Anales PANEL'81/12 JAIIO
Sociedad Argentina de informática
e Investigación Operativa. Buenos Aires, 1981

ASIGNACION CONCURRENTE DE VECTORES Y MATRICES

Juan Carlos ANSELMI
Centro de Informaciones
y Estudios del Uruguay (CIESU)

1. INTRODUCCION

En ALGOL-60, y en muchos otros lenguajes de programación, para intercambiar los valores del J-ésimo y del K-ésimo elemento de un vector "S", es necesario utilizar una variable adicional (por ejemplo "Y") que sea del mismo tipo que los elementos de dicho vector.

```
Y := S(J)  ;  S(J) := S(K)  ;  S(K) := Y  ;
```

Sería mucho más claro y sencillo si esta permutación de valores se pudiera efectuar con una única sentencia, en donde a la izquierda del símbolo ":=" se especificara de alguna manera la pareja ordenada S(J) , S(K) , y a la derecha se especificara la pareja ordenada S(K) , S(J).

Este tipo de asignación múltiple, que permite definir varios elementos de un vector o de una matriz, presenta sin embargo algunos problemas, especialmente cuando se utilizan matrices multi-indizadas, o cuando en el miembro de la derecha se usa también la matriz indicada en el miembro de la izquierda.

E. W. Dijkstra opina que la causa de estos problemas no está en la asignación concurrente sino en la noción de variable indizada.

"However, I have now come to the conclusion that it is not the concurrent assignment, but the notion of the subscripted variable that is to be blamed" (cf. DIJKSTRA-76, capítulo 11, página 95).

El objetivo del presente trabajo es precisamente el de revisar el concepto de matriz, definiendo con precisión y claridad las manipulaciones que serán permitidas y los valores de los índices que podrán ser usados, a efectos de posibilitar la aplicación de la asignación concurrente.

El uso de algún tipo de asignación concurrente en los lenguajes de programación o en los lenguajes de especificación, permite generalmente expresar los algoritmos de cálculo más brevemente y con mayor claridad.

Supongamos, por ejemplo, que se tiene un vector "P" cuyo índice varía entre 1 y 60, y que se desea reubicar todos los valores de elementos correspondientes a valores impares del índice, colocándolos en forma correlativa a partir del elemento P(11) y hasta el elemento P(40).

Esta transformación puede ser efectuada con el siguiente algoritmo:

```
L := 20  ;  M := 19  ;

MIENTRAS  L > 10  REPETIR :  P(L) := P(M) ;
                             L := L - 1 ;
                             M := M - 2 ;
L := 22  ;  M := 23  ;
MIENTRAS  L < 41  REPETIR :  P(L) := P(M) ;
                             L := L + 1 ;
                             M := M + 2 .
```

Aplicando el sistema propuesto en el presente trabajo, se puede indicar esta misma transformación del vector "P" con la siguiente asignación concurrente:

```
P(11:40) := P(1:59:2) .
```

Esta última sentencia es evidentemente mucho más corta que el algoritmo antes detallado.

La economía de líneas de texto del programa cuando se utiliza la asignación concurrente, puede llegar a ser bastante importante, especialmente cuando se utilizan matrices de dos o más índices.

La asignación concurrente abre también nuevas perspectivas (y muy interesantes por cierto) en las computadoras que permiten desarrollar en paralelo varias tareas.

En efecto, en este caso, el programador podría continuar programando en la forma secuencial habitual, y la máquina, en forma automática, podría introducir el paralelismo a nivel interno de cada sentencia de asignación concurrente, ya que una parte de la copia de valores podría ser efectuada por un procesador, y la parte restante podría ser realizada paralelamente por otro procesador.

2. VALORES POSIBLES DE UN INDICE

Los valores permitidos de un índice serán indicados en forma abreviada por tres parámetros enteros, los cuales serán especificados separándolos con el carácter ":". El conjunto índice A:B:V estará compuesto por los valores enteros A , A+V , A+V+V , A+V+V+V , ... , B-V-V , B-V , y B.

Se admitirá que "A" pueda ser menor, igual o mayor que el valor "B". Además, si "A" es igual a "B", el valor "V" podrá ser cualquier valor entero, pero si "A" es distinto de "B", entonces el valor "V" deberá ser siempre entero, no nulo, del mismo signo que (B-A), y tal que el valor absoluto de (B-A) sea un múltiplo entero del valor absoluto de "V".

