18.
17.
16.
15.
14.
13.
12.
11.
10*
9*
8*
7*
6*
5*
4*
3*
2*
1*
5
10
15
20
25
30

Grafico I

CONTROL Y2

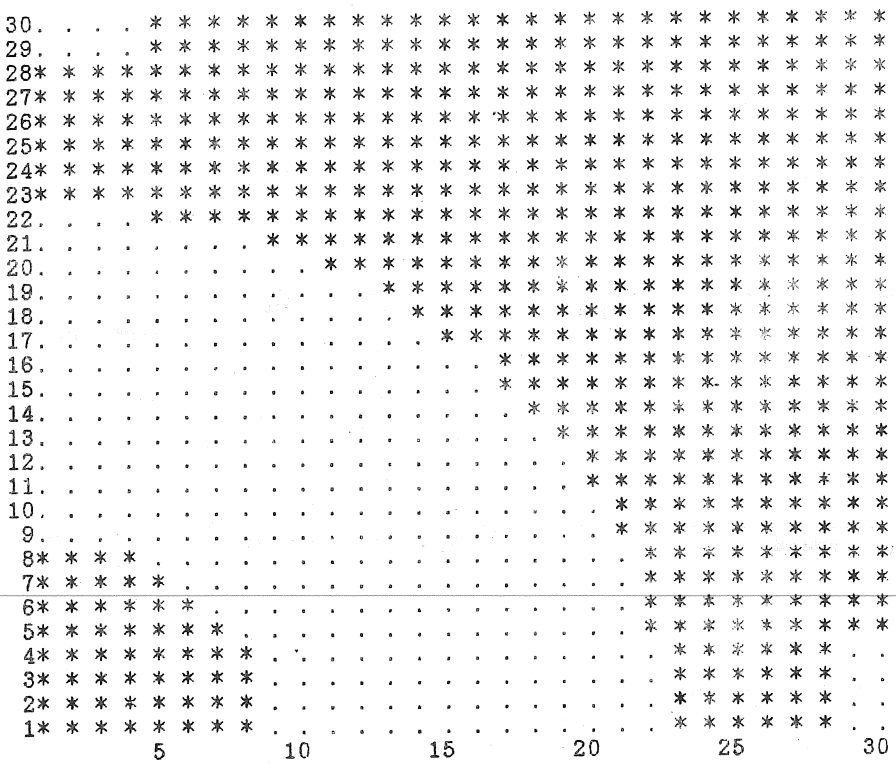


Grafico II

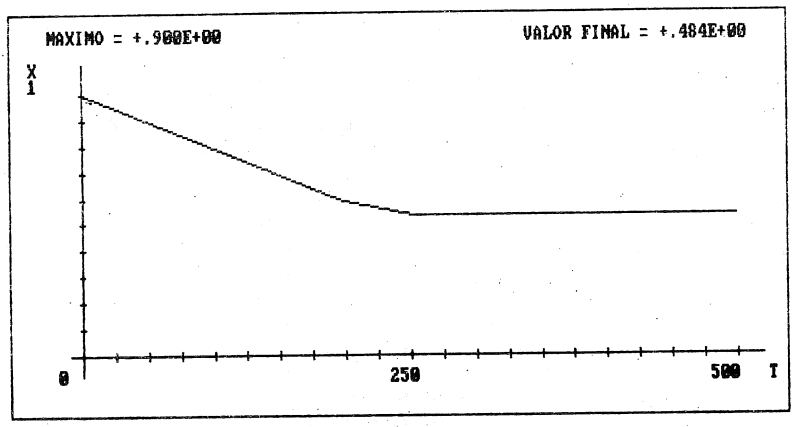


Gráfico III

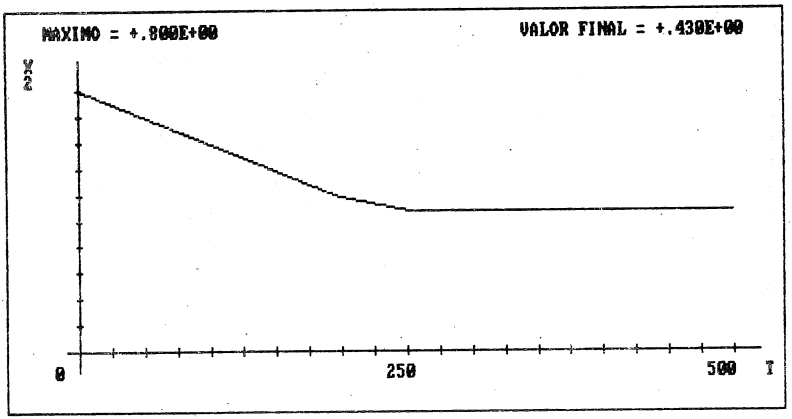


Gráfico IV

XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa.
Buenos Aires, Setiembre 1988.

