ESPECIFICACION FORMAL DE PROTOCOLOS DE COMUNICACION.

P. O'CALLAGHAN*, C. RODRIGUEZ**

RESUMEN

En este trabajo, se presenta el problema de la especificación formal de protocolos de comunicación. Se muestra que los métodos clásicos de especificación son inadecuados para este fin, debido en gran parte a la presencia del tiempo como elemento clave en la descripción de un protocolo. Se propone la utilización de un nuevo formalismo que potencialmente puede superar esta dificultad.

ABSTRACT

In this paper we discuss the problem of the formal specification of communications protocols. Classical methods of specification are shown to be inadequate for the purpose, largely due to the presence of time as a key component in protocol descriptions. We propose the use of a new formalism which may potencially be able to overcome this difficulty.

* Profesor de la Universidad Simón Bolívar, Caracas, Venezuela. Ph.D. en Ingeniería de la Computación, Heriot-Watt University, Edimburgo, Escocia. Areas de interés: sistemas de operación, sistemas distribuidos, redes de computadores.

** Profesor de la Universidad Simón Bolívar, Caracas, Venezuela. Ingeniero de la Computación, Universidad Simón Bolívar. Areas de interés: lenguajes de programación, teoría de la computación, sistemas distribuidos.

INTRODUCCION

Los Sistemas de Especificación formal constituyen un area de investigación muy activa dentro de las Ciencias de la Computación. Una gran proporción de estos esfuerzos está dirigida hacia la especificación de programas secuenciales, y en particular hacia las estructuras de datos. Esto es natural, dada la importancia de la programación secuencial en términos de lo que hacen la gran mayoría de los programadores. Sin embargo, hay un creciente interés en las estructuras de control no secuenciales. Esto se debe en parte al hecho de que tales estructuras pueden ser más "naturales" en ciertas circunstancias, y en parte al simple hecho de que el mundo real no es secuencial, y por lo tanto un programa que pretende funcionar bien en su relación al mundo tiene que acomodarse a este hecho. Ejemplos de tales programas incluyen los sistemas de operación, las redes de computadoras, los sistemas tolerantes a fallas, los sistemas de "tiempo real", e inclusive muchos juegos computarizados.

Aquí nos restringimos a una sub-clase de los sistemas no secuenciales (vea [22] para una clasificación más completa), llamados los *sistemas distribuidos*. Para nosotros, un sistema distribuido es aquello en que no hay ningún control centralizado que sirva para sincronizar los diferentes partes autónomas. Así excluimos los sistemas de operación de máquinas aisladas, y las arquitecturas vectoriales, por ejemplo. En particular, nuestra preocupación es con la especificación de los protocolos de comunicación dentro de una red. El problema surgió cuando tuvimos que explicar algunos protocolos a estudiantes de pregrado. Como la especificación de estos sólo existía en lenguaje natural, dió lugar a muchas ambigüedades y hasta "*huecos*" en la definición. Estamos convencidos que un mecanismo formal para estos casos es no sólo deseable, sino necesario.

En este artículo, nos limitamos al caso de la comunicación entre dos nodos "*vecinos*" (es decir, sin nodos intermedios). Aún así, el problema de la especificación es no trivial, y hay que decir de entrada que no lo hemos resuelto. Sin embargo, nos parece interesante presentar lo que hemos hecho hasta ahora.

PROTOCOLOS DE COMUNICACION.

Un *protocolo de comunicación* es una disciplina o convención seguida por dos o más participantes ("*nodos*") para el intercambio eficiente de información a través de un medio de comunicación sujeto a errores (ver por ejemplo [20]). El hecho que hayan errores de transmisión de mensajes, incluyendo pérdidas totales de mensajes, hace que un protocolo tiene que ser un sistema tolerante a fallas. Para estar seguro de que sus mensajes están llegando bien, el transmisor espera "*acusos de recibo*", o "*ACKs*", enviados por el receptor. Esto quiere decir que las fallas de transmisión se detectan por la no llegada del ACK correspondiente. Debido a esto, el transmisor tiene el problema de cuando puede asumir que el ACK no le va a llegar, y retransmitir el mensaje. El momento en que esto se decide es señalado por un evento llamado *timeout*. (Nótese que puede haber varias causas de la ausencia del ACK en el momento de producirse el timeout, que son: el ACK no fué enviado, fué enviado y se perdió, o simplemente se retrasó. El transmisor no tie

ne forma de distinguir entre estas posiblidades).