Dado el conjunto de valores posibles A:B:V de un índice, diremos que "A" es el primer valor índice, que "B" es el último valor índice, y que "V" es la variación del índice.

El conjunto índice A:B:V, con "B" estrictamente mayor que "A" y "V" igual a la unidad, será indicado abreviadamente por A:B. Cuando "B" es estrictamente menor que "A" y "V" es igual a -1, entonces el conjunto índice A:B:V también será indicado abreviadamente por A:B.

Además, A:A:V será indicado abreviadamente por A:A, cualquiera sean los valores enteros "A" , "V".

Evidentemente, si "A" es igual a "B", el conjunto índice A:B:V estará compuesto por un único valor índice; y si "A" es distinto de "B", el conjunto índice A:B:V estará formado por n = (B-A+V)//V valores índice (la división entera ha sido indicada con el símbolo "//") . La función que expresa el número de valores diferentes de un conjunto índice será indicada en lo sucesivo con el nombre "dim" .

Si $A \neq B$, entonces : (A:B:V).dim = (B-A+V)//V

Si $A = B$, entonces : (A:B:V).dim = 1

Conviene aclarar que un índice está caracterizado por los valores que puede adoptar, y por el orden creciente o decreciente en el cual dichos valores son considerados. En este sentido, el conjunto índice A:B:V es generalmente diferente del conjunto índice B:A:-V , pues si bien ambos definen los mismos valores índice, difieren en el orden en el cual dichos valores índice son considerados en el caso que"A" sea distinto de "B".

Ya hemos dicho que no todas las ternas de parámetros enteros caracterizan sin ambigüedad un conjunto de valores índice, pues se requieren ciertas condiciones. Dichas condiciones pueden ser verificadas con la función "test", que al aplicarla a la terna A:B:V proporciona el valor lógico ".verdadero." cuando "A" es igual a "B" , o cuando siendo "A" distinto de "B" , se cumple que "V" es no nulo, del mismo signo que (B-A), y tal que el valor absoluto de (B-A) es un múltiplo entero del valor absoluto de "V" ; en caso contrario, la función "test" proporciona el valor lógico ".falso.".

Al aplicar la misma función "test" a una terna A:B:V y a un valor entero "C", se puede determinar si dicho valor entero es uno de los valores índice del conjunto A:B:V. En este caso, (A:B:V).test(C) será igual a ".verdadero." si y sólo si se cumplen las siguientes tres condiciones:

a) se verifica que (A:B:V).test = .verdadero. ;

b) el valor absoluto de (C-A) es igual a cero o es un múltiplo entero del valor absoluto de "V" ;

c) el valor (C-A)*(C-B) es siempre negativo o nulo.

Es posible también determinar si todos los índices del conjunto índice C:D:W son también índices del conjunto índice A:B:V , aplicando la función "test" a estas dos ternas ; esta aplicación de la función "test" está evidentemente orientada a la utilización de subconjuntos índice.

Concretamente, (A:B:V).test(C:D:W) será igual al valor lógico ".verdadero." si y sólo si se cumplen las tres condiciones siguientes:

a) (A:B:V).test = .verdadero. ;

b) (C:D:W).test = .verdadero. ;

c) Si (C:D:W).test(E) = .verdadero. , entonces también se cumple (A:B:V).test(E) = .verdadero. .

Detalladamente, las condiciones b) y c) anteriores son equivalentes a las siguientes:

d1) El valor absoluto de (C-A) es igual a cero o es un múltiplo entero del valor absoluto de "V" ;

d2) El valor (C-A)*(C-B) nunca es estrictamente mayor que cero ;

d3) Se verifica una de las siguientes dos situaciones :

1er. caso

1) Se cumple que "C" es igual a "D" ;

2do. caso

2a) Se cumple que "C" y "D" no son iguales ;

2b) El valor "W" es no nulo y del mismo signo que el valor (D-C) ;

2c) El valor absoluto de (D-C) es múltiplo entero del valor absoluto de "W" ;

2d) El valor absoluto de "W" es múltiplo entero del valor absoluto de "V" ;

2e) El valor (D-A)*(D-B) nunca es estrictamente mayor que cero .