PROBLEMAS DE OPTIMIZACION DE SISTEMAS ECONOMICOS CON CRITERIOS CUADRATICOS Y CONTROLES MONOTONOS UN ALGORITMO PARA SU SOLUCION NUMERICA

Laura Aragone

PROMAR- Instituto de Matemática Beppo Levi
Avenida Pellegrini 250 (T.E 217998-TELEX: 41817 CIROSAR)
Universidad Nacional Rosario
(2000) Rosario - Argentina

Resumen: En este trabajo se presenta un algoritmo numérico para resolver la ecuación de Hamilton-Jacobi-Bellman asociada a la función de costo óptimo de problemas de control de sistemas económicos que involucran la explotación de recursos no renovables. También estos problemas aparecen en el control de sistemas monetarios cuando sólo se considera en el mismo la inyección de recursos. En ambos casos los controles utilizados son monótonos (no crecientes o no decrecientes), pudiendo ser incluso discontinuos. Un estudio casi completo de este problema y de la caracterización teórica de su solución ha sido realizado en [1]. En ese artículo la solución del problema queda reducida al tratamiento de una inecuación cuasivariacional elíptica en el intervalo [0,T]. En este trabajo se realiza una extensión de la metodología presentada en [7] al análisis y solución numérica de esta inecuación, dando un algoritmo de resolución, propiedades de convergencia del procedimiento de discretización y los resultados computacionales obtenidos al resolver con este método un ejemplo presentado en [1].

I-Introducción

Se considera un problema de control óptimo de sistemas dinámicos descriptos por ecuaciones diferenciales ordinarias. El sistema es unidimensional y el intervalo de tiempo en consideración es finito; el funcional a optimizar es cuadrático y toma la siguiente forma explícita

$$\min_{p(\cdot)} \int_0^T e^{-\alpha s} \left\{ h\,[y(s)-f(s)]^2 + k\,[p(s)-f'(s)]^2 \right\} ds \qquad (1)$$

siendo

$$\frac{dy}{dt} = p(t) \qquad (2)$$

p es la variable de control y pertenece a $\mathfrak{R}^+ = [0,\infty)$

El problema de minimización estudiado es con condiciones iniciales libres, por lo que realmente el mínimo debe buscarse con respecto a y(0), p(·)

Se tiene en consecuencia que y es una variable no decreciente, de allí el nombre de controles monótonos que reciben este tipo de problemas, ya que puede considerarse a y como la variable de control.

α es el coeficiente de actualización, h y k son coeficientes de ponderación y f es una trayectoria deseable a perseguir.

Este tipo de problemas aparecen en cuestiones económicas como el control de la política monetaria, donde p representa la inyección de circulante e y es el stock monetario. Problemas similares tienen lugar cuando se considera el control de la explotación de recursos no renovables, como yacimientos de petróleo o minerales, bosques, política pesquera, etc.

En [1] se ha presentado la caracterización de la solución de este problema utilizando la metodología de inecuaciones variacionales en espacios de Sobolev.

Definiendo

$$Y(t) = \int_0^t y(s)\,ds \qquad\qquad F(t) = \int_0^t f(s)\,ds \qquad (3)$$

se obtiene la siguiente ecuación diferencial de optimalidad para Y

$$Y'' + (\,Y - F + F''\,)^- = 0 \qquad \text{en } (\,0,T\,) \qquad (4)$$

con condiciones de frontera

$$Y(0) = 0 \tag{5}$$

$$Y(T) = F(T) \tag{6}$$

Estas condiciones pueden ser obtenidas también directamente aplicando el principio de máximo de Pontriaguin. En particular (6) es consecuencia directa de considerar la optimización bajo condiciones iniciales libres para $y(0)$

La solución deseada de (4) posee la siguiente propiedad de minimalidad:

Teorema:

$$Y \leq u \text{ en } [0,T], \quad \forall\, u \in U \tag{7}$$

siendo

$$U = \left\{ u \in H^2[0,T] \,/\, u'' \leq 0, \;\; u'' - u \leq F'' - F, \;\; u(0) \geq 0, \;\; u(T) \geq F(T) \right\} \tag{8}$$

La demostración está contenida en [1]

