# Interpretación reactiva de sistemas basados en conocimiento

José Antonio Alonso Jiménez[1], Joaquín Borrego Díaz[2] y Mario de Jesús Pérez Jiménez[2]
Departamento de Ciencias de la Computación e Inteligencia Artificial
Facultad de Informática y Estadística - Universidad de Sevilla.
E-mail: jborrego@cica.es

## Resumen

Una cuestión fundamental en la Ingeniería del Conocimiento es la necesidad de especificar adecuadamente los sistemas basados en conocimiento. En el presente trabajo proponemos un marco formal que combina el carácter estático de las lógicas de especificación orientadas a datos, con el carácter dinámico de las orientadas a la ejecución, con el objeto de especificar ciertas anomalías asociadas a la base de conocimiento de tales sistemas.

## Introducción

Uno de los problemas fundamentales en la Ingeniería del conocimiento (IC), es la verificación (corrección) de los sistemas basados en conocimiento (SBC). La verificación de los SBC es una actividad profundamente ligada a la adquisición de conocimiento, pues dirige los ciclos de adquisición y refinamiento de éste [1].Si bien para la Ingeniería del Software (IS), existen herramientas formales (por ejemplo, las basadas en lógicas de especificaciones) que representan una referencia formal para el proceso de la verificación de programas, en el caso de los SBC no existen herramientas de este tipo lo suficientemente generales para obtener un planteamiento *fundacional* del problema, debido, principalmente, al carácter *híbrido* de este tipo de sistemas. De hecho, el diseño de marcos lógicos para unificar las distintas cuestiones asociadas al problema es un objetivo permanente en el campo de la verificación de los SBC [2].

Un analisis detallado de los paradigmas de trabajo en IS e IC constata las diferencias de los métodos utilizados para la verificación de sus correspondientes sistemas, pero también establece algunas características comunes. Como se afirma en [3], la especificación completa es ilusoria no sólo para SBC, sino incluso en IS; los SBC son, simplemente, un caso extremo de sistemas con baja especificación.

Un ejemplo es la dificultad de obtención de especificaciones del sistema que relacionen la corrección de éste con la demostración de cierta especificación objetivo, en una teoría de especificaciones. En ciertos sistemas, algunas de las especificaciones naturales no son invariantes de la ejecución (por ejemplo, cuando la regla a la que hace referencia es eliminada en algún momento). Esta deficiencia es debida al carácter *estático* de las especificaciones basadas, en la lógica de primer orden. Una forma de evitar esta limitación consiste en usar lógicas temporales, que utilizan asertos de carácter *dinámico*.

Estudios empíricos han demostrado que las bases de conocimiento contienen, usualmente, potenciales errores que pueden ser descubiertos mediante análisis estáticos y/o dinámicos de la lógica de dicho conocimiento [1]. En este trabajo proponemos la utilización de semánticas asociadas a (y con origen en) la verificación formal de sistemas reactivos, para analizar y especificar la corrección de las bases de conocimiento de los SBC, y para la correción misma del sistema, entendida ésta como la comprobación de que el sistema satisface las especificaciones asociadas a su estructura lógica; es decir, su verificación.

## Formas lógicas del conocimiento

En este trabajo nos ceñimos a SBC basados en reglas, formalizadas en lógica de primer orden. El conocimiento está representado por un conjunto de hechos (literales) y un conjunto de reglas (de producción), que tienen la estructura

$$L_1 \& L_2 \& \ldots \& L_n \dashrightarrow M$$

[1]Financiado por el proyecto DGES PB96-0098-C04-04 del Ministerio de Educación y Cultura y PAI TIC--137 de la Junta de Andalucía.

[2]Financiado por el proyecto DGES PB96-1345 del Ministerio de Educación y Cultura y PAI TIC-137 de la Junta de Andalucía.

con $L_1,...,L_n,M$ literales en un cierto lenguaje. En general, un sistema que utilice la lógica de primer orden está diseñado para atrapar, mediante computaciones, el concepto lógico de prueba formal en la lógica utilizada.