A partir de estas definiciones, se posible deducir varias propiedades de la función "test" , entre ellas las siguientes:

1) Si (A:B:V).test , entonces : (B:A:-V).test ;
2) Si (A:B:V).test(C) , entonces : (A:B:V).test ;
3) Si (A:B:V).test(C) , entonces : (A:C:V).test ;
4) Si (A:B:V).test(C) , entonces : (C:B:V).test ;
5) Si (A:B:V).test , entonces : (A:B:V).test(A) ;
6) Si (A:B:V).test , entonces : (A:B:V).test(B) ;
7) Si (A:B:V).test , entonces : (A:B:V).test(A:B:V) ;
8) Si (A:B:V).test , entonces : (A:B).test(A:B:V) ;
9) Si (A:B:V).test , entonces : (A:B:V).test(B:A:-V) ;
10) Si (A:B:V).test(C:D:W) , entonces : (A:B:V).test(C) ;
11) Si (A:B:V).test(C:D:W) , entonces : (A:B:V).test(D) ;
12) Si (A:B:V).test(C:D:W) , entonces : (A:B:V).test ;
13) Si (A:B:V).test(C:D:W) , entonces : (C:D:W).test ;
14) Si (A:B:V).test , entonces : (A:B:V).test(A:A) ;
15) Si (A:B:V).test , entonces : (A:B:V).test(B:B) ;

16) Si (A:B:V).test(C) , entonces : (A:B:V).test(A:C:V) ;

17) Si (A:B:V).test(C) , entonces : (A:B:V).test(B:C:-V) ;

18) Si (A:B:V).test , entonces : (A:B:V).test(A:B:B-A) ;

19) Si (A:B:V).test , entonces : (A:B:V).test(B:A:A-B) ;

20) Si (A:B:V).test(C:D:W) , entonces : (A:B:V).test(D:C:-W) .

3. EL CONCEPTO DE MATRIZ Y LAS FUNCIONES QUE LE SON APLICABLES

Una matriz estará caracterizada por uno o varios conjuntos índice del tipo de los definidos en la sección 2. , por el conjunto de los valores posibles de sus elementos, y por una función de acceso con dominio en el producto cartesiano de los conjuntos índice asociados a la matriz, y con alcance en el conjunto de valores posibles de los elementos de la matriz ; esta función de acceso puede permanecer invariable durante todo el proceso del programa o de la tarea, o puede ser total o parcialmente modificada por una sentencia de asignación o por una operación de entrada, en uno o en varios puntos del desarrollo del algoritmo.

A efectos de simplificar la presentación de este trabajo, en lo que sigue supondremos que todos los vectores y las matrices que se utilizarán serán del tipo "T", mientras que las variables elementales podrán ser del tipo "T" o de tipo entero, y en este último caso serán utilizadas como variables índice o como valores particulares de un índice.

Este tipo "T" podrá ser, por ejemplo, el tipo de datos "real", o el tipo "caracter", o el tipo "lógico", o cualquier otro tipo de datos previsto en el lenguaje de programación a utilizar.

Una matriz multi-indizada será creada por una función especial de creación, que en lo sucesivo llamaremos "crea".

Por ejemplo, U.crea(1:8) permite crear un vector llamado "U" cuyo único índice puede adoptar cualquier valor entero no menor que el valor unidad y no mayor que "8".

Por su parte, R.crea(1:7:2,9:-9) permite crear una matriz bi-indizada "R" cuyo primer índice puede tomar los valores 1, 3, 5, 7 , y cuyo segundo índice puede tomar los valores enteroscomprendidos entre -9 y 9.

En general, S.crea(A:B:V) permite crear un vector "S" cuyo único índice puede adoptar los valores del conjunto índice A:B:V , Z.crea(A:B:V,AA:BB:VV) permite crear una matriz "Z" cuyos dos índices pueden tomar respectivamente alguno de los valores definidos por los conjuntos índice A:B:V , AA:BB:VV , y análogamente para las matrices tri-indizadas y las otras matrices multi-indizadas.

Puesto que parece razonable que las variables elementales tales como "X" o "Y" también puedan ser análogamente creadas con X.crea e Y.crea, es conveniente considerar a dicho tipo - de variables como un caso particular de matriz multi-indiza-- da, las cuales no tendrían asociado ningún índice ; en este - caso, diremos que se trata de matrices elemento. Nótese que - la función de acceso asociada a una matriz de este tipo, permite obtener el valor correspondiente al único punto anónimo del dominio.