El conjunto U es llamado el conjunto de supersoluciones. En base a la caracterización dada por el teorema para el elemento Y, el problema original queda reducido a determinar el elemento mínimo de U. Para hallarlo numéricamente se desarrolla un procedimiento de discretización aplicando la metodología presentada en [7]. El método está basado en el empleo de elementos finitos lineales y en el uso de discretizaciones débiles de las condiciones que intervienen en (8)

II El problema discretizado y sus soluciones

Se realiza una aproximación externa del espacio $H^2[0,T]$ utilizando elementos finitos lineales (elementos de $H^1[0,T]$), (llamamos W^h al conjunto de estos elementos) En consecuencia estas funciones aproximantes quedan determinadas por los valores que asume en los nodos de la discretización. En este caso unidimensional estos valores serán (para una partición uniforme de $[0,T]$) :

$$u^h(k\Delta) \qquad k=0, \ldots, n_h \qquad \Delta = \frac{T}{n_h} \tag{9}$$

La discretización de las restricciones que definen U determinan un conjunto de supersoluciones discretas U^h que tiene la siguiente forma:

$$U^h = \left\{ u^h \in W^h \;\; / \;\; u^h \text{ verifica (11)-(14)} \right\} \tag{10}$$

$$u^h(0) \geq 0 \tag{11}$$

$$u^h(n_h\Delta) \geq F(T) \tag{12}$$

$$u^h(k\Delta) \geq \frac{u^h((k+1)\Delta) + u^h((k-1)\Delta)}{2} \qquad k=1,\ldots,n_h-1 \tag{13}$$

$$u^h(k\Delta) \geq \left(\frac{u^h((k+1)\Delta) + u^h((k-1)\Delta)}{\Delta^2} + F(k\Delta) - F''(k\Delta) \right) \frac{\Delta^2}{2+\Delta^2} \qquad k=1,\ldots,n_h-1 \tag{14}$$

La función Y^h, elemento mínimo de U^h está caracterizado por las condiciones:

$$Y^h(0) = 0 \tag{15}$$

$$Y^h(n_h\Delta) = F(T) \tag{16}$$

$$Y^h(k\Delta) = \max\left\{ \frac{Y^h((k+1)\Delta) + Y^h((k-1)\Delta)}{2}, \left(\frac{Y^h((k+1)\Delta) + Y^h((k-1)\Delta)}{\Delta^2} + F(k\Delta) - F''(k\Delta) \right) \frac{\Delta^2}{2+\Delta^2} \right\}$$

$$k=1,\ldots,n_h-1 \tag{17}$$

Se introduce el operador $P: W^h \to W^h$ de la siguiente forma

$$(PW^h)(0) = 0 \tag{18}$$

$$(PW^h)(n_h\Delta) = F(T) \tag{19}$$

$$(PW^h)(k\Delta) = \max\left\{ \frac{W^h((k+1)\Delta) + W^h((k-1)\Delta)}{2}, \left(\frac{W^h((k+1)\Delta) + W^h((k-1)\Delta)}{\Delta^2} + F(k\Delta) - F''(k\Delta) \right) \frac{\Delta^2}{2+\Delta^2} \right\}$$

(20)

Este operador es monótono, y deja invariante al conjunto U^h. También puede demostrarse que si bien P no es contractivo , si lo es la potencia n_h del mismo.

Llamando $\tilde{Y}^h$ al único punto fijo de P^{n_h} es decir llamando $\tilde{Y}^h$ a la única solución de la ecuación:

$$\tilde{Y}^h = P^{n_h} \tilde{Y}^h \tag{21}$$

resulta tambien $\tilde{Y}^h = Y^h$ y por lo tanto tambien solucion de la ecuacion

$$Y^h = P\ Y^h \qquad (22)$$

El elemento mínimo de U^h es entonces la solución de la ecuación (22). Queda así caracterizada la solución del problema discretizado y al mismo tiempo el operador P_h permite definir un algoritmo para calcular Y^h.

III) Solución del poblema discretizado

La demostración del teorema del punto fijo nos sugiere el siguiente algoritmo para computar Y^h.

ALGORITMO

Paso 0 : Tomar $Y_0 \in W^h$, hacer $\nu=0$

Paso 1 : Hacer $Y_{\nu+1} = P_h\ Y_\nu$

Paso 2 : Si es $Y_\nu = Y_{\nu+1}$ ir a 3, sino, hacer $\nu=\nu+1$ e ir a 1.

Paso 3 : Hacer $Y^h = Y_\nu$ y parar.

