



OFICINA ESPAÑOLA DE
PATENTES Y MARCAS

ESPAÑA



⑪ Número de publicación: **2 166 279**

⑫ Número de solicitud: 009902055

⑤① Int. Cl.⁷: G06F 9/06
G06F 9/30

⑫

PATENTE DE INVENCION

B1

⑫② Fecha de presentación: **09.09.1999**

⑫③ Fecha de publicación de la solicitud: **01.04.2002**

Fecha de concesión: **12.03.2003**

⑫⑤ Fecha de anuncio de la concesión: **16.04.2003**

⑫⑤ Fecha de publicación del folleto de patente:
16.04.2003

⑦③ Titular/es: **UNIVERSIDAD POLITÉCNICA
DE VALENCIA**
Univ. Politécnica Valencia-CTT-
Camino de Vera, s/n
46022 Valencia, ES

⑦② Inventor/es: **Albert Albiol, Elvira;**
Alpuente Frasnado, María y
Vidal Oriola, Germán

⑦④ Agente: **No consta**

⑤④ Título: **Un método y sistema para la optimización de programas multiparadigma por transformación automática.**

⑤⑦ Resumen:

Un método y sistema para la optimización de programas multiparadigma por transformación automática. La presente invención se refiere a un método y sistema automático de especialización de programas que sirve para optimizar programas que integran funciones, lógica de predicados y concurrencia. El método comprende tanto la preservación del comportamiento operacional del programa a especializar, como la efectividad de la transformación, es decir, el incremento de eficiencia de los programas transformados.

ES 2 166 279 B1

Aviso: Se puede realizar consulta prevista por el art. 37.3.8 LP.

Venta de fascículos: Oficina Española de Patentes y Marcas. C/Panamá, 1 - 28036 Madrid

DESCRIPCION

Un método y sistema para la optimización de programas multiparadigma por transformación automática.

Campo técnico

La presente invención se refiere a un método y sistema automático de evaluación parcial de programas que sirve para optimizar programas multiparadigma.

Un lenguaje multiparadigma es aquél que incluye características de varios lenguajes. Concretamente, en esta invención nos referimos a lenguajes que integran funciones, lógica de predicados y concurrencia, y cuya ejecución se basa en la combinación de dos principios operacionales diferentes: estrechamiento y residuación.

Antecedentes de la invención

La evaluación parcial es una técnica de transformación de programas que consiste en la especialización de un programa de ordenador con respecto a ciertos datos de entrada, conocidos en tiempo de compilación. El programa resultante se puede ejecutar más eficientemente ya que, usando el conjunto de datos conocidos, es posible evitar algunas computaciones en tiempo de ejecución que se realizarán, una única vez, durante el proceso de especialización. Una de las principales razones que suscitan el interés por las técnicas de evaluación parcial es la posibilidad de conseguir implementaciones eficientes, por métodos puramente automáticos, de programas generales y altamente paramétricos, como los que se incluyen por ejemplo en las bibliotecas de módulos software. Esto permite escribir programas de ordenador de propósito general, usualmente poco eficientes, y utilizar las técnicas de evaluación parcial para generar programas especializados (de propósito concreto) cuya ejecución es mucho más eficiente.

Los lenguajes que integran funciones y lógica de predicados, conocidos como lenguajes de programación lógico-funcionales, se basan en un mecanismo de ejecución al que nos referiremos como estrechamiento. Los lenguajes lógico-funcionales de nueva generación, conocidos como lenguajes lógico-funcionales con residuación, combinan este mecanismo de estrechamiento con la posibilidad de suspender o residuar algunas llamadas a función, lo cual hace posible su ejecución concurrente. El principio de residuación se basa en la idea de retrasar la evaluación de las llamadas a función hasta que estén listas para una evaluación determinista. Cuando una expresión no está preparada para su evaluación, queda en estado de espera (de un posible valor de instanciación para sus parámetros de entrada) y decimos que es una expresión "suspendida". El mecanismo de estrechamiento realiza la instanciación de variables en las expresiones de entrada y aplica pasos de reducción a la expresión instanciada.

En el modelo combinado, el mecanismo operacional concreto (residuación o estrechamiento) para cada función se especifica por medio de anotaciones de evaluación. Las funciones deterministas se anotan como funciones *rígidas* (lo cual fuerza su evaluación determinista mediante residuación), mientras que las funciones indeterministas se declaran como funciones *flexibles* (per-

mitiendo pasos de estrechamiento).

La evaluación parcial o especialización de programas fue introducida por Futamura en "Partial Evaluation of Computation Process", Systems, Computers, Controls, Vol. 2(5), pp. 45-50, 1971. Posteriormente, se ha aplicado de forma intensiva en el campo de los lenguajes de programación funcional y los lenguajes de programación lógica. Las técnicas convencionales de evaluación parcial para programas funcionales se basan en la reducción de expresiones y en la propagación de constantes, mientras que las técnicas empleadas para la evaluación parcial de programas lógicos explotan la propagación bidireccional de parámetros basada en el mecanismo conocido con el nombre de unificación, introducido por J. A. Robinson en "A machine-oriented logic based on the resolution principle", Journal of the ACM, Vol. 12, pp. 23-41, 1965, y cuya patente US5222221 para el caso de los lenguajes lógicos concurrentes fue concedida a A. Hourí y E. Shapiro. Debido a la unificación de expresiones, que supone encontrar valores para las variables libres de las expresiones de manera que ambas expresiones sean idénticas, se obtiene un mecanismo de especialización más potente que el de los programas funcionales. Por otro lado, el trabajo de V. F. Turchin "The Concept of a Supercompiler", ACM TOPLAS, Vol. 8(3), pp. 292-325, 1986, relacionado con la supercompilación (compilación supervisada) de programas funcionales, juega también un papel importante dentro del desarrollo de la evaluación parcial. La supercompilación incluye un mecanismo conocido como conducción, que puede considerarse un método de transformación de funciones basado en el método de unificación. Concretamente, el método hace uso de un tipo de maquinaria muy similar al estrechamiento para construir espacios de estados (posiblemente infinitos) para un programa de entrada y una expresión a evaluar dados. Gracias al procedimiento de conducción, la supercompilación de programas funcionales permite obtener la misma cantidad de propagación de información (basada en la unificación) y especialización de programas que en la evaluación parcial de programas lógicos. Sin embargo, el proceso de conducción de Turchin puede ser infinito y no preserva el comportamiento operacional del programa, ya que puede extender el dominio de las funciones. En M. H. Sorensen y R. Glueck, "Generalization in Positive Supercompilation", Actas de ILPS'95, pp. 465-479, MIT Press, 1995, se estudian distintas técnicas para asegurar la terminación del mecanismo de supercompilación.