Está claro, entonces, que toda buena especificación de un protocolo debe incluir el tiempo como elemento básico. Esto es lo que los hace difíciles a especificar, y a la vez interesantes a estudiar.

SOBRE LAS ESPECIFICACIONES.

Entendemos por una *especificación* una descripción o definición de una clase de objetos que lo distingue en forma precisa del resto del universo. En otras palabras, la especificación indica lo que todos los posibles miembros de la clase deben tener en común. Concretamente, en el caso de los protocolos, podemos decir que todas las posibles implementaciones de un protocolo dan todos los miembros de la clase definida por la especificación del mismo. Algunas ventajas de trabajar con buenas especificaciones son:

(1) Poder saber si una implementación cumple o no con su especificación.
(2) Poder demostrar que dos implementaciones son equivalentes.
(3) Poder demostrar propiedades de la especificación en sí, y por lo tanto de toda implementación posible.

Este último punto es tal vez el más interesante, en el sentido de que, si la especificación se hace con una herramienta formal, es decir bien fundada en la matemática o la lógica, nos abre la posibilidad en principio de poder *verificar* la especificación. En otras palabras, podríamos eventualmente determinar si la especificación es "*correcta*" o no, bajo ciertos criterios. Por ejemplo, en una especificación de protocolo, nos interesa saber si permite estados de "*deadlock*", si la secuencia de mensajes aceptados por un receptor es una secuencia inicial de los que el transmisor quiere enviar y que todo mensaje enviado será aceptado después de un número acotado de repeticiones (determinado por la frecuencia de errores en el medio). Además, nos interesan propiedades más universales - por ejemplo que la especificación es completa y que no es ambigua. En este artículo nos concentramos en las técnicas de especificaciones en sí, sin preocuparnos demasiado por los métodos de verificación.

A continuación, presentaremos en forma muy suscinta las características resaltantes de algunas metodologías que se han usado para la especificación de protocolos.

MAQUINAS DE ESTADOS FINITOS.

Los modelos basados en Máquinas de Estados Finitos fueron usados para especificaciones de protocolos desde el mismo momento en que se intentó resolver este problema [1, 2, 3]. Esto se debió, principalmente, a que los nodos se comportan como este tipo de máquinas: en todo instante de tiempo están en un estado diferenciable de todos los demás y ante la ocurrencia de algún evento, se produce una transición determinística a otro estado.

Cuando se usa este tipo de especificaciones, se considera como un estado cada combinación posible de valores de las variables del sistema, lo que excluye la especificación de procesos o entidades con variables con rangos de valores de cardinalidad infinita.

Por otra parte, aún en el caso de conjuntos finitos de valores, el número es estados a considerar crece exponencialmente si se aumenta el rango de valores de algunas de las variables o se añade un nuevo elemento al sistema, lo que hace la especificación intratable por su tamaño. Su aplicabilidad queda entonces restringida a protocolos muy sencillos que controlen un número pequeño de entidades.

Modelos posteriores intentaron paliar estos problemas mediante la compactación de la información de varios estados en un sólo estado. Esto por supuesto, disminuía la cantidad de estados a considerar, pero esta reducción no es lo suficientemente fuerte como para dejar una especificación clara y manejable del problema.

MODELOS BASADOS EN REDES DE PETRI.

Las redes de Petri [4] son el formalismo que más ha sido usado en la especificación de protocolos de comunicación. Esto se debe a que desde su creación han sido muy usadas en la descripción de sistemas de procesos concurrentes que se comunican. En el caso de protocolos, se puede detallar una larga lista de trabajos sobre el uso de esta metodología (ver [5, 6, 7, 8, 9, 10] a manera de ejemplo), pura o con enriquecimiento (Redes de Petri con condicionales [10, 12], con emisión y recepción de mensajes [11], etc). Son más aplicables que las máquinas de estados finitos, ya que en una red finita, donde el número de fichas puede crecer arbitrariamente, se pueden representar infinitos estados. Pero, a pesar de proporcionar facilidades para efectuar verificaciones sobre algunas especificaciones (*volveremos sobre este punto luego*), al intentar especificar problemas más complicados que los del Bit Alternado, el tamaño de la red resultante hace que la especificación sea inmanejable.

ESPECIFICACIONES USANDO LOGICA.