Este algoritmo termina en un número finito de pasos o genera una sucesión de elementos Y_ν que converge a Y^h, verificándose la siguiente desigualdad:

$$\| Y_\nu - Y^h \| \leq (\tilde{\beta})^{\tilde{\nu}} \| Y_0 - Y^h \| \qquad , \ \tilde{\beta} < 1$$

siendo $\tilde{\nu} = \text{int}(\frac{\nu}{n_h})$

Puede observarse también, en virtud de las propiedades de P_h, que si se elige Y_0 perteneciente a U^h, la sucesión Y_ν verifica:

$$Y_\nu \in U^h$$

$$Y_\nu \geq Y_{\nu+1} \geq Y^h$$

iV) Convergencia de las soluciones aproximadas

La solución Y^h del problema discretizado converge uniformemente hacia la función Y solución de (4) cuando $n_h \to \infty$ ($\Delta \to 0$).

En efecto y en virtud del tipo de discretización empleada, se obtiene que es válido un principio de máximo discreto lo que implica la convergencia uniforme de las aproximaciones (ver [4], [6]). Pudiéndose demostrar además que

$\| Y^h - Y \| \leq C\, h$

V) Ejemplo de aplicación

Se considera uno de los problemas dados en [1] donde:

$T = \pi$

$f(t) = \text{sen}(t)$

Luego

$F(t) = 1 - \cos(t)$

La solución numérica se calculó tomando una partición del $[0,\pi]$ con $n_H = 100$

La solución exacta es

$$Y(t) = \begin{cases} ct & \text{si } t \in (0,x) \\ c_1 \left(e^{t} - e^{2\pi - t} \right) + 1 - \cos(t) & \text{si } t \in (x,\pi) \end{cases}$$

siendo

$x = 1.753508...$ el punto de conmutación de la política óptima ($p=0$ en $(0,x)$)

$c = 0.7775233...$

$c_1 = -0.0020895877...$

El error máximo obtenido entre la solución exacta Y y la numérica Y^h es $7.37\ 10^{-2}$

En la gráfica I se muestra la solución numérica Y^h del problema y en la gráfica II el error $Y - Y^h$

Conclusiones:

En este trabajo, siguiendo la sugerencia expuesta en [1], pag. 305 se ha realizado una aplicación de la metodología establecida en [8] al problema de control óptimo continuo con criterios cuadráticos y controles monótonos. El método está basado en la aproximación externa del espacio de Sobolev $H^2[0,T]$ por medio de elementos finitos lineales y discretizaciones que satisfacen un P.M.D. , constituyendo así una extensión del procedimiento presentado en [5], [7], para solucionar problemas de control óptimo estocásticos modelizados con difusiones continuas.

El procedimiento es absolutamente convergente, obteniéndose además convergencia

monótona en el caso de elegir como punto inicial para el algoritmo iterativo una supersolución discreta Y^h_0. En virtud de la regularidad especial $(C^{2,\alpha})$ de la solución exacta Y se demuestra una estimación del error de discretización del tipo:

$$\| Y^h - Y \| \leq C h^2$$

Referencias

[1] Barron, E. N. & Jensen, R. & Mallianis, A. G. "Minimizing a quadratic payoff with monotone controls" (1986) Mathematics of Operations Research Vol 12, N° 2, May 1987.

[2] Aragone, L. S. & Gonzalez, R. L. & Tidball, M. M. "Teoria de Juegos Diferenciales con Controles Monótonos". Aceptado para ser presentado en el Séptimo Congreso Brasilero de Automática. 15- 19 Agosto 1988 Saó José Dos Campos- Brasil

[3] Aragone, L. S. & González, R. L. & Tidball, M. M. "Solución numérica de Juegos Diferenciales " Rapport de Recherche INRIA. En preparación.

[4] Ciarlet, P.G. & Raviart, P.A. "Maximum principle and uniform convergence for the finite element method", Computer Methods in Applied Mechanics and Engineering, 2 (1973), 17-31

[5] Gonzalez, R. L. & Medina, M. A.(1987) "Sobre la solucion numérica de un problema de control óptimo de una difusión bidimensional" Anales del 5° Encuentro Nacional de Investigadores y Usuarios del Método de Elementos Finitos (Por aparecer en Mecánica Computacional Vol. VI, EUDEBA)

[6] Ciarlet, P. G. (1969) "Discrete maximum principle for finite-difference operators" Aequationes Mathematicae Vol 4 fasc 3, 1970, pág 338-352.

