Fundamentos de Inteligencia artificial

Mecanismo de inferencia tipo OPS5 para un esquema de representación basado en redes de Petri (1)

J. A. Bañares, P. R. Muro-Medrano, J. L. Villarroel
Departamento de Ingeniería Eléctrica e Informática, Universidad de Zaragoza

1. Introducción

Los sistemas de eventos discretos (SED) han constituido uno de los más relevantes dominios de aplicación de los sistemas basados en reglas (SBAR). Sin embargo, los SBR tradicionales carecen de recursos específicos para modelar aspectos importantes en los SED como paralelismo, concurrencia, sincronización, etc., así como adolecen de la perspectiva y la posibilidad de un análisis más formal, que resulta muy útil cuando los sistemas a representar adquieren cierta complejidad. El interés por crear un esquema de representación apropiado para la representación de sistemas de eventos discretos y de una metodología de soporte para la generación de modelos nos condujo a la definición de KRON (Knowledge Representation Oriented Nets) [Muro89] , un esquema de representación del conocimiento para sistemas de eventos discretos que integra frames, programación orientada a objeto y redes de Petri de alto nivel. Se eligió el formalismo de las redes de Petri de alto nivel [Jensen91] (RAN) porque ha demostrado ser adecuado para la representación y análisis del comportamiento dinámico de sistemas concurrentes discretos.

En el presente trabajo nos centramos en la descripción de un mecanismo de inferencia eficiente desarrollado para KRON. Nuestra aproximación está basada en la similitud entre el mecanismo de interpretación de las RAN y el mecanismo de inferencia de los lenguajes basados en reglas. En ambos casos la mayor fuente de ineficiencia es el proceso de *correspondencia de patrones simbólicos* [2]. El algoritmo de RETE [Forgy82] fue desarrollado para dotar a OPS5 [Browston85] de un eficiente mecanismo de inferencia reduciendo el número de operaciones de correspondencia en cada ciclo. La conveniencia de utilizar OPS5 para la implementación del mecanismo de interpretación de RAN ha sido apuntada en trabajos como [Bruno86] y [Duggan88]. Sin embargo, las restricciones impuestas por la utilización de RAN, permite mejorar la eficiencia sobre una implementación directa sobre un lenguaje como OPS5. Se propone una implementación concreta que refleja algunos aspectos, como la reducción del árbol de test en el proceso de correspondencia de patrones, o las peculiaridades derivadas de aspectos como la concurrencia del modelo en la fase de ejecución del ciclo de inferencia en KRON. El presente trabajo se estructura de la siguiente forma: en el apartado §2 se ilustra el mecanismo de Inferencia de OPS5 y los conceptos del algoritmo de RETE. En §3 se muestran las ideas básicas de KRON. En §4 se explica el mecanismo de interpretación de KRON, detallando las fases de reconocimiento, selección y ejecución del ciclo de inferencia. Finalmente en el apartado §5 se comentan aspectos de eficiencia.

2. Ilustración del mecanismo de Inferencia de OPS5

Un sistema basado en reglas consta básicamente de tres componentes: Una *Memoria de Trabajo* que contiene símbolos que representan hechos y aserciones utilizados para resolver un problema, *Reglas* que contienen el conocimiento del dominio del problema, y un *Motor de Inferencia* que selecciona una regla entre las que se adecúan a la configuración de datos y la ejecuta. El ciclo de control usado por los motores de inferencia se denomina ciclo *reconocimiento/actuación* y consta de los siguientes pasos: *Reconocimiento, selección y ejecución.*

En cada ciclo de reconocimiento/actuación de un motor de inferencia se debe comprobar repetidamente si la parte precondición de cada una de las reglas es satisfecha por los datos de la memoria de trabajo para determinar que reglas son ejecutables (las reglas ejecutables forman el denominado *conjunto conflicto*). El algoritmo de RETE fue desarrollado con el objetivo de reducir el número de operaciones de correspondencia en cada ciclo reconocimiento/actuación. La persistencia de información entre los ciclos de inferencia se denomina *redundancia temporal*.. RETE se aprovecha de la redundancia temporal evitando los cálculos de correspondencia de patrones realizados en ciclos anteriores con datos que no han cambiado. RETE se beneficia también de la *similitud estructural* entre reglas, evitando repetir aquellas correspondencias de patrones que son iguales en diferentes reglas.