Otro formalismo usado en la especificación de protocolos, es la lógica proporcional. Basados en ella, Chen y Yeh [13] desarrollaron un lenguaje llamado EBS (*Event Based Specification Language*), que sirve para formular aseveraciones que describen las secuencias válidas de eventos que puede generar un conjunto de procesos. Una característica resaltante es la no utilización de relojes globales que regulen el comportamiento de todo el sistema. Definen, sin embargo, una relación de orden parcial, denotada por ' $\rightarrow$ ', para representar la precedencia entre eventos. Los elementos incomparables bajo esta relación son llamados concurrentes. Utilizan además, para formular sus aseveraciones, la relación de causalidad: a $\Rightarrow$ b significa que la ocurrencia del evento 'a', causa la ocurrencia en el futuro del evento 'b'. Nótese que si a $\Rightarrow$ b, entonces en toda secuencia válida de eventos a $\rightarrow$ b, pero el contrario no es necesariamente cierto.

LOGICA TEMPORAL.

Otro enfoque es la utilización de Lógica Temporal [14], constituída por elementos de la Lógica Clásica de Primer Orden mas los operadores temporales $\Box$, , y *Until*. Para interpretar estos operadores, se supone que el sistema está descrito por un conjun

to de secuencias de estado, que son sus posibles comportamientos. La fórmula $\Box A$ ("*henceforth* A") es cierta si A es cierta en el estado actual y en todos los estados futuros. Si A es cierta, entonces $\Box A$ es cierta en el estado actual o en algún estado futuro. A *until* B es cierta si A es cierta en todos los estados hasta el primer estado donde B es cierta.

Tanto EBS como la Lógica Temporal son resultados interesantes dentro del campo de la especificación de protocolos, pero tienen sus limitaciones. Facilitan la descripción de las secuencias correctas de eventos o estados de un sistema, pero no permiten decir en forma clara y precisa como lograr la producción de esas secuencias.

LENGUAJES DE PROGRAMACION DE ALTO NIVEL

Otra idea dentro del campo de la especificación de protocolos es la utilización de lenguajes de alto nivel para representar las entidades del sistema. Tiene la ventaja de describir en forma precisa la secuencialidad de las acciones que debe efectuar cada elemento del sistema y las estructuras usadas para manejar la información, pero no siempre proveen mecanismos claros para describir la concurrencia.

Uno de los lenguajes empleados para especificar protocolos es AFFIRM [15] , que es usado principalmente para hacer especificaciones algebraicas. Para especificar un protocolo en AFFIRM, se supone que cada estación es un tipo de datos abstracto. La ocurrencia de un evento se representa como la ejecución de una de las operaciones posibles sobre el tipo de datos. Una de sus principales desventajas es que no provee mecanismos para relacionar o sincronizar entidades separadas. Esto imposibilita hacer la especificación de una red, pudiendose sólo describir sus componentes.

Hemos experimentado también con el lenguaje CSP de Hoare [16] CSP tiene buenas estructuras de control para representar la concurrencia y el no determinismo. En particular, la comunicación entre procesos en CSP puede ser usado para modelar las actividades de nodos en una red. Sin embargo, no es propiamente un lenguaje de especificación, sino de programación, por lo tanto al usarlo, obtenemos sólo un miembro de la clase a ser especificada.

TECNICAS DE VERIFICACION

Una característica altamente deseable de un formalismo de especificación es la posibilidad de probar propiedades como las indicadas en la Introducción. En el espacio disponible aquí, no podemos profundizar sobre este aspecto. Sólo haremos las siguientes observaciones sobre algunos de los formalismos de especificación referenciales anteriormente.

Las técnicas de verificación más comunes, cuando se especifica usando Redes de Petri o metodologías afines, consisten en la generación exhaustiva de todos los estados alcanzables por la máquina que describe la especificación. Para ello, existen herramientas automatizadas, tales como OGIVE/OVIDE [18] y CESAR [17], que pueden realizar análisis de alcanzabilidad de estados dentro de una red acotada. Esto resulta ser suficiente para

probar muchas propiedades de la especificación, incluyendo "*liveness*" [23] y ausencia de "*deadlocks*".

En el caso de los lenguajes de programación de alto nivel, las verificaciones pueden hacerse mediante las técnicas clásicas de pruebas de invariantes. Este método tiene la desventaja de necesitar de demostradores automáticos de teoremas para poder realizar las verificaciones, y las herramienta mecanizadas disponibles aún tienen un dominio de aplicación muy limitado.