A los tipos de anomalías asociadas a la ejecución del sistema, debemos añadir las que están asociadas al conocimiento base, y es necesario (e importante) descubrir y corregir éstas mediante un análisis de dicho conocimiento. Algunas de las anomalías son[1]:

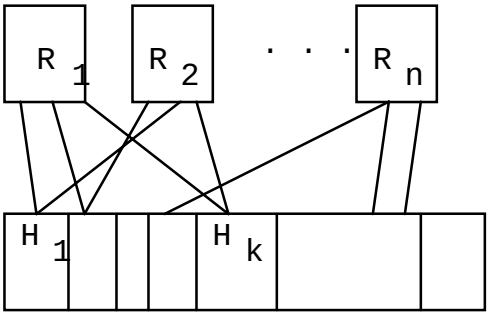
1) *Inconsistencia lógica:* la base de conocimiento no admite un modelo.
2) *Inconsistencia procedural:* El sistema produce, en su ejecución, una inconsistencia. En general, debido a limitaciones del sistema, la inconsistencia procedural puede no coincidir con la inconsistencia lógica.
3) *Redundancia y reglas no usadas:* Algún hecho o regla es innecesaria para producir nuevos resultados, es decir, su eliminación de la base de conocimiento no reduce el conjunto de resultados obtenidos (redundancia), o bien no es posible disparar la regla, o su consecuencia no es usable. Estas anomalías no son, rigurosamente hablando, un error del sistema, y están relacionadas, fundamentalmente, con la eficiencia. Sin embargo, la existencia de estas anomalías suelen ser un síntoma de probables errores en la confección de la base de conocimiento.
4) *Ambivalencia:* Existen reglas con consecuencias inconsistentes, en su conjunto, con el modelo de conocimiento a estudiar por el sistema.
5) *Circularidad:* Existe un conjunto de reglas que, al ser disparadas, produce una ejecución infinita.
6) *Incompletitud lógica:* Existen hechos válidos que no se infieren (lógicamente), de la base de conocimiento.
7) *Incompletitud procedural o deficiencia:* Existen hechos válidos que no los puede generar el sistema.

En el caso de los sistemas basados en la lógica, podemos afirmar que algunas de estas anomalías tienen su origen en que este tipo de representación del conocimiento no permite una buena organización del mismo, desde el punto de vista de la ejecución eficiente y previsible del sistema. Su deficiente modularidad los diferencia de los programas tradicionales, desde el punto de vista de la verificación.

## Bases de conocimiento como sistemas reactivos

Para paliar la baja especificación de los SBC utilizaremos modelos formales diseñados en la IS para sistemas concurrentes: los sistemas reactivos [4]. Un sistema reactivo está formado, básicamente, por un conjunto de procesos que reaccionan a estímulos externos ejecutando ciertas tareas. En cierta medida, se pueden describir como máquinas del tipo MIMD; es decir, máquinas paralelas en las que cada procesador ejecuta instrucciones propias sobre datos propios. Una característica común de todos los modelos formales asociados a estos sistemas es la necesidad de considerarlos como no deterministas, y por tanto, a la hora de especificar y deducir la corrección de los sistemas así diseñados, debemos distinguir entre una ejecución correcta y que *todas las ejecuciones posibles sean correctas*. Para este fín se utilizan distintas lógicas temporales [4].

La modelización reactiva que proponemos consiste en asociar a la base de hechos un sistema (formal) reactivo donde los procesos están, esencialmente, diseñados para el disparo, cuando sea posible, de una única regla. La decisón del disparo de una regla dada está, pues, en el procesador correspondiente, que lee, en la base de hechos (memoria global), los distintos conjuntos de hechos (de entre los ya obtenidos y los de la base de hechos original) hasta encontrar uno que permita disparar su regla. La lectura puede ser considerada, a todos los efectos, como concurrente, mientras que la escritura es exclusiva (del tipo CREW).




[1] En este trabajo nos restringiremos, fundamentalmente, a anomalías asociadas a la base de conocimiento, que no son detectables mediante un análisis sintáctico de las reglas y hechos (reglas subsumidas, no disparables, reglas aisladas, etc...).

Fig. 1: Esquema asociado a las reglas $R_1, \ldots, R_n$ y los hechos $H_1, \ldots, H_k$.

El esquema algorítmico del procesador asociado a cada regla R es el siguiente:

```
* L1: Lectura de una n-upla de hechos, no considerada
     anteriormente, que siguen el patrón de las premisas de R.

* L2: Chequeo de disparo de la regla. Si no es posible, ir a L1.

* L3: Disparo de la regla.

* L4: Insertar el resultado en la memoria de hechos.

* L5: Volver a L1.
```

Por supuesto, es posible que el procesador se mantenga en espera en alguna de las fases del esquema, por ejemplo, cuando espera nuevos literales. En realidad, podemos considerar que el procesador se activa al recibir un *estímulo*, la incorporación de un nuevo literal que es sensible para la regla R[2]. Otras características del modelo son:

* Cada procesador sólo necesita acceso a la parte de la memoria donde residen los literales sensibles para su regla.