Obsérvese que no hemos especificado los valores de los elementos de las matrices creadas de esta forma, pues se supone que dichos valores serán posteriormente definidos por una o varias sentencias de asignación o por otro procedimiento conveniente ; inicialmente, todos los elementos de una matriz podrán ser definidos en forma automática con un valor estandar (por ejemplo, cero, .falso. , etc. , o con un valor especial que señale la indefinición) .

El número de índices de una matriz multi-indizada puede hacerse accesible con una función llamada "dom" ; para las ma-- trices de los ejemplos anteriores, se cumplirá :

```
U.dom = 1  ;  R.dom = 2  ;  S.dom = 1  ; Z.dom = 2  ;
X.dom = 0  ;  Y.dom = 0  .
```

Un índice particular de una matriz multi-indizada podrá -- ser representado por el número de orden correspondiente antecedido por el signo "?" ; los diferentes índices de una ma- - triz "Q" serán así identificados por ?1 , ?2 , ?3 , etc., hasta ?(Q.dom) .

El conjunto índice asociado a cierto índice de una matriz puede ser accedido con una función de acceso llamada "ci" ; para las matrices "U" , "R" , "Z" anteriormente creadas, se tendrá:

```
U.ci(?1) = 1:8 ; R.ci(?1) = 1:7:2 ; S.ci(?1) = A:B:V ;
Z.ci(?1) = A:B:V ; Z.ci(?2) = AA:BB:VV .
```

Por su parte, puede accederse también al primer índice, al último índice, y a la variación de un índice de una matriz - multi-indizada, con las funciones de acceso "pi" , "ui" , "vi" ; estas funciones pueden ser directamente aplicadas a una matriz, o aplicadas al resultado proporcionado por la función "ci" . Por ejemplo:

```
U.ci(?1).pi = U.pi(?1) = 1  ;U.ci(?1).vi = U.vi(?1) = 1 ;
R.ci(?2).ui = R.ui(?2) = -9 ;R.ci(?2).vi = R.vi(?2) = -1 ;
Z.ci(?1).pi = Z.pi(?1) = A  ;Z.ci(?1).ui = Z.ui(?1) = B ;
U.ci(?1).ui = U.ui(?1) = 8  ;R.ci(?1).vi = R.vi(?1) = 2 ;
```

R.ci(?2).pi = R.pi(?2) = 9 ; S.ci(?1).vi = S.vi(?1) = V .

Análogamente se utilizará la función "dim", que expresa el número de valores diferentes de un índice.

U.ci(?1).dim = U.dim(?1) = 8 ; R.ci(?2).dim = R.dim(?2) = 19

Es posible considerar todos o una parte de los elementos de una matriz, estructurándolos de tal forma de obtener una nueva matriz ; diremos que esta nueva matriz es una submatriz de la anterior.

Una submatriz de la matriz "Z" quedará perfectamente definida si, para cada índice de "Z" , se especifica el único valor índice que será considerado, o si se especifica el subconjunto índice que será tomado en cuenta.

El número de índices de una submatriz de "Z" será pues -- igual al número de subconjuntos índice utilizados ; evidentemente, dicho valor nunca podrá exceder al número de índices - de la propia matriz "Z".

Una submatriz de la matriz "Z" será especificada utilizando una función especial que llamaremos "sub" . Dicha función se aplicará a la matriz original y a la lista de valores índice y de conjuntos índice que caracterizan a la submatriz.

Se indican a continuación varios ejemplos de submatrices - de las matrices "U", "R" antes definidas.

U.sub(4) ; U.sub(1:8) ; U.sub(1:4) ; U.sub(1:7:2) ;

U.sub(8:1) ; U.sub(5:5) ; U.sub(8:2:-2) ; U.sub(8) ;

R.sub(5:5,0:9) ; R.sub(3,9:1:-2) ; R.sub(1:7:6,0:0) ;

R.sub(5:1:-4,-9:9:2) ; R.sub(1,0) ; R.sub(7:7,-9:-9) .

Obsérvese que la submatriz U.sub(4:4) es diferente de la submatriz U.sub(4), ya que la primera submatriz tiene un único índice, mientras que la segunda no tiene ninguno.

Obsérvese también que la propia matriz original es siempre submatriz de sí misma.