En la próxima sección, presentamos algunas ideas sobre el uso de otro formalismo de especificación que ofrece ciertas ventajas en este aspecto.

SPC

Una metodología de especificación que nos parece muy interesante es el Lenguaje Formal para Especificación de Procesos Concurrentes [21]. En este formalismo cada proceso se representa como un conjunto finito de reglas de reescritura con eventos. Este conjunto de reglas se puede interpretar como la descripción finita de un sistema (*posiblemente infinito*) de transición de estados. Bajo esta óptica, cada una de las reglas representa un posible cambio de estado del proceso, junto con la realización de un conjunto de comunicaciones (*posiblemente vacio*). Por ejemplo, si queremos describir el proceso Pila, que recibe números naturales por una puerta de entrada y los emite por una puerta de salida según el orden LIFO (*último en entrar, primero en salir*), el conjunto de reglas sería:

Estado Inicial	p( )	
p(W)	$\xrightarrow{\{\alpha.e(N)\}}$	p(cons(W,N))
p(cons(W,N))	$\xrightarrow{\{s.\alpha(N)\}}$	p(W)

Las comunicaciones en este formalismo se denotan por términos de la forma p1.p2(T), donde p1 es la puerta emisora, p2 la receptora y T el elemento que se transmite. En caso que el emisor o el receptor sea el ambiente donde se está ejecutando el programa, se coloca como puerta el símbolo 'α'.

La primera de las reglas especifica que si el contenido de la pila es la secuencia W, y llega un entero N como entrada, el resultado será una pila con N como elemento tope y la secuencia W como resto. La segunda dice que si la pila tiene como elemento tope N y resto W, puede emitir por la puerta de salida a N y quedar con W como contenido.

Una vez que se tienen las descripciones de un conjunto de procesos, se puede construir la descripción de un sistema de procesos comunicantes mediante la combinación de las especificaciones de los procesos constituyentes y el uso de los operadores ||, +, y - . El operador || (*paralelismo*), recibe como entrada las especificaciones de dos procesos P1 y P2 y produce como salida las reglas del proceso P que resulta al ejecutar P1 en paralelo con P2. El operador '+' toma como entrada las reglas que describen un proceso P

y un par de puertas (*una de entrada y otra de salida*) y produce la descripción del proceso que resulta a partir de P conectando las puertas indicadas. El operador'-' (*restricción*) produce la descripción de un proceso que resulta de "*ocultar*" las comunicaciones a través de un canal C, dentro de un proceso P, que le son dados como parámetros.

A contiuación usaremos esta metodología para especificar el protocolo del Bit Alternado. Supondremos la red constituida por cuatro procesos: el Transmisor, el Receptor, un proceso para representar el canal de comunicación simple del transmisor al receptor, y un proceso para representar el canal en sentido inverso. Representamos los canales como procesos porque, desde el punto de vista del protocolo, son entes activos que pueden distorsionar o perder parte de los mensajes que circulen por ellos. Para los efectos de la especificación, la distorsión y la pérdida de mensajes se consideran eventos equivalentes.

Transmisor

Este proceso tiene tres puertas, dos de entrada y una de salida. A través de la puerta de entrada 'α.e' recibe, desde el anfitrión, los mensajes a transmitir; por la puerta de salida 't.α' los transmite al exterior y por la puerta 'α.a' recibe ACKs. El conjunto de reglas que describen al proceso es:

Estado Inicial	i(0)	
i(X)	$\xrightarrow{\{\alpha \cdot e(M)\}}$	r(X,M))
r(X,M)	$\xrightarrow{\{t.\alpha(c(X,M))\}}$	1(X,M)
1(X,M)	$\xrightarrow{\{\alpha.a(X)\}}$	i(inv(X))
1(inv(X),M)	$\xrightarrow{\{\alpha.a(x)\}}$	r(inv(X),M)

Receptor

Tendrá dos puertas de salida y una de entrada. A través de 'α.s1' recibe mensajes que pasa al anfitrión usando la puerta 'o.α' y envía ACKs por la puerta 'i.α'.