En RETE cada ciclo recoge los cambios en la memoria de trabajo como entrada y produce cambios en el conjunto conflicto como salida, eliminando la necesidad de recalcular totalmente el conjunto conflicto. Para cumplir este objetivo RETE debe guardar la información relativa a correspondencias previas para evitar repetirlas. Para ello asocia a cada condición de la regla una lista que recoge los elementos de la memoria de trabajo que se corresponden con la condición. Estas listas son asociadas a nodos que representan la condiciones de la regla y a su vez los nodos son enlazados en una red. La red que contiene la información del proceso de correspondencia consta de dos clases de nodos:

· *Nodos de una entrada*: cada una de las condiciones de una regla es trasladada a una secuencia de test sobre el valor de diferentes atributos. Cada test es representado por un nodo de una entrada. A cada cadena de test de una entrada que representa una condición de la regla se le denomina *árbol de test*.

· *Nodos de dos entradas*: los nodos de dos entradas tratan los test necesarios para encontrar los datos de la memoria de

trabajo que son consistentes con todas las condiciones de la regla. La salida de un nodo de dos entradas es conectada al nodo de dos entradas que representa la siguiente condición de la regla. La otra entrada de un nodo de dos entradas es la salida de un árbol de test. Una cadena de nodos de dos entradas la denominaremos *árbol de consistencia* [3].

Sólo las listas asociadas con las condiciones que se corresponden con elementos de la memoria de trabajo que han cambiado son actualizadas, mientras el resto permanece igual. Sólo las reglas con condiciones cuyas listas han sido actualizadas son tenidas en cuenta para determinar el actual conjunto conflicto. Por último, una de las reglas del conjunto es seleccionada para ser ejecutada.

3. KRON

En KRON el comportamiento dinámico de las entidades se articula mediante tres clases de objetos: *objetos de estado*, *transiciones* y *marcas*:

· Los objetos de estado centralizan la información relacionada con una entidad dinámica. El estado se representa mediante el valor de una serie de atributos que se corresponden con los lugares de una RAN. Por otra parte existen referencias a cada una de las actividades que pueden cambiar el estado de la entidad.

· Los objetos transición definen las posibles actividades de una entidad. Equivalen a las transiciones de una RAN. Todo objeto transición tiene unas precondiciones y postcondiciones que quedan definidas por las relaciones entre los objetos transición y los atributos lugar de los objetos de estado y que equivalen a los arcos de una RAN.

· Las marcas representan entidades pasivas al nivel de abstracción considerado. El estado del sistema es definido mediante marcas en los atributos lugar de los objetos de estado. Desde el punto de vista de la RAN equivalen a las marcas que evolucionan en la red.

La composición de dichos objetos permite obtener de forma directa una RAN subyacente al modelo global. La interpretación de los modelos se basa en la interpretación de la RAN subyacente. En la **fig. 1** se muestra una red KRON sencilla. La red está compuesta por cinco lugares y una transición que representa la carga de un conjunto de maquinas. Los arcos están etiquetados por expresiones que son listas de pares atributo-variable. Cada par define posibles *ligaduras* entre una variable y los valores de los atributos de los objetos de marcado que se encuentran en el correspondiente lugar de entrada. El conjunto de todas las variables incluidas en las expresiones de arco es asociado a la transición.

Por ejemplo, la expresión de arco {name <part> machine<machine> tool <tool>} en la transición LOAD permite tres diferentes ligaduras para las variables <part>, <machine> y <tool> para el marcado presente: (<part>=P1;<machine>=M1;<tool>=T1), (<part>=P3;<machine>=M3;<tool>=T1) y (<part>=P11; <machine>=M2;<tool>=T5).

Una transición está sensibilizada si existe una ligadura consistente para cada una de sus variables asociadas. Cada una de las posibles ligaduras consistentes define un diferente *modo de disparo*. En el ejemplo de la **figura 1** la transición está sensibilizada por (<part>=P1;<machine>=M1;<tool>=T1).