En la literatura no existen muchos trabajos relacionados directamente con la especialización de programas lógico-funcionales. G. Levi y F. Sirovich definieron en "Symbolic Evaluation and Logical Procedural Semantics", Springer LNCS Vol. 32, pp. 294-301, 1975, un método rudimentario de evaluación parcial, para el lenguaje funcional TEL, que usa un mecanismo de ejecución basado en la propagación bidireccional de parámetros mediante un procedimiento similar al estrechamiento, pero cuya terminación no está asegurada. Lo mismo puede afirmarse sobre el

método de J. Darlington y H. Pull propuesto en "A Program Development Methodology based on a Unified Approach to Execution and Transformation", Actas del symposium sobre Partial Evaluation and Mixed Computation, pp. 117-131, North-Holland, Amsterdam, 1988, donde se esboza un evaluador parcial para el lenguaje funcional HOPE (extendido con unificación) sin considerar los aspectos de control del proceso, terminación o preservación del comportamiento operacional de los programas de ordenador transformados.

Los fundamentos de la evaluación parcial de programas lógico-funcionales se introducen en M. Alpuente, M. Falaschi y G. Vidal, "Partial Evaluation of Functional Logic Programs", ACM Transactions on Programming Languages and Systems, Vol. 20(4), pp. 768-844, 1998. En este trabajo se sigue una aproximación similar al marco propuesto en J. W. Lloyd y J. C. Shepherdson, "Partial Evaluation in Logic Programming", Journal of Logic Programming, Vol. 1, pp. 217-242, 1991, para la evaluación parcial de los programas lógicos puros. En el método de M. Alpuente, M. Falaschi y G. Vidal, la especialización de programas integrados se basa directamente en el mecanismo operacional de estrechamiento que, debido a la propagación bidireccional de parámetros realizada a través de un proceso de unificación similar al de los lenguajes lógicos, permite obtener optimizaciones muy significativas. La especialización de programas basada en el mecanismo operacional de estrechamiento aporta una visión unificada de los mecanismos de ejecución y de transformación que explota la sinergia existente entre ambas componentes del lenguaje, funcional y lógica, con el beneficio de que las optimizaciones obtenidas quedan compiladas en el programa transformado. Este método constituye la primera aproximación completamente automática, correcta y finita para la especialización de programas lógico-funcionales integrados. Sin embargo, el método propuesto no puede aplicarse a los lenguajes lógico-funcionales con residuación, que sirven para programar procesos concurrentes y que representan el estado del arte más reciente en el campo técnico al que se refiere esta invención.

En la literatura sobre patentes no existen antecedentes del estado de la técnica en el campo específico de esta invención, ni para los lenguajes lógicos puros ni para los lenguajes lógico-funcionales integrados. En la patente U.S. 5241678 concedida a Y. Futamura y N. Kenroku, se revela un método de especialización de programas funcionales (sin predicados ni concurrencia) escritos en el lenguaje LISP que usa la información de un entorno operativo representado por una fórmula. Si dicha información permite demostrar la validez o inconsistencia de (alguna parte de) una condición de alguna de las reglas del programa, se simplifica dicha condición mediante un proceso de computación parcial, reemplazándola por el resultado obtenido.

La patente U.S.5586020 concedida a H. Isozaki enseña un método y sistema de transformación de programas que se aplica a la optimización de procesadores de lenguajes tales como un

compilador. Se diferencian dos niveles de optimización, uno local y el otro global. En el nivel local, se ejecutan las expresiones e instrucciones de asignación que están listas para su evaluación. En el nivel global, se simplifican algunos elementos sintácticos cuyo ámbito excede el de un simple bloque, como por ejemplo la eliminación de sub-expresiones comunes, tal y como se describe en A. V. Aho et al "Compilers: Principles, Techniques and Tools", Addison-Wesley, pp. 627-631 y 633-634. El proceso de especialización se desarrolla como sigue. En primer lugar, el sistema transforma el programa fuente a optimizar a una forma que permite su procesamiento. A continuación, este código intermedio se almacena en un fichero y es sometido a un proceso de análisis basado en el flujo de los datos. Sobre la base de este análisis, se ejecutan finalmente las optimizaciones y se genera el código objeto.

En la patente japonesa JP62008230 concedida a T. Fujioka, por otro lado, se especializa un programa para generar un conjunto de programas que desempeñan diferentes funciones y que se obtiene estableciendo físicamente las correlaciones entre partes de los datos de entrada y los procesos a los que se somete cada parte.

El objetivo de la presente invención es un método de especialización que mejora y generaliza los métodos convencionales mencionados anteriormente para integrar funciones, predicados y concurrencia. El método de especialización automática comprende tanto la preservación del comportamiento operacional del programa a especializar -es decir, el programa especializado produce exactamente las mismas soluciones que el original- como la efectividad de la transformación -es decir, el incremento de eficiencia para una gran parte de programas. Además, obtiene programas pertenecientes a la misma clase que los originales y que, por tanto, pueden ser ejecutados usando el mismo mecanismo y sistema de ejecución que los programas originales.

Otro objetivo de la invención es proporcionar un sistema de transformación automático, que es capaz de producir programas integrados optimizados que requieren menores tiempos de ejecución. Este objetivo se consigue mediante un sistema que procesa programas automáticamente y que comprende un medio de análisis sintáctico que somete un programa fuente en memoria a un proceso de análisis de la sintaxis y generación de un código intermedio en un formato predeterminado, y un medio propiamente de optimización que somete el código intermedio a un proceso de evaluación parcial automática utilizando información de un entorno operativo representado por una estructura de datos de tipo lista. El proceso global conduce a un código objeto en memoria con el menor tamaño y mayor eficiencia posibles, y a unos datos de salida acordes con el código generado.

Descripción de la invención

El objeto mencionado anteriormente se alcanza por el asunto objeto de la reivindicación 1, que define un método y sistema para la transformación automática de programas lógico-funcionales con residuación, durante el cual se computa automáticamente un programa especializado

a partir de un programa en memoria y un conjunto de datos introducidos a través de un dispositivo interactivo de entrada.

De conformidad con ello, el objetivo principal de esta invención es proporcionar un método y sistema nuevos para derivar programas optimizados que se ejecuten más eficientemente que los programas originales usando una información parcial sobre los datos de entrada.