Estado Inicial	e(0)	
e(X)	$\xrightarrow{\{\alpha.s1(c(X,M))\}}$	k(X,M)
k(X,M)	$\xrightarrow{\{o.\alpha(M).i.\alpha(X)\}}$	e(inv(X))
e(inv(X))	$\xrightarrow{\{\alpha.s1(c(X,M))\}}$	m(inv(X))
m(inv(X))	$\xrightarrow{\{i.\alpha(X)\}}$	e(inv(X))

Canal del Transmisor al Receptor (Proceso C1):

Recibirá mensajes a través de una puerta de entrada, que en forma no determinística emitirá por la puerta de salida 's.α', o no.

Estado Inicial W

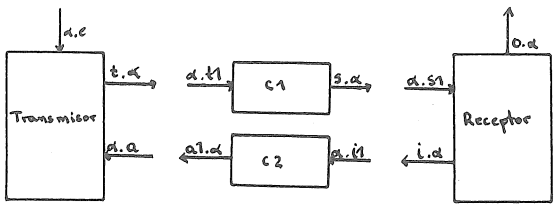
W	$\xrightarrow{\{\alpha.t1(X)\}}$	f(X)
f(X)	$\xrightarrow{\{s.\alpha(X)\}}$	W
f(X)	$\xrightarrow{\{\ \}}$	W

Canal Del Receptor al Transmisor (Proceso C2)

Estado Inicial d

d	$\xrightarrow{\{\alpha.i1(X)\}}$	h(X)
h(X)	$\xrightarrow{\{a1.\alpha(X)\}}$	d
h(X)	$\xrightarrow{\{\ \}}$	d

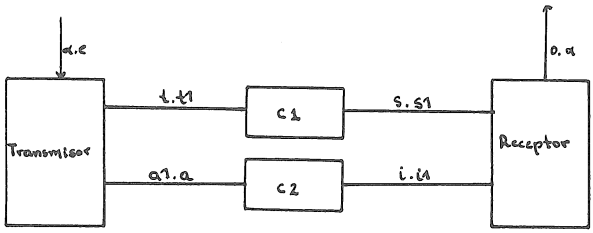
Hasta el momento tenemos las especificaciones aisladas de cuatro procesos. Gráficamente se puede ver de la siguiente forma:



Para tener la especificación del sistema de procesos que describa el comportamiento de la red, se debe calcular el proceso que resulta de colocar en paralelo los cuatro procesos, conectar los pares de puertas ('t.α', 'α.t1'),(s'.α','s1.α'),('α.a','a1.α'), ('i.α', 'α.i1') y *"ocultar"* las comunicaciones por las puertas 't.α','α.t1','s.α','s1.α','α.a', 'a1.α','i.α','i1.α'. Una expresión que representa lo anterior es:

$$((t \,\|\, C1) + 't.t1 - t.\alpha - \alpha.t1) \,\|\, ((R \,\|\, C2) + i1.i - \alpha.i - i1.\alpha)) + s.s1 + a.a1 - s.\alpha - \alpha.s1 - a1.\alpha - \alpha.a$$

El sistema resultante es:



Para poder estudiar con comodidad la aplicabilidad de este formalismo en la especificación de protocolos, se ha desarrollado un programa en PROLOG [19] que realiza evaluaciones de expresiones de este álgebra de procesos. Paralelamente, se están estudiando mecanismos para realizar verificaciones de propiedades de la especificaciones de redes de procesos que resultan de evaluar estas expresiones, con la finalidad de aplicarlas al caso específico de los protocolos. Estos estudios se encuentran avanzados y proximamente publicaremos los resultados obtenidos.

CONCLUSIONES

El problema de la especificación formal de los protocolos de comunicación es difícil. Las técnicas tradicionales de especificación de programas secuenciales, e incluso de sistemas de procesos concurrentes, no satisfacen los requerimientos necesarios para especificar un protocolo en forma cabal. Específicamente, carecen de mecanismos para describir restricciones de tiempo, lo cual es imprescindible en este contexto, y en muchos casos sus facilidades de verificación están limitadas tanto por el tamaño de lo especificado como por las propiedades que se pueden estudiar.

En este trabajo, hemos propuesto utilizar el formalismo SPC [21] , no empleado previamente para este problema. Bajo este formalismo, podremos tener una descripción finita de la máquina de estados que define el protocolo, con la ventaja adicional de poder hacer verificación de muchas propiedades de la especificación, incluyendo "*liveness*", ausencia de "*deadlocks*" y otros invariantes sobre secuencias de eventos.
Estamos trabajando ahora en el problema de como introducir el tiempo en SPC, para así contar con una herramienta más aplicable a este problema. Esperamos que ella pueda servir también para problemas relacionados, tales como sistemas de tiempo real, sistemas tolerantes de fallas, etc.