Una red KRON puede ser interpretada como un SBR donde las transiciones tienen el papel de reglas y las marcas juegan el papel de elementos de la memoria de trabajo. La expresión de una transición KRON en una regla OPS5 se puede hacer fácilmente teniendo en cuenta que el proceso de disparo de una transición en una RAN implica retirar las marcas, que han producido la ligadura consistente, de los lugares de entrada a

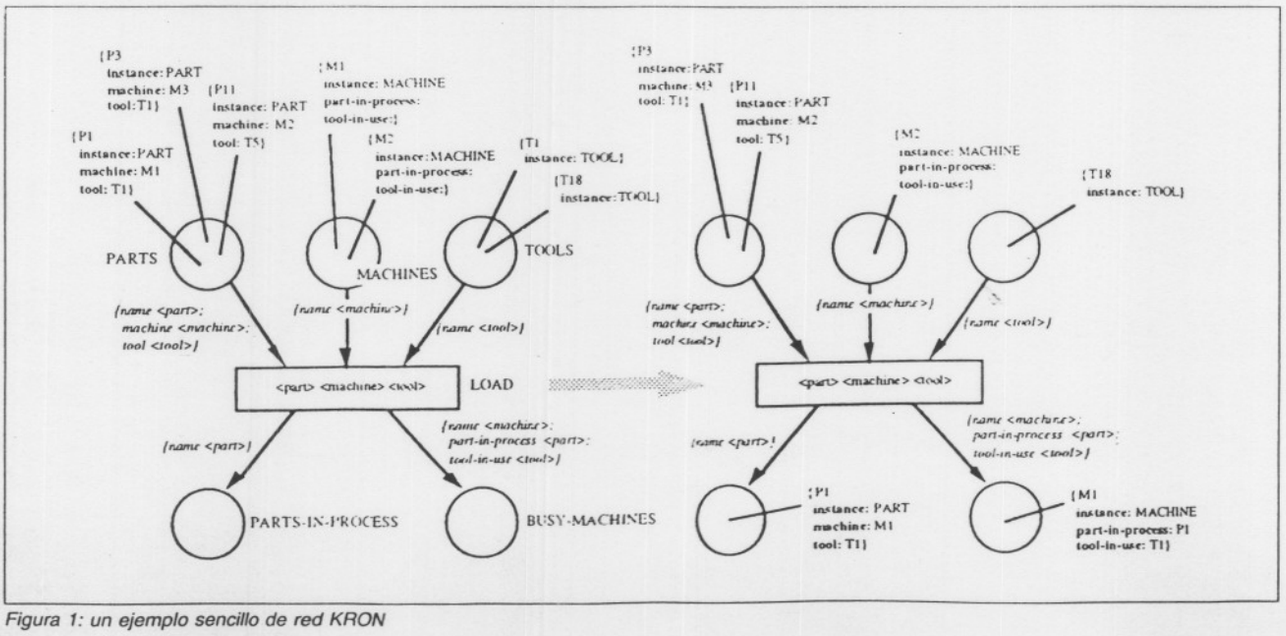


Figura 1: un ejemplo sencillo de red KRON

la transición y poner marcas en los lugares de salida de la transición. De esta forma una transición de una red KRON se traduce en una regla OPS5 en la que la parte precondición de la regla se compone con las expresiones que etiquetan los arcos de entrada, y la parte conclusión recoge la actualización de marcado debida a las condiciones de arco de entrada y salida de la transición. En cuanto a la expresión de las marcas, como elementos de la memoria de trabajo, hay que añadir el atributo de lugar para indicar donde se encuentra la marca. La siguiente sección muestra como se ha implementado el mecanismo de inferencia de KRON adaptando el algoritmo de RETE a las peculiaridades de las RAN.