* La organización de la memoria de hechos es, en general, sencilla, pues la profundidad de los literales es, en la práctica, baja (es decir, en el caso de lógicas de primer orden, en los términos ocurren pocos símbolos de función).

## Lógicas temporales y sistemas de transición

La lógicas temporales representan una herramienta formal adecuada para especificar sistemas reactivos [4]. Dependiendo del grado de especificación y del tipo de sistema, se eligen distintas lógicas temporales que permiten expresar diferentes propiedades del sistema: precedencia temporal, métricas temporales, continuidad, no determinismo, etc. [5].

La lógica temporal lineal (LTL) considera el flujo temporal como lineal y discreto; cada instante tiene un sucesor y un predecesor (salvo el *instante inicial*). Básicamente, la LTL se obtiene añadiendo a una lógica clásica (proposicional o de primer orden) nuevas conectivas: El operador *siempre en el futuro*, $\Box$, *alguna vez en el futuro*, $\Diamond$, y *en el instante siguiente*, $\bigcirc$ (a los que se les puede añadir las conectivas correspondientes para el pasado). Los modelos de la LTL están formados por una sucesión de estados

$$\sigma: s_0, s_1, \ldots$$

donde cada estado es una interpretación de la lógica clásica empleada. La validez de una LTL-fórmula en un instante $j$ se define de la siguiente manera: $\Box A$ es válida en el estado $s_j$ si para todo estado posterior a éste es válida la fórmula A, $\Diamond A$ es válida si en algún estado posterior a $s_j$ es válida A, y $\bigcirc A$ lo es si A es válida en el estado $s_{j+1}$.

Esta lógica permite especificar propiedades acerca de la ejecución de sistemas reactivos, utilizando un modelo matemático para analizar la ejecución de los mismos: los sistemas de transición [4]. Un sistema de transición consiste, básicamente, en un conjunto de *estados*, asociados a la ejecución del sistema, relacionados entre sí por la ejecución de tareas por parte de algún procesador, y que representan todas las ejecuciones posibles del proceso reactivo[3]. Si bien es natural considerar *lógicas temporales ramificadas* (donde cada instante puede tener más de un sucesor), para analizar tales ejecuciones se puede emplear la LTL, interpretando que las conectivas afectan a *toda posible ejecución.*

[2] Es decir, que unifica con alguna premisa de R.

[3] Hay que hacer notar que, debido a la interacción entre los distintos procesadores del sistema, existe más de una ejecución posible.

## Especificaciones reactivas

Con la lógica temporal se pueden expresar algunas de las anomalías más importantes de los SBC. Con este objetivo, utilizamos en el lenguaje de especificaciones dos tipos de predicados: los del lenguaje de la base de conocimiento y uno nuevo por cada uno de éstos, que tienen la siguiente intención: si p es un símbolo de predicado, introducimos un nuevo símbolo, p', de la misma aridad que p, que expresará la inclusión en la base de hechos de la instancia correspondiente para p.

Utilizando la LTL, podemos especificar distintas propiedades exigidas al SBC (es decir, la ausencia de anomalías), como son las siguientes (notaremos por Lc el literal complementario a L):

-Consistencia lógica: para todo literal L,

$$\neg\Diamond(L \ \& \ Lc)$$

(En todo estado de la ejecución, la base de conocimiento usada es lógicamente consistente).

-Consistencia procedural fuerte para L:

$$\Box\neg(L \ \& \ (Lc)')$$

(No es posible que el sistema deduzca la negación de un hecho válido *en ningún momento*).

-Consistencia procedural débil para L:

$$\Diamond\Box\neg(L \ \& \ (Lc)')$$

(A partir de un cierto estado no se deducen hechos no válidos).

-Completitud procedural para un literal L:

$$L \dashrightarrow \Diamond L'$$

Es decir, si el hecho es válido, entonces se deduce en algún momento. Esta versión de la completitud es la versión para sistemas monótonos (los hechos deducidos siempre permanecen válidos, es decir, en la memoria global de hechos). En el caso de razonamiento no monótono es:

$$L \dashrightarrow \Diamond\Box L'$$

(A partir de algún instante, L' permanece en la base de hechos).