Otro objetivo primordial del método de acuerdo con esta invención es preservar el comportamiento operacional con respecto a las computaciones que terminan con éxito, es decir, el programa original y el especializado deben computar las mismas soluciones. Por otra parte, también se preserva el comportamiento con respecto a los bloqueos, es decir, las expresiones que suspenden en el programa original son las mismas que en el transformado.

Otra ventaja del método propuesto es el hecho de que el propio método se puede utilizar para demostrar la propiedad de ausencia de bloqueos para una clase de objetivos en un programa dado. Concretamente, en el programa especializado es suficiente comprobar que, tras la especialización, ningún símbolo de función tiene asociada una anotación rígida en el programa resultante.

Un objetivo adicional de esta invención es proporcionar un método que permita simplificar el comportamiento dinámico del programa. En particular, cuando existen ciertas expresiones que pueden suspender, nuestro método permite evaluar (en tiempo de especialización) otras expresiones que se encuentran activas, asignándose así un valor a las variables de los argumentos de entrada de las funciones, con lo que se activarán las computaciones suspendidas consiguiendo de este modo una optimización muy significativa.

Descripción de los dibujos

Una descripción más completa de la invención y de muchas de las ventajas esperables de la misma se obtendrá fácilmente a medida que se comprenda mejor el sistema propuesto con referencia a la siguiente descripción detallada, cuando se considere con respecto a los dibujos anexos, en los cuales:

La Figura 1 es un diagrama de bloques que ilustra un sistema automático de transformación de programas configurado para realizar la optimización global.

Dicho sistema comprende un medio de análisis sintáctico que somete un programa fuente en memoria a un proceso de análisis de la sintaxis y generación de un código intermedio en un formato predeterminado, y un medio de optimización que somete el código intermedio a un proceso de optimización utilizando información de un entorno operativo representado por una estructura de datos de tipo lista y que conduce finalmente a un código objeto en memoria con el menor tamaño y mayor eficiencia posibles, y a unos datos de salida acordes con el código generado.

El proceso comienza con la introducción, por parte del usuario, del programa a transformar (el cual es almacenado en memoria) y de los datos con respecto a los cuales se va a especializar dicho programa (introducidos a través de una interfaz interactiva). A continuación, se realiza un análisis

sintáctico del programa para transformarlo a un formato interno que se almacena en un fichero. El fichero almacenado es empleado posteriormente por el módulo de optimización para realizar el proceso de especialización propiamente dicho, el cual genera dos salidas: el programa especializado (el cual se almacena en memoria) y los datos de entrada transformados (renombrados) para que se puedan ejecutar en el programa especializado.

La Figura 2 es un esquema que ilustra los componentes estructurales del optimizador. El optimizador incluye una etapa de inicialización que obtiene, a partir de la lista de datos de entrada, un conjunto de llamadas a función (o expresiones) a especializar. En la etapa siguiente se construye, a partir del conjunto de llamadas, un conjunto de reglas de programa, llamadas resultantes. A continuación, se inspeccionan los resultantes que acaban de ser generados, con el fin de determinar si éstos contienen alguna llamada a una función que no queda cubierta suficientemente por las definiciones de función expresadas en los resultantes. Si existen llamadas que no están cubiertas, éstas se incluyen en el conjunto de llamadas que deben ser especializadas y se repite el proceso de construcción de los resultantes, y su posterior inspección, tal y como aparece en el diagrama de la Figura 2.

Para simplificar la descripción, omitimos el detallar la forma en que se debe extender el conjunto actual de llamadas con la información contenida en los resultantes de cada iteración. El proceso termina cuando coincide el conjunto extendido de llamadas con el conjunto inicial de la iteración correspondiente. El programa transformado se obtiene entonces como el conjunto de resultantes (o evaluación parcial) de las expresiones del conjunto final de llamadas.

La Figura 3 es un esquema de flujo que ilustra la construcción de las reglas llamadas resultantes. Esto comprende una fase de selección de uno de los elementos del conjunto de llamadas, seguido de una fase de formación de los resultantes asociados a dicha expresión. Estas fases se repiten hasta que todas las llamadas del conjunto han sido procesadas.

La Figura 4 muestra esquemáticamente la manera de formar los resultantes asociados a cada llamada individual especializada. El primer paso es obtener una derivación o secuencia (posiblemente parcial) de pasos de ejecución para la llamada considerada, de conformidad con el mecanismo operacional propio de los lenguajes lógico funcionales con residuación ya descrito en los antecedentes de la invención. Distinguimos dos tipos de derivaciones:

1. derivaciones suspendidas, en las cuales el paso inicial para la expresión considerada es incompleto, es decir, la expresión a evaluar queda suspendida debido a alguna función rígida que no está lo suficientemente instanciada;
2. derivaciones completas, es decir, aquéllas que no involucran ningún tipo de suspensión.

En el caso en el que la derivación sea completa,

se forma el resultante asociado y el proceso termina. Por ejemplo, dada la expresión $f(X,Y)$ y la derivación completa $D = (f(X,Y), 2, \{X/0, Y/1\})$, el resultante asociado es la regla: $f(0,1) \rightarrow 2$.

En caso de que sea incompleta, se procede a completar el último paso ignorando las anotaciones de evaluación y se aplica el proceso que se detalla en la Figura 5.

La Figura 5 ilustra el esquema de construcción de las listas de sustituciones anotadas que se usan para formar los resultantes asociados a las derivaciones incompletas. El proceso toma como entrada la derivación D cuyo paso final acaba de completarse siguiendo el método descrito en la Figura 4. A continuación, se construye la lista L de sustituciones, obtenidas como composición funcional de las sustituciones que se hayan obtenido de forma consecutiva en D y que, además, provengan de la evaluación de funciones con la misma anotación (flexible o rígida). Dicha lista L se usa para guiar el proceso de construcción de los resultantes asociados como sigue. Concretamente, se crean tantas reglas intermedias (usando símbolos de función nuevos cuyos argumentos son variables nuevas y distintas entre sí) como sustituciones haya en la lista L , de forma que la anotación de evaluación asociada a cada regla viene determinada por la marca de evaluación de la correspondiente agrupación de sustituciones. Así, las marcas de suspensión producen funciones nuevas que se anotan como rígidas, mientras que las de no suspensión producen funciones flexibles. Por último, se renombran las expresiones de cada regla mediante el proceso que se detalla en la Figura 6. Por ejemplo, dada la expresión $f(X,g(Y,h(Z)))$ y la derivación incompleta $D = (f(X,g(Y,h(Z))), \{a,g(Y,h(Z)), \{X/a\}\})$, el método procede a completarla, obteniendo así la nueva derivación completa: $D' = (f(X,g(Y,h(Z))), f(a,g(b,c)), \{X/a, Y/b, Z/c\})$. A continuación, se forma la correspondiente agrupación de sustituciones anotadas $L = [\{X/a\}^{flex}, \{Y/b\}^{rigid}, \{Z/c\}^{flex}]$, de forma que los resultantes generados son:

$$\begin{aligned} f'(a,Y,Z) &\rightarrow f_1(Y,Z), \\ f'_1(b,Z) &\rightarrow f'_2(Z), \\ f'_2(C) &\rightarrow \text{ren}(f(a,g(b,c))), \end{aligned}$$

donde el renombramiento utilizado es:

$$\text{ren}(f(X,g(Y,h(Z)))) = f'(X,Y,Z).$$

La Figura 6 define el método de renombramiento para las llamadas especializadas. El proceso comienza con el conjunto de expresiones S que, mientras no sea vacío, se procesa como sigue. Se selecciona un elemento del conjunto y se asocia a éste una expresión de llamada, encabezada por un símbolo de función nuevo cuyos argumentos son variables diferentes. Por ejemplo, como veíamos en la descripción de la figura anterior, el renombramiento para la expresión $f(X,g(Y,h(Z)))$ es:

$$\text{ren}(f(X,g(Y,h(Z)))) = f'(X,Y,Z)$$

donde f' es un símbolo de función nuevo y $X, Y,$

Z son las variables diferentes de la expresión que pretendemos renombrar. Este proceso se repite para cada uno de los elementos de S , formando así un conjunto de expresiones renombradas correspondientes S' , que se utiliza para la operación que se describe en la Figura 7.

La Figura 7 enseña cómo debe realizarse el renombramiento de resultantes usando un conjunto de llamadas S , su correspondiente renombramiento S' (computado según el método de la Figura 6) y una expresión de llamada. Básicamente, se procede como sigue: si la expresión es una variable o una constante, se devuelve la expresión misma; si se trata de un término encabezado por un símbolo constructor, se devuelve un término encabezado por el mismo símbolo pero renombrando consecuentemente cada uno de sus argumentos; y, por último, si el término está encabezado por un símbolo de función definido en el programa, entonces se elige una pareja de términos $\langle s, s' \rangle$ (original y renombrado) de S' , de forma que el término s sea más general que la expresión de entrada. En este último caso, se devuelve el término renombrado s' , tras haberle aplicado la sustitución correspondiente, también renombrada. Por ejemplo, dado el renombramiento $S' = \{ \langle f(c,X), f'(X) \rangle, \langle g(b,h(Y)), g'(Y) \rangle \}$, el término $f(c,g(b,h(a)))$ se renombra como $F(g'(a))$.

Aplicación industrial

Una de las características comunes de las aplicaciones industriales de las técnicas de transformación mediante evaluación parcial es la posibilidad de conseguir implementaciones eficientes, por métodos puramente automáticos, de programas generales y altamente paramétricos. Una de las áreas en las que se ha conseguido un mayor nivel de optimización es la de los gráficos por ordenador. Concretamente, existen aplicaciones gráficas en las que se computa repetidamente información sobre la forma en que los rayos de luz atraviesan una escena dada desde diferentes orígenes y en diferentes direcciones. La especialización de una aplicación de estas características para una escena fija ha dado lugar a mejoras en el tiempo de ejecución de entre un 150 % y un 300 %. Otra área en la que se han obtenido mejoras muy significativas es la de las aplicaciones de bases de datos. En este caso, cada vez que se realiza una modificación sobre una base de datos, el gestor de la misma debe comprobar que no hayan sido violadas las restricciones de integridad. Puesto que las restricciones de integridad no suelen modificarse durante toda la vida de la base de datos, es posible especializar el procedimiento de comprobación de las restricciones de integridad con respecto a las mismas. De esta forma, ha sido posible acelerar el proceso entre un 500 % y un 10000 %. Otras aplicaciones en las que se han empleado con éxito las técnicas de evaluación parcial son: reconocimiento de patrones, redes neuronales, hojas de cálculo, aplicaciones científicas, simulación, etc.

REIVINDICACIONES

1. Un sistema automático de transformación de programas lógico-funcionales con residuación **caracterizado** porque comprende un medio de análisis sintáctico que somete un programa fuente en memoria a un proceso de análisis de la sintaxis y generación de un código intermedio en un formato predeterminado, y un medio de optimización que somete el código intermedio a un proceso de optimización utilizando información de un entorno operativo representado por una estructura de datos de tipo lista y que conduce a un código objeto en memoria con el menor tamaño y mayor eficiencia posibles, y a unos datos de salida acordes con el código generado.

2. Un método de optimización **caracterizado** porque evalúa el conjunto de llamadas que aparecen en las expresiones a especializar y extrae de los resultantes obtenidos aquellas llamadas que no quedan cubiertas por el conjunto de funciones especializadas; a continuación, evalúa de la misma

forma las llamadas a función que son introducidas en cada iteración, hasta que la repetición del proceso no provoca la introducción de nuevos elementos en el conjunto.

3. Método según la reivindicación 2 **caracterizado** además porque mantiene la equivalencia operacional entre el programa original y el especializado, es decir, que preserva las soluciones computadas.

4. Método según la reivindicación 3 **caracterizado** además porque sintetiza adecuadamente las anotaciones de evaluación para las funciones especializadas, preservando por tanto el comportamiento respecto a las suspensiones del programa original.

5. Método según la reivindicación 4 **caracterizado** además porque obtiene programas pertenecientes a la misma clase que los originales y que, por tanto, pueden ser ejecutados usando el mismo mecanismo y sistema de ejecución que los programas originales.