4. Mecanismo de Inferencia en KRON

La **fig. 2** muestra el ciclo de inferencia de KRON. La fase de reconocimiento se realiza cada vez que se añade o se extrae una marca de un atributo del objeto de estado. Veremos como el proceso de correspondencia de patrones se puede beneficiar de la partición de la memoria de trabajo que ofrece la estructura de la RAN. El disparo de transiciones equivale a la fase de ejecución de un SBR. Las fases de selección y disparo deben tener en cuenta la concurrencia inherente a las RAN, pudiendose disparar más de un modo de disparo en cada ciclo.

4.1. Fase de reconocimiento

Las principales diferencias entre SBR y RAN están relacionadas con la memoria de trabajo [Valette90, Bañares93]. Un SBR tiene una única memoria de trabajo para todas las reglas, mientras que una RAN tiene su memoria de trabajo dividida en lugares. Las precondiciones de una transición sólo se aplican a las marcas de sus lugares de entrada. Esta partición produce los siguientes efectos sobre el árbol de test:

- El nodo raíz es dividido en diversos nodos raíz. Cada lugar define una memoria de trabajo y por lo tanto se define un nodo raíz por cada lugar.

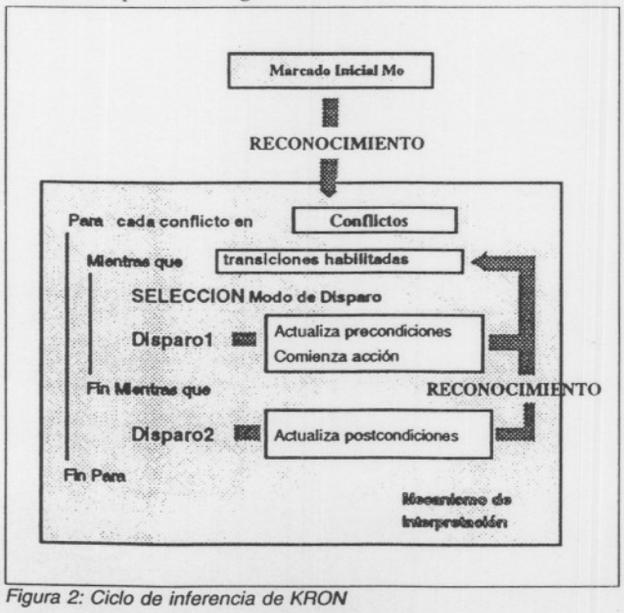


Figura 2: Ciclo de inferencia de KRON

- El árbol de test se reduce porque no son necesarios los test de clase y lugar. Los test de lugar están implícitos en la estructura de la RAN, y cada lugar tiene asociado un tipo de marca que puede contener.
- El uso de la similitud estructural no resulta tan interesante. La similitud estructural permite compartir partes de las ramas del árbol de test cuando existen los mismos test en diferentes precondiciones. Con excepción de los lugares que son compartidos por muchas transiciones, las ramas del árbol de test vienen de diferentes nodos. Este hecho hace imposible compartir nodos.

Si no se hace uso de la similitud estructural, el árbol de test obtenido es un conjunto de ramas independientes que parten de los nodos raíz. De esta forma cada rama se puede reducir a un único nodo que representa a todos los test de una expresión de arco. El nodo memoriza todas las ligaduras que resultan de la correspondencia de la expresión de arco con las marcas en los lugares previos.

Existe otra importante diferencia entre el algoritmo de RETE y el mecanismo de inferencia de KRON. En el algoritmo de RETE cuando un elemento es eliminado de la memoria de trabajo todas las referencias a ese elemento que se encuentren en el árbol de test y en el árbol de consistencia deben ser eliminadas. El mecanismo utilizado por el algoritmo de RETE para encontrar estas referencias es repetir el proceso de correspondencia como si este elemento fuera de nuevo añadido a la memoria de trabajo. Cuando se alcanza un nodo se procede a la extracción de la referencia buscada de la lista de referencias memorizadas en cada nodo. Esta fuente de ineficiencia no es muy importante en un SBR debido a que sacar un elemento de la memoria de trabajo no es una acción que se repita sistemáticamente. Sin embargo en RAN cada disparo de transición implica sacar marcas de los lugares de entrada. Por esta razón en KRON cada ligadura tiene un enlace con las ligaduras que la generan mediante el test de consistencia y con las ligaduras generadas al hacer el test de consistencia con otras ligaduras. Las marcas están también enlazadas con las ligaduras que generan. De esta forma se permite una rápida actualización de la información ya que se evitan la repetición del proceso de correspondencia y las búsquedas.