Las propiedades anteriores preservan el grado de complejidad sintáctica del lenguaje, es decir, si la lógica es proposicional, estas especificaciones son fórmulas proposicionales. A modo de ilustración, consideraremos a continuación tres anomalías que utilizan, necesariamente, lenguajes de primer orden. En este caso, consideramos los hechos y reglas como *datos*, y por tanto ciertas variables de la especificación (y de los procesadores) tienen rango en ese conjunto. Utilizaremos variables que nos indican en qué estado de la ejecución se encuentra cada regla-procesador: las constantes predicativas $at_iL_j$ expresar el hecho de que el procesador i está en el punto de ejecución Lj, y la variable ei, que tiene como valor el último hecho deducido por Ri.

-Circularidad: Sea A una conclusión de la regla Ri. Dicha regla provoca un ciclo infinito de disparos con conclusión A si el sistema de transición asociado al modelo reactivo satisface la siguiente fórmula:

$$\Box\Diamond(at_iL_4 \wedge ei = A')$$

-Redundancia: Desde el punto de vista procedural, la regla Ri es redundante si todos los hechos que deduce son deducidos en algún momento posterior por otra regla:

$$\Box(at_iL_4 \wedge ei=x \dashrightarrow \Diamond(\vee_{i\neq j} ej=x))$$

donde x es una variable global con rango en los literales del lenguaje.

-Ambivalencia: Las reglas R1, . . . , Rk son ambivalentes si

$$\neg(A_1 \wedge \ldots \wedge A_k)$$

(donde $A_1, \ldots, A_k$ son las conclusiones de R1, . . . , Rk) un sistema es *efectivamente ambivalente* respecto a tales reglas si todas éstas son ejecutadas, es decir,

$$\Diamond at_1L_4 \wedge \ldots \wedge \Diamond at_kL_4$$

Es evidente que un conjunto de reglas ambivalente y no efectivamente ambivalente contiene alguna regla no usada, y por tanto, es síntoma de un probable error en la confección de la base de conocimiento, como ya hemos comentado.

La verificación de este segundo tipo de anomalías depende fuertemente de la especificación del sistema, concretamente de la especificación del motor de inferencia, y no sólo de la base de hechos. Sin embargo, el uso de una lógica temporal de primer orden no altera sustancialmente las técnicas de detección de anomalías, pues, en la práctica, los modelos de las especificaciones tienen un número finito de estados.

## Detección de anomalías

Una ventaja muy importante de la LTL es la posibilidad de construir, a partir de un conjunto finito de fórmulas consistente, un modelo finito de dicho conjunto (la propiedad de los modelos finitos). Esta construcción se puede, incluso, hacer automáticamente para ciertas lógicas temporales, entre las que se encuentra la LTL. Usando dicha propiedad se establece la ausencia de las anomalías a partir de una especificación temporal del sistema. Esta especificación debe incluir parte del metaconocimiento acerca del sistema concreto a estudiar (es decir, del motor de inferencia) y, por tanto, debe relacionar la deducción lógica con la ejecución del sistema.

Para decidir si de la especificación se deduce la propiedad es posible utilizar técnicas de construcción de modelos. En general, si la especificación (o el sistema) tiene asociado de manera natural un sistema de transiciones (que puede ser leido como un modelo temporal ramificado), para comprobar que el sistema posee la propiedad requerida, se aplican técnicas para validar la fórmula en dicho modelo (*model checking*), que en general, suelen ser más eficientes que las de pura demostración automática [6]. Un análisis preciso de la *efectividad* de las propiedades estudiadas necesita de lógicas más expresivas [5], [7], [8], utilizando teorías basadas en la inducción acotada [9] sobre el flujo temporal.

## Conclusiones y futuro trabajo

Esta organización de las bases de conocimiento es útil para especificar, utilizando lógicas temporales, cuestiones acerca de la verificación del SBC (inconsistencia, incompletitud, etc...). De este modo, la semántica reactiva, al permitir el uso de lógicas temporales, aproxima formalmente las verificaciones orientadas a la ejecución (de tipo dinámico) a las orientadas a los datos (estáticas), y, en general, solventar algunas de las deficiencias formales en la especificación de los SBC que no permiten el uso de técnicas tradicionales de verificación de IS [10].