(1) Kawashima, Futani y Kand: "*Functional Specification of Call Processing by State Transition Diagram*", IEEE Trans. Comm. Tech., Vol COM19, Oct 1971.

(2) Birke: "*State Transition Programming Techniques and Their Use in Producing Teleprocessing Device Control Programs*", IEEE Trans. Comm., Vol. COM20, Nro 3, Junio 72.

(3) Bochmann C.: "*Communications Protocols and Error Recovery Procedures*", ACM Operating System Review, Vol 9, Nro. 3, Julio 1975.

(4) Petri C.A.: "*Kommunikation mit Automaten*", Schriften des Rheinish-Westfalischen Institutes fur Instrumentelle Mathematik an der Universitat Bonn, Heft 2, Bonn, W. Germany 1962; Tech. Fep. BADC-TR-65-337.

(5) Keller, R. M. "*Formal Verification of Parallel Programs*",CACM, Vol. /, 1976.

(6) Merlin, P. "*Specification and Validation of Protocols*", IEEE Trans. Comm. Vol. COM27, Nro. 11, Nov 1979.

(7) Ellis: "*Consistency and Correctness of Duplicate Database Systems*", Proc. of Sixth ACM Symp. in Operating Systems Principles, Nov 1977.

(8) Barlett, Scantlebury y Wilkinson: "*A note on Reliable Full-Duplex Transmissions over Half-Duplex Links*", CACM, Vol. 12, p. 260, 1969.

(9) Yoeli y Barzilai: "*Behavioral Descriptions of Communication Switching Systems Using Extended Petri Nets*", Digital Processes, Vol 3, 1977.

(10) Berthelot, Terrat: "*Petri Net Theory for the Correctness of Protocols*", IEEE Transactions on Communications. Vol Cc 30, Nro 12, Dic 1982.

(11) Jurgensen, Vuong: "*Formal Specification and Validation of ISO Transport Protocols Components Using Petri Nets*", ACM SIGCOMM, Sept. 1984.

(12) Courtiat, Ayache, Algayres: "*Petri Nets are Good for Protocols*", ACM SIGCOMM, Sept. 1984.

(13) Chen, Yeh: "*Formal Specification and Validation of Distributed Systems*" IEEE Proc. 2nd. Intl. Conf. on Distributed Computing Systems.

(14) Schwartz, Mellar-Smith: "*From State Machines to Temporal Logic: Specification Methods for Protocol Verification*", IEEE Trans. Comm. Vol COM 30, Nro 12, Dic 1982.

(15) Sunshine, Thompson, Brickson, Gerhart, Schabe: "*Specification and Validation of Communication Protocols in* AFFIRM *Using State Transiton Models*", IEEE Trans. on Soft. Eng., Vol SE8, Nro 5, Sept. 1982.

(16) C.A.R. Hoare: *"Communicating Sequential Processes"*, CACM, Agosto 1978.

(17) Queille: *"The CESAR System: A Computer Aided Design and Certification System for Distributed Applications"*. Proc. 2nd Int. Conf. on Distributed Computing Systems, Abril 1981.

(18) Montel: *"OVIDE: A software Package for the Validation of Systems Represented by Petri Net Pased Models"*, 4th. European Workshop on Application and Theory of Petri Nets, Toulouse, Sept. 1980.

(19) Roussel Ph.: *"PROLOG: Manuel de Référence et d'Utilisation"*, Groupe d'Intelligence Artificielle, Université d'Aix-Marseille.

(20) Tanenbaum A.: *Computer Networks"*, Prentice Hall Software Series, 1981.

(21) Pereira J. M.: *"Procesos Communicants"; Un Language Formal et sus Models, Problemes d'Analyse"*, Thêse de 3'eme Cycle, Junio 1984, Université de Grenoble.

(22) O'Callaghan y Foulk, *"Formal Description of Parallel Structure in AIDS"*, IEE Trans. on Computers and Digital Techniques, Marzo 1980, pp. 55-63.

(23) Peterson J.: *"Petri Net Theory and Modelling of Systems"*, Prentice-Hall Inc.