En KRON, como consecuencia de la partición de la memoria de trabajo y de no utilizar la similitud estructural, se asocia un árbol a cada transición. Este árbol es la adaptación de los árboles de test y consistencia de RETE a las peculiaridades del proceso de correspondencia de patrones en RAN. La **fig. 3** muestra el árbol asociado a la transición LOAD.

4.2. Fase de selección y disparo

Los modelos realizados con redes de Petri pueden no ser totalmente deterministas, por lo que es necesario seleccionar qué transiciones van a ser disparadas. Existen conflictos estructurales cuando diferentes transiciones comparten los mismos

lugares de entrada, y conflictos dentro de una transición cuando las mismas marcas en los lugares de entrada pueden dar lugar a diferentes modos de disparo excluyentes. Por lo tanto, el conjunto conflicto puede ser dividido en distintos conjuntos conflictos, los cuales agrupan las transiciones que están en conflicto estructural. Los modos de disparo de diferentes conjuntos conflictos pueden ser ejecutados concurrentemente. Incluso dentro de un objeto conflicto pueden existir modos de disparo que se pueden ejecutar concurrentemente. En cada ciclo de inferencia se tratan todos los objetos conflicto que contienen transiciones sensibilizadas, y el resultado de la resolución de cada conflicto son todos los posibles modos de disparo que pueden ser ejecutados concurrentemente. De esta forma se preserva la concurrencia del modelo.

Como se ve en la **fig. 2**, la forma de obtener todos los modos de disparo concurrentes de cada conflicto es seleccionar y disparar uno a uno los modos de disparo hasta que no queden transiciones sensibilizadas en el conflicto. El disparo de las transiciones actualiza la lista de transiciones sensibilizadas al retirarse las marcas que dan lugar al modo de disparo. Cada transición puede tener asociada una tarea que es ejecutada concurrentemente con las tareas de otras transiciones disparadas en el presente ciclo. El disparo de las transiciones se ha separado en dos acciones (**fig. 2**): Disparo1 dentro del bucle más interno actualiza las precondiciones y comienza la ejecución de las tareas asociadas. Al actualizarse las precondiciones se actualiza la lista de transiciones sensibilizadas. De esta forma se evita que dos modos de disparo que necesitan las mismas marcas puedan acceder a ellas en el mismo ciclo, y se evita que la ejecución de dos modos de disparo excluyentes dejen al sistema en un estado incoherente. Disparo2 se realiza fuera del bucle anterior y se encarga de actualizar las postcondiciones de todas las transiciones disparadas. De esta manera se evita el tener en cuenta nuevas transiciones sensibilizadas hasta el siguiente ciclo de inferencia.

5. Consideraciones sobre la eficiencia

El número de operaciones en el proceso de correspondencia de patrones, que se hace cuando se añade una marca a un lugar depende de los siguientes factores:

· El número de lugares de entrada a la transición. Un gran número de éstos puede implicar una gran cantidad de test de consistencia.

·El número de marcas en los lugares de entrada o si las condiciones de arco son más o menos restrictivas.

·El orden de los arcos. El proceso de correspondencia se realiza evaluando secuencialmente las expresiones que etiquetan los arcos, parando cuando no existen ligaduras con sistentes con una secuencia inicial de expresiones. El situar primero las expresiones más restrictivas, o aquellas cuyo lugar de entrada permanece normalmente vacío reduce el número de test de consistencia.

· El número de transiciones descendientes de un lugar. Si el número de transiciones descendientes de los lugares es alto habrá que actualizar un número mayor de árboles asociados a las transiciones.