Evidentemente, a efectos de definir correctamente una submatriz de una matriz "Z", se deberá especificar para cada índice de "Z", o bien un valor índice factible, o bien un subconjunto índice del conjunto índice originalmente asociado a ese índice de "Z" ; por lo tanto, si aplicamos la función -- "test" a este último conjunto índice, y al valor índice o al subconjunto índice correspondiente y asociado a la submatriz que se desea definir, se debe obtener siempre el resultado -- ".verdadero." si la submatriz es viable. Parece entonces natural introducir un mecanismo que permita efectuar este control

en forma concurrente para todos y cada uno de los índices de una matriz.

Dicho mecanismo será implementado permitiendo la aplicación de la función "test" a una matriz y a una lista de valores índice y de conjuntos índice. Deberá suponerse en este caso que la función "test" se aplica a cada conjunto índice de la matriz, y al valor índice o al conjunto índice que le corresponda en la lista. El resultado global de la función "test" será igual a ".verdadero." si y sólo si cada una de estas evaluaciones parciales proporciona el resultado ".verdadero." .

Evidentemente, si "S" es un vector, entonces se cumplen las siguientes relaciones:

```
S.test(C) = S.ci(?1).test(C)
S.test(C:D:W) = S.ci(?1).test(C:D:W)
```

Los siguientes ejemplos de aplicación de la función "test" a las matrices "U", "R" antes definidas dan todos como resultado el valor lógico ".verdadero." .

```
U.test(7:7) ; U.test(8:1:-7) ; U.test(2:5) ; U.test(2) ;
U.test(6:4) ; R.test(7:1:-2,9:-9:-2) ; R.test(5,-7:7) ;
R.test(7:3:-4,-7) ; R.test(5:5,4:4) ; R.test(5,-7:-3) .
```

Por el contrario, las siguientes evaluaciones dan todas como resultado el valor lógico ".falso." .

```
R.test(7:3:2,5:-8) ; R.test(5,9:0:-2) ; R.test(0,0) ;
R.test(9:-9,1:7:2) ; R.test(1:7:3,-9:9:3) ; U.test(9) ;
U.test(1:8:2) ; R.test(7:1,7:1) ; R.test(7:3:1.7:1) .
```

Obsérvese que la función "sub" permite especificar una subestructura de datos totalmente contenida en una matriz multiindizada. Esto puede ser necesario a efectos de re-definir dicha subestructura con una sentencia de entrada de información o con una sentencia de asignación, o a efectos de especificar un argumento de salida o de entrada/salida en una invocación a un subprograma.

En otros casos, sólo interesará acceder a los valores de los elementos de una submatriz, y no a la subestructura misma ; a efectos de permitir este acceso, será necesario introducir una nueva función que llamaremos "val" .

La función "val" también se aplicará a una matriz y a una lista de valores índices y de conjuntos índice, en forma similar a lo que se efectuaba con la función "sub".

Nótese que las funciones "sub" y "val" permiten expresar una sentencia de asignación de una manera mucho más explícita.

Evidentemente, es mucho más significativa para el programador una sentencia como U.sub(2) := U.val(8) , que la usual y más desprolija sentencia de asignación U(2) := U(8) .

Claro que puede argumentarse que esta última forma es una abreviación de la primera, lo cual es cierto. Esto no desmerece la utilidad de las funciones "sub" y "val" , no sólo desde el punto de vista conceptual, sino por las grandes ventajas que aportan al utilizarlas para indicar los argumentos en una invocación a un subprograma.

En efecto, con el uso de dichas funciones, resulta totalmente innecesario plantear a nivel práctico los clásicos dos tipos de transmisión de argumentos (por valor y por nombre) que son causa frecuente de errores de programación. Además, ello permite que se puedan efectuar ciertos controles suplementarios durante la compilación del programa, ya que será incorrecto utilizar la función "val" cuando el argumento es de salida o de entrada/salida, y será innecesario usar la función "sub" cuando el argumento es de entrada. Evidentemente, estos controles podrán ser efectuados automáticamente por el compilador, si el lenguaje de programación utilizado permite declarar los distintos tipos de argumentos formales empleados.

Diremos que dos matrices son homólogas cuando tienen igual número de índices, y cuando sus índices correspondientes tienen igual dimensión.

Por lo tanto, dos matrices "Z", "ZZ" serán homólogas si y sólo si se cumplen las siguientes condiciones:

(a) Z.dom = ZZ.dom

(b) Z.dim(?k) = ZZ.dim(?k) para k en (1..(Z.dom))

El conjunto de valores enteros comprendidos entre 1 y el valor Z.dom ha sido indicado abreviamente por (1..(Z.dom)) ; debe entenderse que dicho conjunto es vacío si Z.dom = 0 .