La semántica reactiva, es decir, asociar a una base de conocimiento un modelo reactivo, pretende ser universal, en el sentido de poder simular, en dichas máquinas, la ejecución de otros sistemas, sin más que sumergir los modelos (paralelos) reactivos del tipo CREW-MIMD (asincronizados) en modelos CREW-SIMD, utilizando un controlador que, mediante variables compartidas, gestione el disparo de las reglas-procesadores. De este modo restringimos el sistema de transición asociado al modelo, e integramos de manera natural el gestor de reglas, para ser verificado conjuntamente con la base de conocimiento. En el caso de secuenciar el disparo de las reglas según los criterios de un sistema concreto, se simularía la ejecución de éste.

El objetivo de esta formalización no es obtener SBC paralelos, ni resuelve, en principio, algunos de los problemas que se plantean en el procesamiento paralelo de conocimiento [11]. Sin embargo, existiría la posibilidad de mostrar algunos resultados acerca de la *eficiencia* de sistemas obtenidos a partir del modelo reactivo (estimaciones acerca del tiempo de obtención de ciertos hechos [7], detección de algunos tipos de incorrección, etc.), aplicando técnicas y resultados bien conocidos de simulación entre modelos de computación paralela, así como diversas funciones asociadas a la eficiencia de éstos. La *lógica temporal factible*, introducida en [7], es una lógica que permite expresar versiones más débiles de las anomalías de los SBC, asociadas a ejecuciones *realistas* del éste.

Otra línea prometedora de investigación es la de sintetizar automáticamente SBC *ad hoc* para una base de conocimiento concreta, usando técnicas de síntesis de esquemas de programas a partir de especificaciones temporales [12] (véase también [13])**Referencias**

[1] Preece, A.D.; Shinghal, R.; Batarekh, A.: Principles and Practice in verifying rule--based systems. *The Knowledge Engineering Review*, 7*(2), 1992, 115--141 .*
[2] Guida, C.; Mauri, G.: Evaluating performance and quality of knowledge-based systems: Foundation and

methodology. *IEEE Transactions on Knowledge and Data Engineering, 5*(2), 1993,204-224.
[3] Juristo, N.; Morant, J.L.: Common framework for the evaluation process of KBS and conventional software. *Knowledge-Based Systems*, 11, 1998,145-159.

[4] *Manna, Z.; Pnuelli, A.: The temporal logic of reactive and concurrent systems. Heidelberg: Springer--Verlag, 1991.*[5] Manna, Z. Pnuelli, A.: Models of reactivity. *Acta Informatica, 20, 1993,609-678.*

[6] *Halpern, J.; Vardi, M.: Model Checking vs. Theorem proving: A manifesto. En Lisfschitz, V. ed. Artificial Intelligence and mathematical theory of computation. London: Academic Press, 1991.*

[7] *Borrego Díaz, Fernández Margarit, A.; Pérez Jiménez, M.J.: Especificación y deducción de propiedades temporales factibles. El sistema FDUX. En Toval, A. y Nicolas J. ed. III Jornadas sobre Ingeniería del Software, Murcia: Marín ed., 1998, 351-362.*

[8] Borrego Díaz, J.; Kemme, B.: Aplicación de la lógica temporal lineal a la representación y deducción de conocimientos temporales. *Lenguajes Naturales y Lenguajes Formales X, 1994, 459-466.*

[9] Borrego Díaz, J.; Fernández Margarit, A.; Pérez Jiménez, M.J.: On overspill principles an axiom schemes for bounded formulas. *Mathematical Logic Quarterly, 42, 1996, 341-348.*

[10] *Schultz, R.; Greissman, J.R.: Bridging the gap between static and dynamic verification. AAAI-88 Workshop on Validation and testing Knowledge--Based Systems, 1988, 1--15.*

[11] *Yan, J.C.: Towards parallel knowledge processing. En Bourbakis, N. ed. Knowledege Engineering Shells , 87--122. Singapore: World Scientific (1993).*
[12] Clarke, E.; Emerson, E.: Synthesis of Synchronization Skeletons for Branching Time Temporal Logic. *Lecture Notes in Computer Science 131, 1981, 52-71.*

[13] *Audureau, E.; Enjalbert, P.; Fariñas del Cerro, L.: Logique temporelle. Semantique et validation de programmes parallèles. Paris: Masson, 1989.*