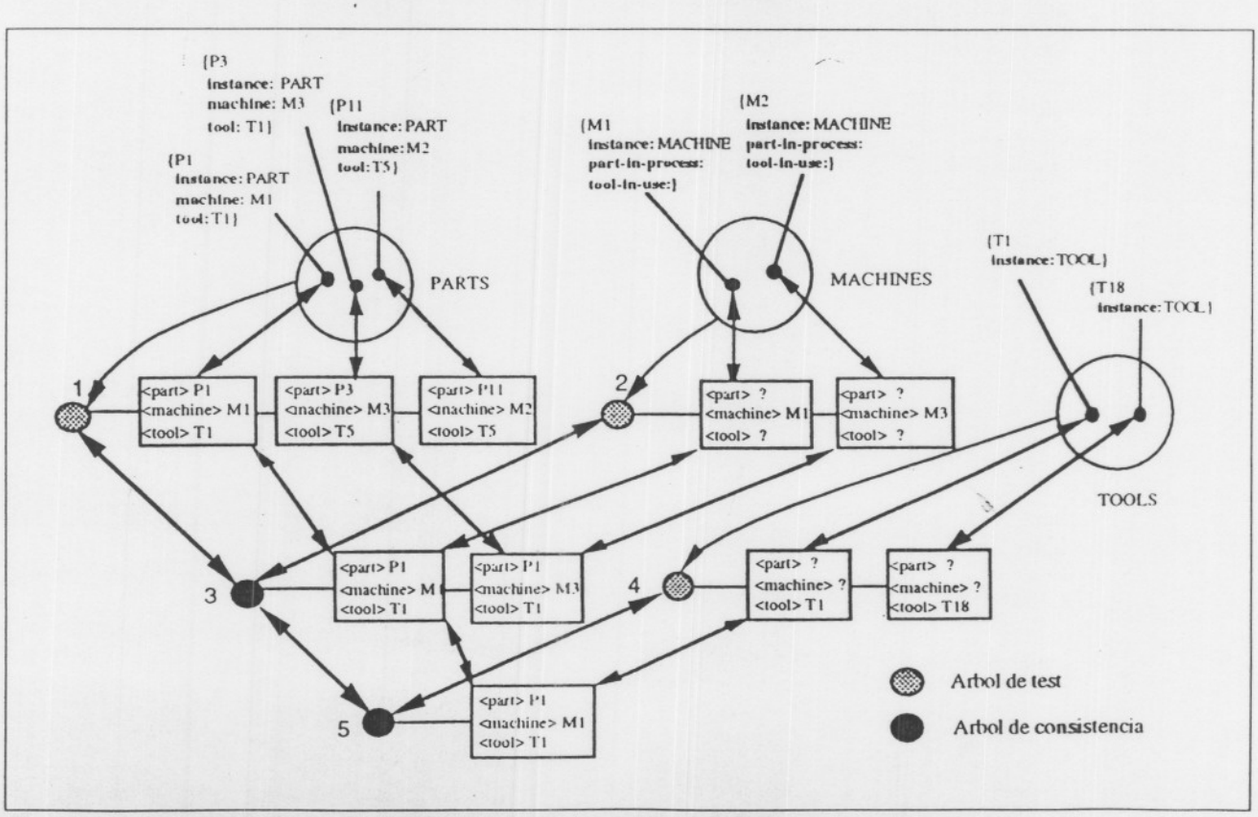


Figura 3: árbol generado para la transición LOAD aprovechando la estructura de la RAN

Para concluir, el mecanismo de interpretación de KRON será más eficiente cuando se pueda beneficiar de la persistencia de información. Esto es, cuando haya muchas marcas y la mayoría de ellas no cambien de lugar en cada ciclo. KRON es más eficiente si el número de marcas involucradas en cada disparo es bajo en comparación con el numero total de marcas y el número promedio de transiciones descendientes de los lugares es bajo.

El coste de extraer una marca de un lugar depende de los mismos factores, pero es mucho menor que el de añadir ya que no se repite el proceso de correspondencia de patrones y se eliminan búsquedas innecesarias.