Se indican seguidamente algunos ejemplos de matrices que son homólogas.

(1) U.sub(7:4) y R.sub(1:7:2,4)

(2) Z.sub(A,BB) , U.sub(5) , X.sub e Y.sub

(3) Z.sub(B,BB:BB) y U.sub(3:3)

(4) S.sub(A:B:V) y Z.sub(B:A:-V,AA)

(5) R.sub(7:3:-2,0:9) y R.sub(1:5:2,-9:9:2)

Evidentemente, la homología entre matrices es una relación de equivalencia.

Diremos que dos matrices "Z" , "ZZ" son homólogas en el sentido amplio, cuando se cumple la relación Z.dim(?k) igual a ZZ.dim(?k) para todo valor entero positivo de k que sea menor o igual que el máximo de los valores Z.dom , ZZ.dom . Debe entenderse que si k es estrictamente mayor que Z.dom , entonces debe sustituirse en la relación anterior Z.dim(?k) por el valor unidad ; análogamente, si k es estrictamente mayor que ZZ.dom , entonces debe operarse la sustitución de ZZ.dim (?k) por el valor unidad.

La homología en el sentido amplio también es una relación de equivalencia definida en el conjunto de las matrices.

Evidentemente, la matriz elemento S.sub(A) es homóloga en el sentido amplio al vector U.sub(3:3) y al vector R.sub(5,9:9) .

4. ASIGNACION CONCURRENTE DE UNA MATRIZ CON LOS VALORES DE OTRA MATRIZ.

La definición concurrente de los elementos de una matriz con los valores de los elementos de otra matriz sólo será posible cuando ambas matrices son homólogas en el sentido amplio.

La asignación se hará elemento a elemento ; cada elemento de la matriz indicada en el miembro izquierdo de la sentencia de asignación será definido con el valor del elemento correspondiente del otro miembro.

Por ejemplo, U.sub(1:8:7) := U.val(8:1:-7) permite permutar los valores de los elementos U(1) y U(8).

Por su parte, U.sub(1:8) := U.val(8:1) es equivalente al algoritmo detallado de asignación que se indica a continuación.

```
LA := 4 ; LB := 5 ;

MIENTRAS  LA > 0  REPETIR:

          Y := U(LA) ;
          U(LA) := U(LB) ;
          U(LB) := Y ;
          LA := LA - 1 ;
          LB := LB + 1 .
```

En el caso U.sub(2:3) := R.val(3,0:-3:-3) , el elemento U(2) es definido con el valor de R(3,0) , y el elemento U(3) con el valor de R(3,-3).

5. COMPOSICION DE VALORES

Las constantes y los valores de los elementos de las matrices pueden componerse para formar estructuras de datos mayores, utilizando los siguientes operadores de composición:

a) el operador "," para la composición al nivel del primer índice ;

b) el operador ";" para la composición al nivel del segundo - índice ;

c) el operador ";;" para la composición al nivel del tercer índice ;

d) en general, el operador ";?k;" para la composición al nivel del k-ésimo índice (este operador incluye como casos - particulares a los tres anteriores) .

Para poder componer los valores de una matriz "Z" con los valores de una matriz "H" al nivel del k-ésimo índice, se debe verificar la siguiente condición :

a) Se debe cumplir que Z.dim(?j) = H.dim(?j) para todos los - valores de j diferentes de k y menores o iguales que el máximo de los valores Z.dom , H.dom . Debe entenderse que si j es estrictamente mayor que Z.dom , entonces en la relación anterior debe sustituirse Z.dim(?j) por el valor unidad ; análogamente, si j es estrictamente mayor que H.dom, entonces debe operarse la sustitución de H.dim(?j) por el valor unidad.