FIGURA 1

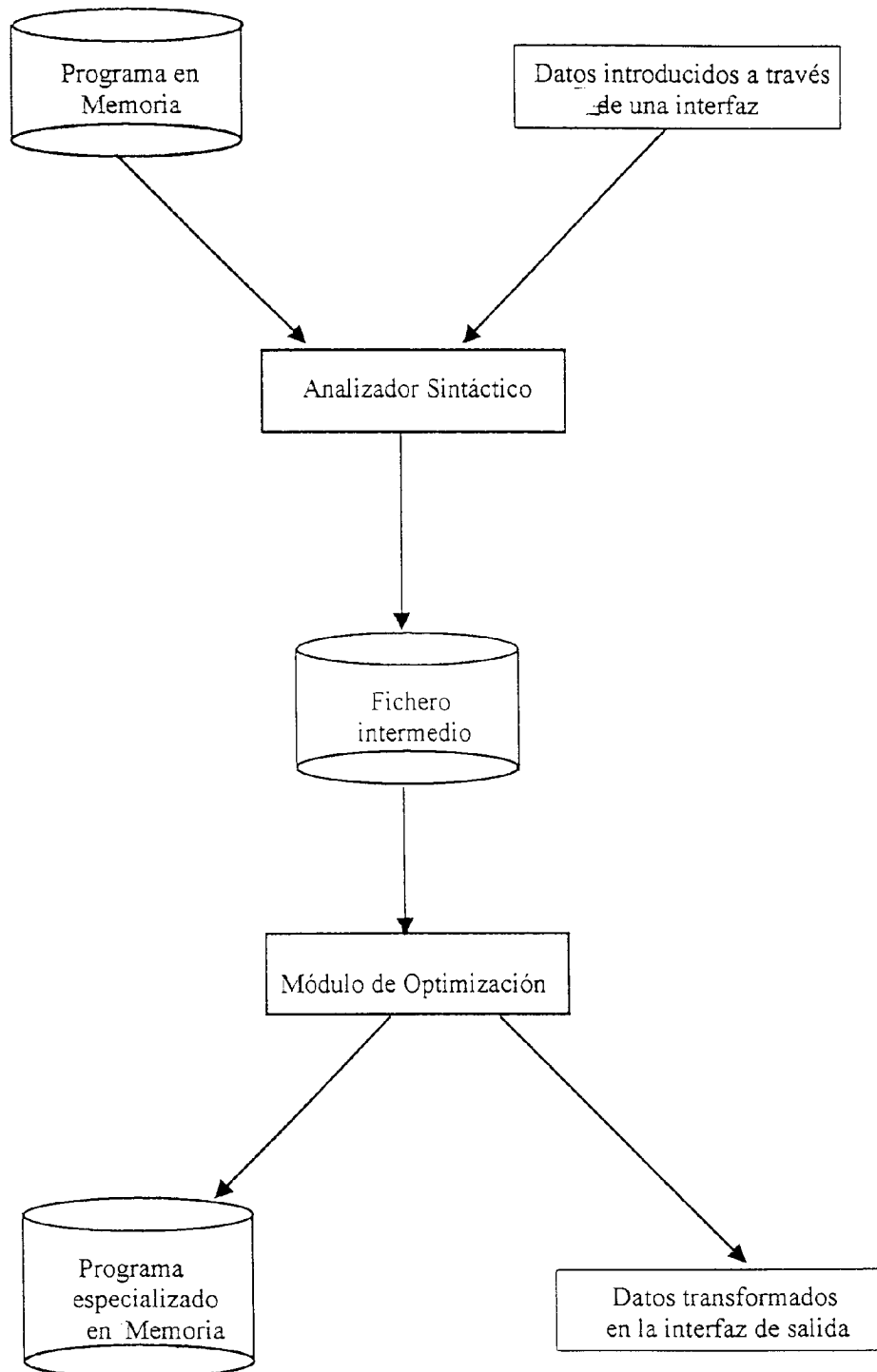


FIGURA 2

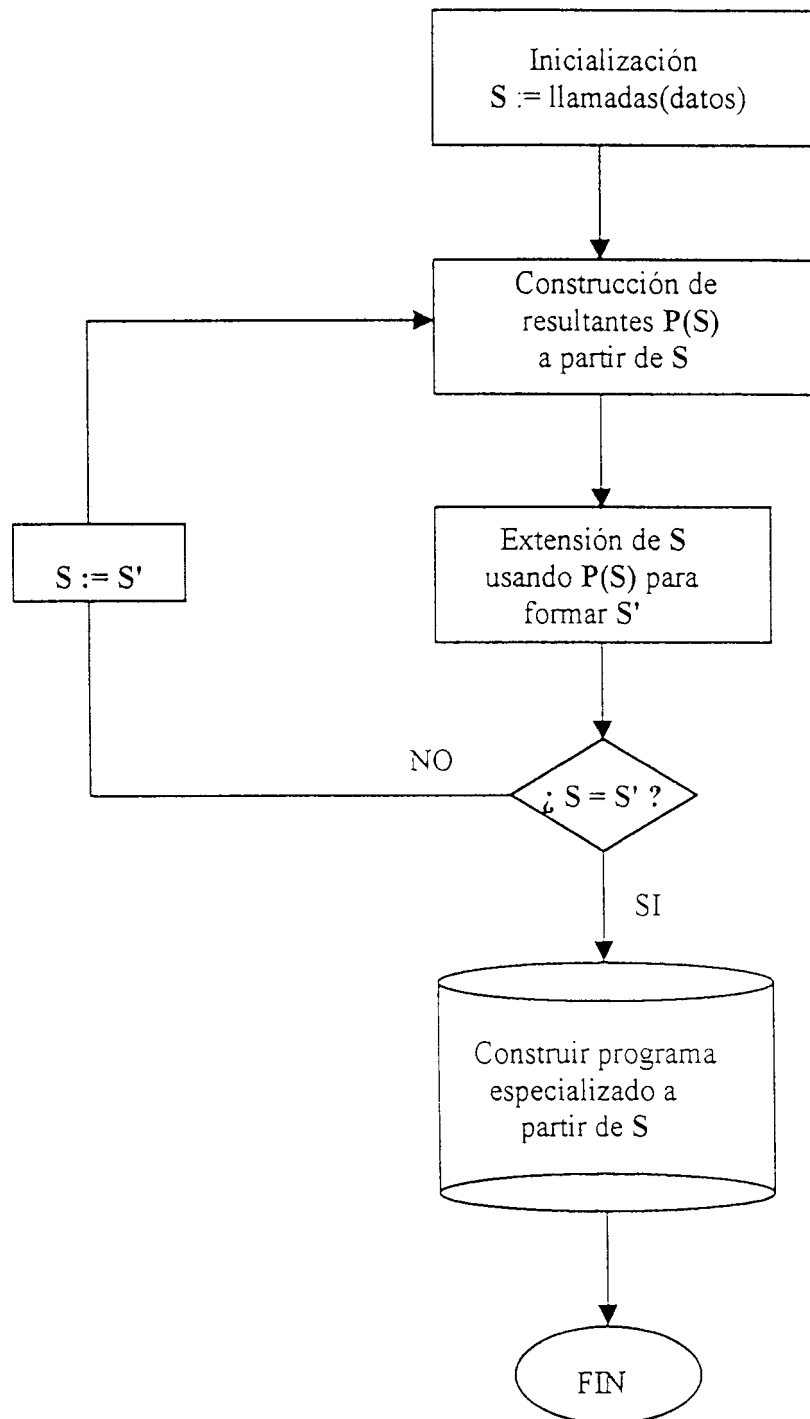


FIGURA 3

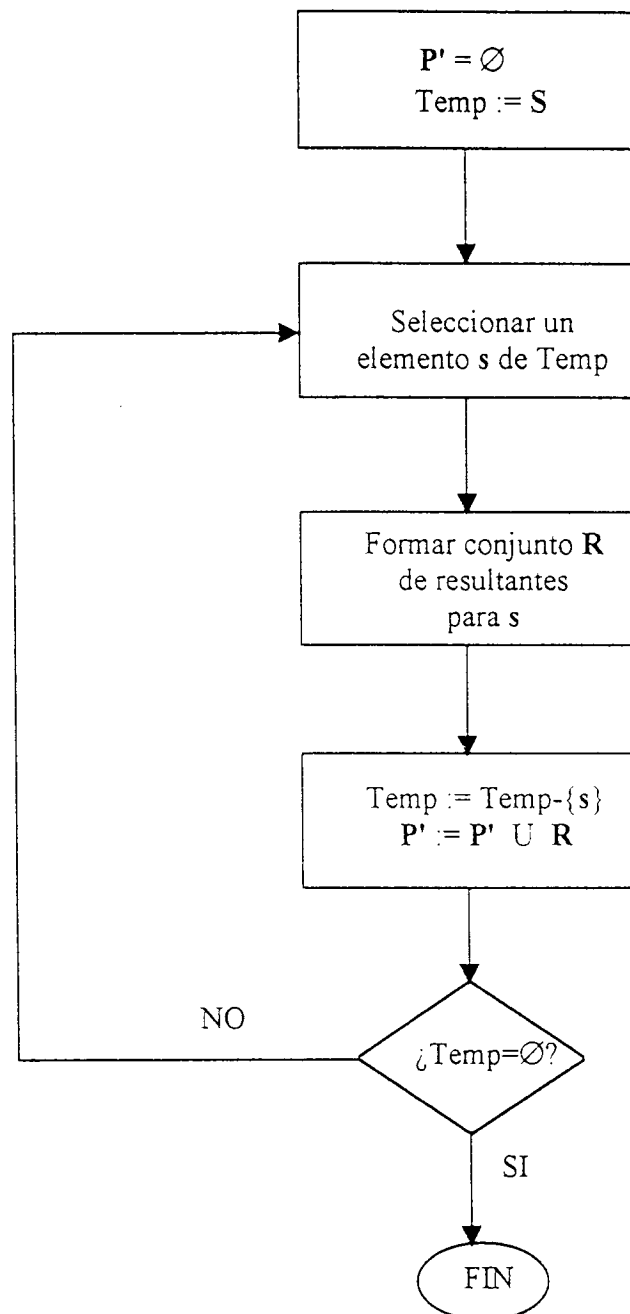


FIGURA 4

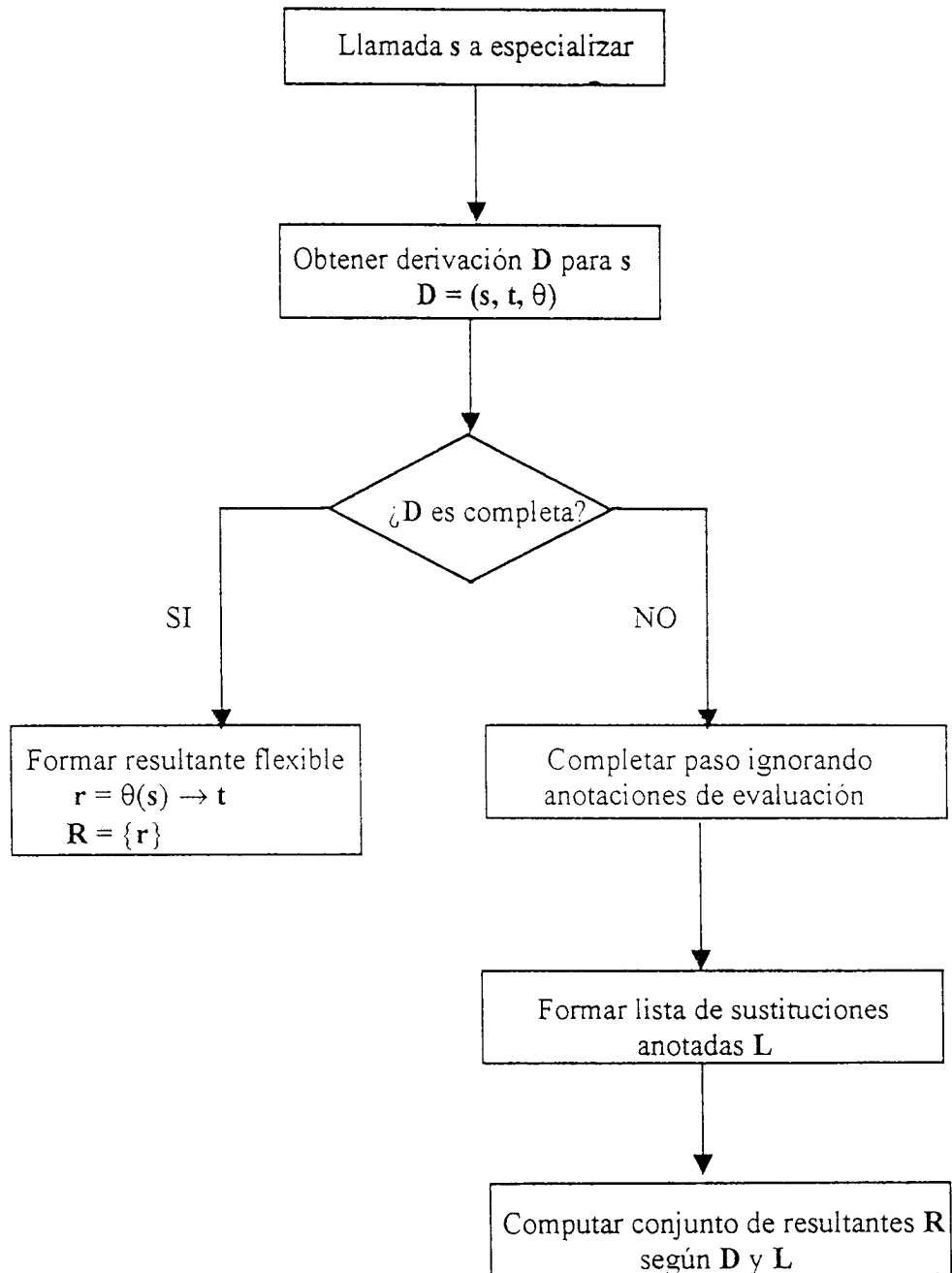


FIGURA 5

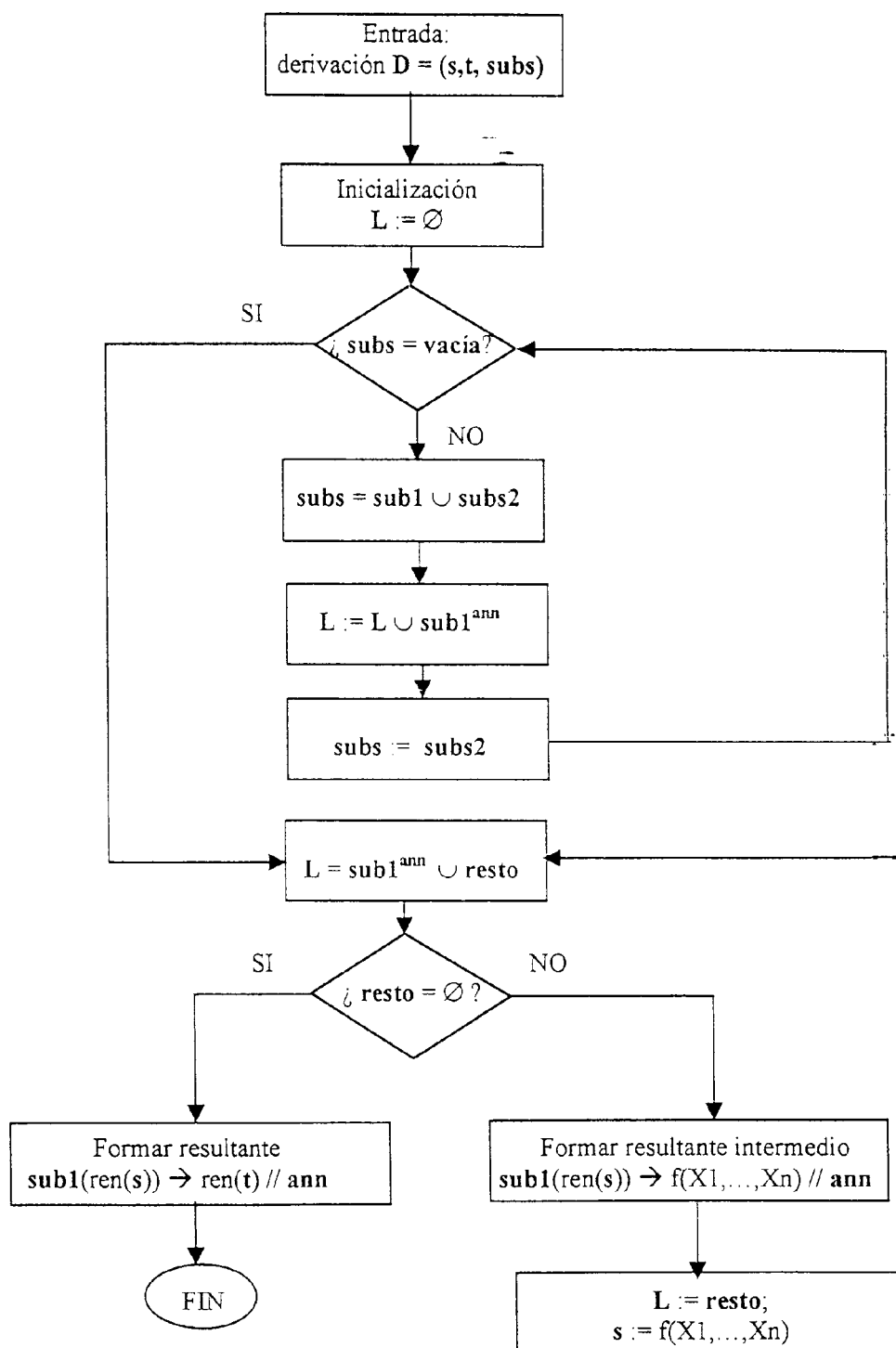


FIGURA 6

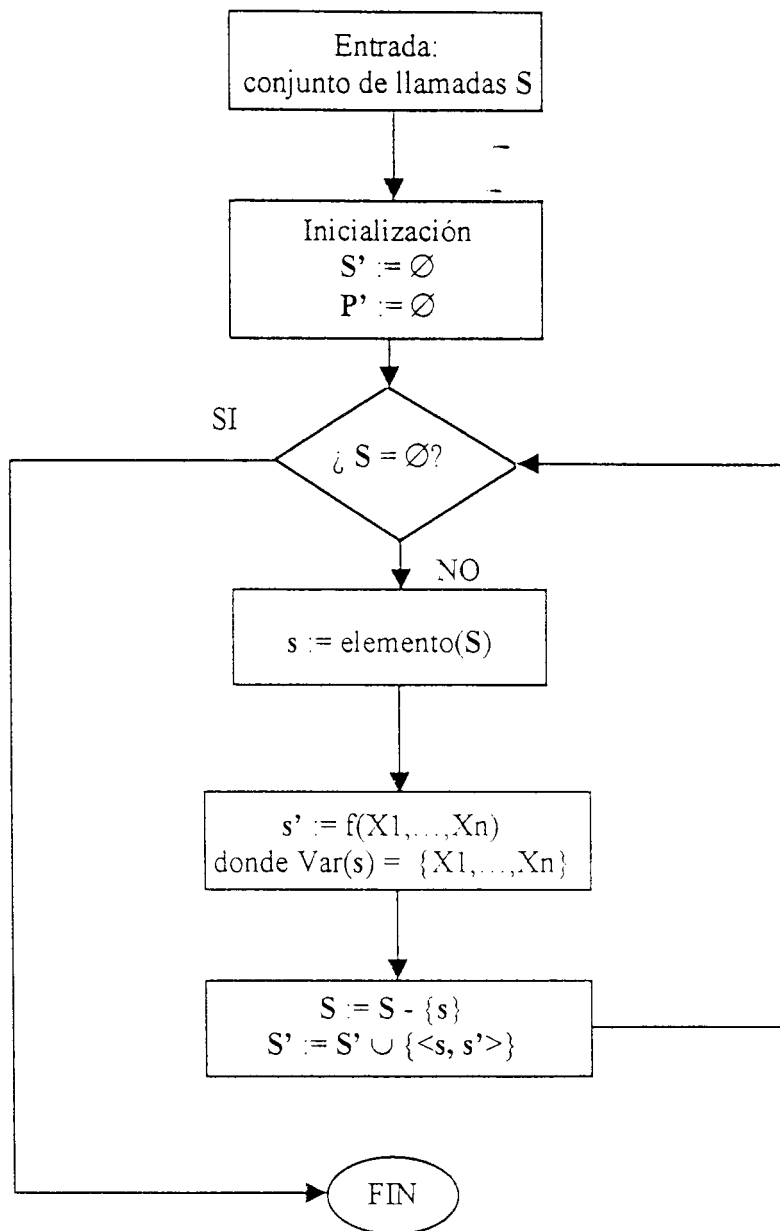
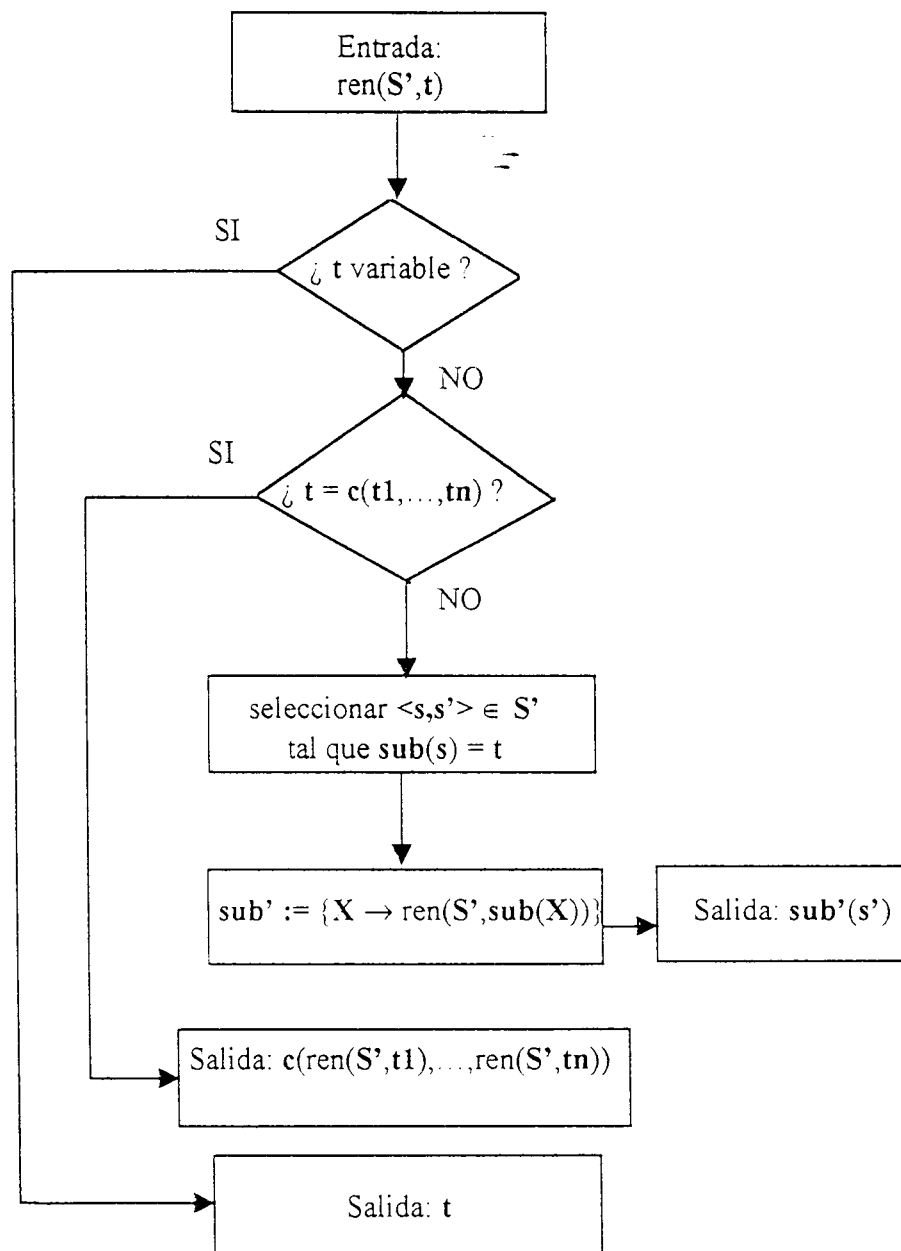


FIGURA 7





OFICINA ESPAÑOLA
DE PATENTES Y MARCAS
ESPAÑA