6. Conclusiones

El trabajo muestra la implementación del mecanismo de inferencia de KRON, un lenguaje de representación del conocimiento para sistemas de eventos discretos. La interpretación de la red de alto nivel subyacente en los modelos KRON constituye la base para su mecanismo de inferencia. El mecanismo de inferencia de KRON hace uso de las similitudes entre el mecanismo de interpretación de RAN y de los SBR. La adaptación del algoritmo de RETE aprovecha las peculiaridades de las redes de alto nivel mejorando la eficiencia del mecanismo de inferencia de KRON. El mecanismo de inferencia de KRON se aprovecha de la partición de la memoria de trabajo en lugares y evita búsquedas y la repetición del proceso de correspondencia de patrones cuando se extrae un elemento de la memoria de trabajo. Dicho mecanismo es más eficiente cuando hay persistencia de la información entre cada ciclo de inferencia y el número de transiciones que desciende de cada lugar no es muy alto. Por otro lado, en KRON, el conjunto conflicto se divide en varios conjuntos conflictos como consecuencia de la estructura de la red de alto nivel subyacente. Esto permite la definición de políticas específicas de resolución según el tipo de aplicación. Por último, el ciclo de inferencia de KRON preserva la concurrencia inherente a las redes de alto nivel, ejecutando en el mismo ciclo todos los modos de disparo concurrentes.

KRON ha sido implementado en KEE sobre una estación de trabajo SUN, mientras que el mecanismo de interpretación se ha implementado usando exclusivamente primitivas de Common Lisp.

Referencias

[Bañares93] J.A. Bañares, P.R. Muro-Medrano y J.L. Villarroel. *"Advances in Petri Nets 1993"*. Chapter Taking advantages of temporal redundancy in high level petri nets implementations, pages 32-48. Number 691 in Lecture Notes in Computer Science. Springer Verlag, 1993.

[Browston85] L. Browston, R. Farrel , R. Kant, N. Martin. *"Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming"*. Adisson-Wesley, 1985.

[Bruno86] G. Bruno, A. Elia. *"Operational specification of process control systems: Execution of prot nets using ops5"*. En Proc. of IFIC'86, Dublin (1986).

[Duggan88] J. Duggan, J. Browne. Espnet: expert-system-based simulator of petri nets. *IEEE . Proceedings 135*,4 (July 1988).239-247.

[Forgy82] C. Forgy. *"A fast algorithm for many pattern / many object pattern match problem"*. Artificial Intelligence 19 (1982), 17—37.

[Jensen91] K. Jensen , G. Rozenberg Eds. *"High-level Petri Nets"*. Springer-Verlag, Berlin, 1991.

[Muro89] P.R. Muro and J.L. Villarroel. *"KRON: Redes Orientadas a la Representación del Conocimiento"*. En Actas de la III Reunión Técnica de la AEPIA. (Madrid, 1989).

[Valette90] R. Valette, B. Bako. *"Software implementation of petri nets and compilation of rule-based systems"*. En 11th International Conference on Application and Theory of Petri Nets (Paris, 1990).

Notas

[1] Este trabajo ha sido soportado en parte por el proyecto ROB91-0949 de la Comisión Interministerial de Ciencia y Técnología de España y el proyecto IT-10/91 de la Diputación General de Aragón.

[2] 'Symbolic pattern matching' en la denominación inglesa.

[3] 'Join tree' en la denominación inglesa.