Esta composición produce una estructura de datos de tipo - matricial, que llamaremos Z;?k;H , y que tiene las siguientes características :

a) El valor (Z;?k;H).dom es igual al máximo de los tres valores k , Z.dom , H.dom .

b) Se cumple (Z;?k;H).dim(?j) = Z.dim(?j) = H.dim(?j) para todos los valores de j diferentes de k y menores o iguales - que el valor (Z;?k;H).dom ; en la relación anterior, debe sustituirse Z.dim(?j) y/o H.dim(?j) por el valor unidad, toda vez que exista indefinición de dichos valores.

c) Se verifica que (Z;?k;H).dim(?k) = Z.dim(?k) + H.dim(?k) ; debe suponerse que si k es estrictamente mayor que el valor Z.dom, entonces en la relación anterior debe sustituirse Z.dim(?k) por el valor unidad (y análogamente para - - H.dim(?k)).

En todo lo anterior, se ha supuesto que tanto j como k son siempre valores enteros positivos.

La composición (Z;?k;H) produce pues una estructura matricial de valores de las características antes indicadas, en -- donde para las primeras ocurrencias del k-ésimo índice se tienen los valores de los elementos de la matriz "Z" , y para - las últimas ocurrencias del k-ésimo índice se tienen los valores de los elementos de la matriz "H" .

La composición puede también operarse entre valores de submatrices, entre constantes, o entre constantes y valores de - submatrices ; en estos últimos dos casos, las constantes serán asimiladas a matrices elemento.

Los distintos operadores de composición pueden aplicarse - en forma reiterada ; en este caso, la utilización de paréntesis permitirá evitar ambigüedades.

Por ejemplo, (0 , 1 , 28 , -2) forma un vector de cuatro elementos.

Por su parte, el siguiente ejemplo forma una matriz de dos índices, el primero de los cuales tiene dimensión dos y el segundo dimensión tres.

((1 , 2) ; (11 , 12) ; (21 , 22))

La misma estructura, con los mismos valores,puede ser también expresada de la forma que se indica a continuación.

((1 ; 11 ; 21) , (2 ; 12 ; 22))

La composición (0.0 ; 1.0) forma una matriz tri-indizada cuyo primer índice tiene dimensión 1 , su segundo índice tiene también dimensión 1 , y su último índice tiene dimensión - 2 .

6. LA ASIGNACION CONCURRENTE EN EL CASO GENERAL

El miembro izquierdo será siempre una matriz o una subma-- triz. El miembro derecho será una composición de valores.

Las dimensiones de los índices correspondientes de ambos - miembros deben ser iguales entre sí, y las dimensiones de los índices que no tengan correspondiente en el otro miembro deben ser iguales a la unidad.

La asignación se hará elemento a elemento. Cada elemento - de la matriz indicada en el miembro izquierdo será definido - con el valor correspondiente indicado en el otro miembro.

A efectos de que el algoritmo que permita efectuar la asignación concurrente sea razonablemente simple, no se permitirá

utilizar en el miembro derecho más de una submatriz correspondiente a la matriz indicada en el miembro izquierdo.

Por ejemplo, la sentencia

```
P.sub(1:3:2,0:2:2) := (R.val(3:1:-2,2) ; ( 5.0 , 8.1 ) )
```

asigna el valor de R(3,2) al elemento R(1,0) , el valor de R(1,2) al elemento R(3,0) , el valor 5.0 al elemento R(1,2) , y el valor 8.1 al elemento R(3,2) .

7. CONCLUSIONES

Opinamos que los conceptos vertidos son importantes en varios aspectos.

Para el programador de aplicaciones, la asignación concurrente le permitirá plantear algoritmos más cortos y claros (aunque más no sea en un lenguaje de especificación y como etapa previa a la codificación del programa en un lenguaje de programación), y las funciones "sub" y "val" le permitirán expresar los argumentos de los subprogramas de una manera más racional.

Para el investigador que se preocupa de la definición de nuevos lenguajes de programación, el nuevo concepto de matriz introducido en este trabajo lo llamará a la reflexión sobre las ventajas que dicho concepto aporta, permitiéndole posiblemente apreciar más claramente los problemas que puede causar la utilización de este tipo de estructura de datos.

8. BIBLIOGRAFIA

(DIJKSTRA-76) - A discipline of programming - E.W. Dijkstra - Prentice-Hall, Englewood Cliffs (New Jersey), 1976.

(ANSELMI-78a) - Sous-structures d'un tableau - J.C. Anselmi - Electricidad de Francia, Clamart (Francia), Publicación EDF HI 2695/02, febrero de 1978.

(ANSELMI-78b) - An algorithm descriptive language, the "LEAL" Language : treatment of data structures - J.C. Anselmi - Electricidad de Francia, Clamart (Francia), agosto de 1978.