[7] González, R. L. & Rofman, E. (1987) "Stochastics control problems: an algorithm for the value function and the optimal policy. Some applications." Proceedings of 13th IFIP Conference on Systems Modelling and Optimization, Tokio, Aug.31-Sept 4

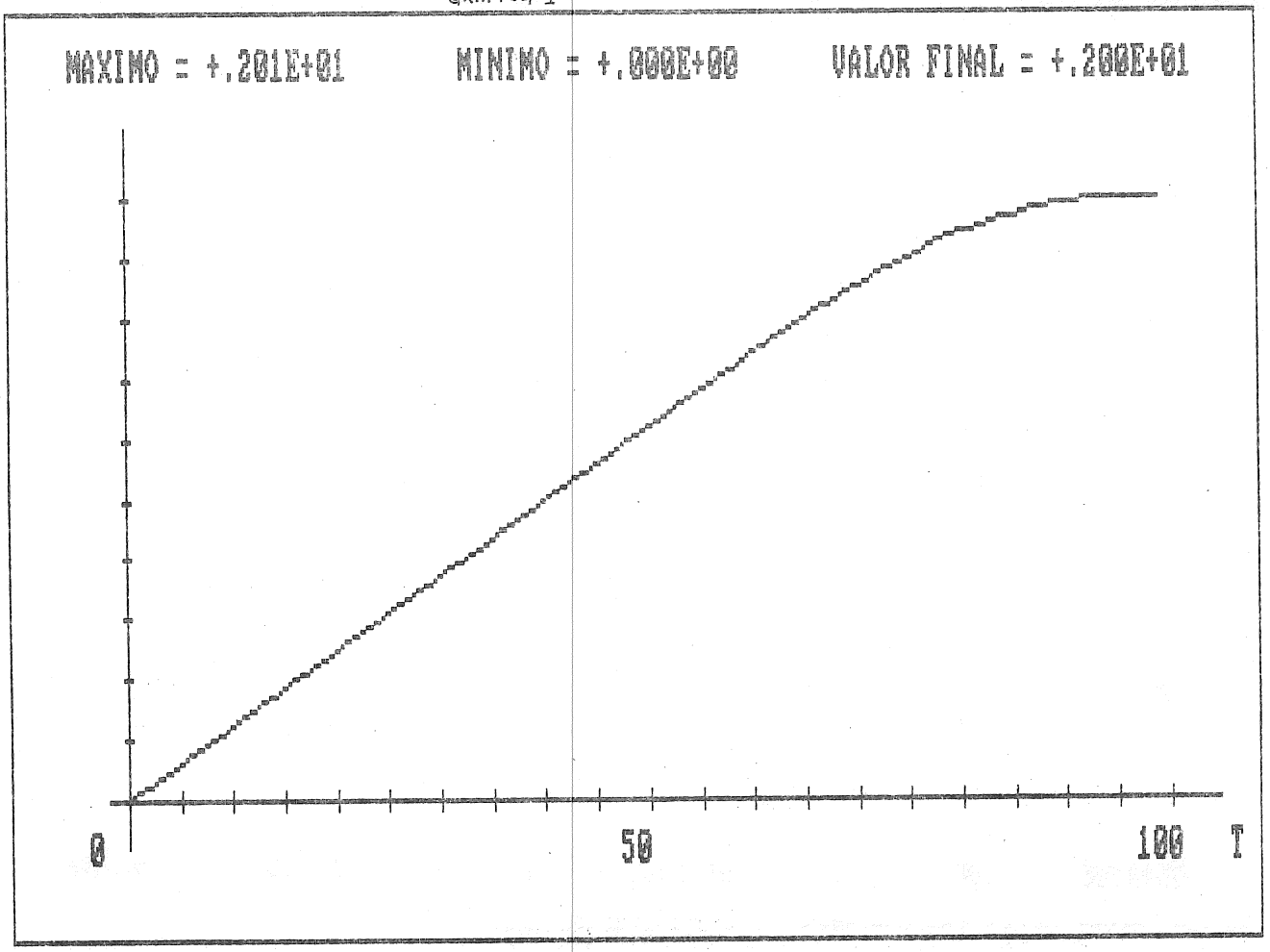
[8] González, R. L. & Rofman, E. (1985) "On deterministic control problem: an aproximation procedure for the optimal cost", Parts I and II, SIAM J. Control and Opt., 23, 2 (1985) 242-266 and 267-285.

[9] González, R. L. & Rofman, E. "Computational methods in scheduling optimization", Fermat Days 85: Mathematics for Optimization 135-155

[10] Protter, M. H., (1977) "A first course in real analysis", Springer Verlag, New York.

[11]Lions, J.L. & Stampacchia, G. (1967) "Variational inecualities" , Comm. Pure Applied Math., 20, 493-519.

GRAFICA I



GRAFICA II