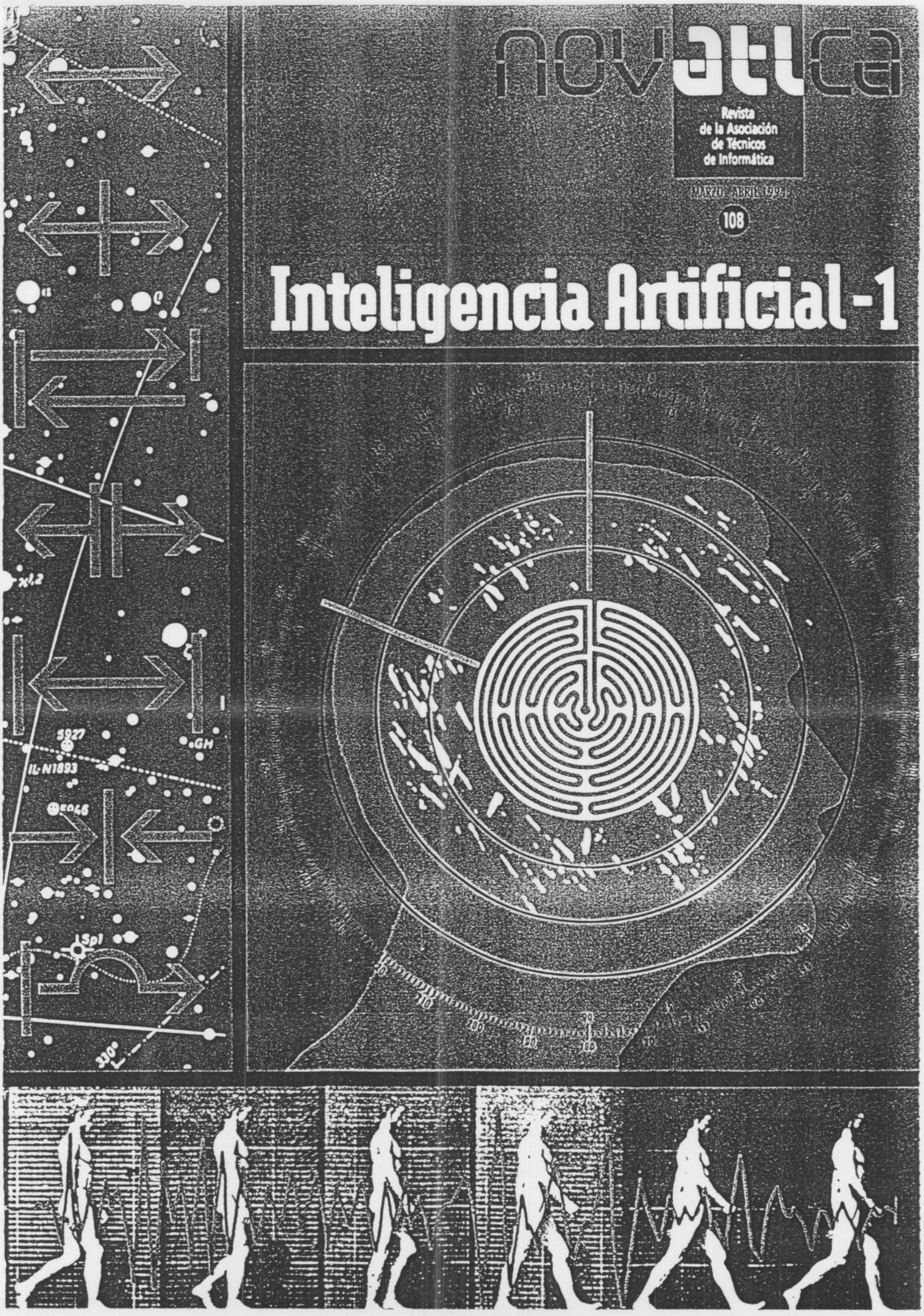
novática

Revista de la Asociación de Técnicos de Informática

MARZO - ABRIL 1994

108

Inteligencia Artificial - 1



Marzo y abril de 1994

NOVATICA es el órgano oficial de Formación permanente de ATI, Asociación de Técnicos de Informática

Redacción (ATI Valencia)
Tirso de Molina 3,14ª,46009 Valencia
(96)3480418; Fax (96)3480683
E-mail: jmarcelo@guest4.atimdr.es

Administración/Publicidad (ATI Cataluña)
Via Laietana 41, 1º, 1ª, 08003 Barcelona
(93)4125235; Fax (93)4127713
E-mail: atibarna@ac.upc.es

Delegación (ATI Madrid)
Padilla 66,3º,28006 Madrid
(91)4029391;Fax (91)3093685
E-mail: secremdr@atimdr.es

Imprenta: LITOTIP, S.A.
Pere IV, 118, 08005 Barcelona. (300 67 61)
Depósito Legal: B 15.154-1975
ISBN: 0211-2124; CODEN NOVAEC

Sumario