- ⑪ ES 2 166 279
⑫ N.º solicitud: 009902055
⑬ Fecha de presentación de la solicitud: 09.09.1999
⑭ Fecha de prioridad:

INFORME SOBRE EL ESTADO DE LA TECNICA

⑮ Int. Cl.⁷: G06F 9/06, 9/30

DOCUMENTOS RELEVANTES

Categoría	Documentos citados	Reivindicaciones afectadas
A	US 5586020 A (ISOZAKI, H.) 17.12.1996, todo el documento.	1-5
A	US 5222221 A (HOURI, A. et al.) 22.06.1993, columnas 1-8; reivindicaciones.	1,2
A	US 5241678 A (FUTAMURA, Y. et al.) 31.08.1993, todo el documento.	1,2
A	BASE DE DATOS PAJ de JPO. JP 08-249193 A (HITACHI LTD.; HITACHI SOFTWARE ENG. CO. LTD.) 27.09.1996, resumen.	1
A	BASE DE DATOS PAJ de JPO. JP 04-278633 A (HITACHI LTD.) 05.10.1992, resumen.	1

Categoría de los documentos citados

X: de particular relevancia

Y: de particular relevancia combinado con otro/s de la misma categoría

A: refleja el estado de la técnica

O: referido a divulgación no escrita

P: publicado entre la fecha de prioridad y la de presentación de la solicitud

E: documento anterior, pero publicado después de la fecha de presentación de la solicitud

El presente informe ha sido realizado

☒ para todas las reivindicaciones

☐ para las reivindicaciones n.º:

Fecha de realización del informe

26.02.2002

Examinador

O. González Peñalba

Página

1/1