Programas Moleculares y Sistemas Lógico-Formales

Ángel Nepomuceno-Fernández*
Mario de J. Pérez-Jiménez**
(*Grupo de Lógica, Lenguaje e Información, **Grupo de Computación Natural
Universidad de Sevilla)
nepomuce@us.es
marper@us.es

1. Introducción

Presentamos una ejemplificación de la relación entre sistemas formales y programas moleculares de un modelo de computación molecular. Para ello, se asocia un sistema lógico-formal, mediante una lógica pura de segundo orden, a cada programa molecular que resuelve un problema de decisión y se justifica que la verificación formal de dicho programa (respecto del problema en cuestión) equivale a establecer la adecuación, o corrección, y la completitud del sistema formal asociado.

En un primer apartado introducimos la lógica de segundo orden, teniendo en cuenta la tradicional distinción entre *semántica estándar* y *semántica de Henkin*; se estudia un sistema lógico- formal de segundo orden cuyo lenguaje no tiene variables individuales ni funcionales, se esboza un cálculo deductivo y se establecen su adecuación y su completitud. Sigue un apartado dedicado a introducir una sencilla explicación de qué son los programas moleculares; se da una formalización de un programa molecular y se asocia a éste un sistema lógico-formal basado en la lógica de segundo orden estudiada. En el último apartado se caracteriza la verificación formal de programas moleculares en términos de las propiedades metateóricas del sistema lógico-formal asociado. Concluimos con unas breves consideraciones finales y una bibliografía básica.

2. Lógica de segundo orden

En general, la lógica de segundo orden se diferencia de la de primer orden en que la cuantificación se define sin restricciones; es decir, en lógica de segundo orden, además de cuantificar variables de individuo, se cuantifican variables cuyo rango son subconjuntos del universo del discurso, y las relaciones definidas en el mismo, entendiendo por rango justamente el conjunto de valores semánticos posibles que se pueden asignar a los signos no lógicos del lenguaje de que se trate. El cálculo deductivo de segundo orden, *CD2*, como una ampliación de un cálculo deductivo natural de primer orden, se define mediante axiomas y/o reglas que vienen a garantizar que el rango de las variables predicativas sea, según su aridad, una clase de subconjuntos *definibles* del universo de discurso, o subconjuntos *definibles* del correspondiente conjunto potencia. Por lo que respecta a la semántica en segundo orden, ésta puede ser descrita de dos maneras posibles. De una parte, dado un conjunto no vacío D, para cada aridad $n \geq 1$, se considera una clase Dn de subconjuntos de la potencia cartesiana n-ádica de D (es decir, los elementos de Dn son conjuntos de n-tuplas de D) que es cerrada respecto de las operaciones conjuntistas de unión, intersección y complementario. Dicho con otras palabras, para cada $\boldsymbol{R} \in Dn$, su conjunto complementario $-\boldsymbol{R}$ también está en Dn, y dados dos elementos de Dn, tanto su unión como su intersección son elementos de Dn. Para cada fórmula $A(Y)$, en la cual la única variable libre es la variable predicativa n-ádica Y, se verifica que el par (D, Dn) *satisface* la fórmula $\exists Y A(Y)$ si y sólo si –en adelante "syss"–, existe un elemento $\boldsymbol{R} \in Dn$ tal que (D, Dn) satisface $A(\boldsymbol{R})$ –tomando Y precisamente $\boldsymbol{R}$ como su valor semántico-; simbólicamente, se suele expresar como sigue:

$(D, Dn) \models \exists Y A(Y)$ syss existe $\boldsymbol{R} \in Dn$ tal que $(D, Dn) \models A(\boldsymbol{R})$

En este caso estamos ante la *semántica de Henkin*, en la que, por así decir, todos los predicados o relaciones posibles son justamente los que son definibles. De otro lado, si Dn es el conjunto de "todas las relaciones posibles" (es decir, si Dn es el conjunto de todos los subconjuntos posibles de la potencia cartesiana n-ádica de D), entonces pasamos a lo que se denomina *semántica plena* o *semántica en sentido estándar*.

Los sistemas formales de segundo orden constan de un lenguaje formal de segundo orden y de un *mecanismo deductivo*. Con estos sistemas se consigue una gran expresividad: el *principio de inducción completa* (también llamado *principio de inducción matemática*) es expresable, no ya como un esquema, como ocurriera en primer orden, sino como una sentencia (fórmula sin variables libres) de segundo orden; asimismo, son expresables la finitud, el *axioma de buen orden*, la *hipótesis del*

continuo, etc. No obstante, esa extraordinaria expresividad tiene un coste, a saber, la pérdida de ciertas ventajas que teníamos con sistemas lógicos menos expresivos. Podemos resumir las principales características del cálculo deductivo de segundo orden, *CD2*, en los puntos siguientes:

1. *CD2* es completo respecto de la clase de los modelos de Henkin; es decir, si una fórmula es válida en el sentido de la semántica de Henkin, entonces es demostrable en *CD2*,
2. Tomando la semántica de Henkin, a diferencia de lo que ocurre con la semántica estándar, se verifica *compacidad* (si cada subconjunto finito de un conjunto de fórmulas tiene un modelo de Henkin, entonces tal conjunto tiene un modelo de Henkin), así como el llamado *teorema de Löwenheim-Skolem* (si un conjunto de fórmulas posee un modelo, entonces tal conjunto posee un modelo a lo sumo numerable; es decir, con una cardinalidad no mayor que la de los números naturales-)
3. *CD2* es incompleto en el sentido de la semántica estándar; es decir, cabe hallar una fórmula válida en sentido estándar que no es demostrable en *CD2*: si *G* es la fórmula de Gödel y *AP* la formalización de los axiomas de Peano (en segundo orden, que resulta una fórmula como conjunción de las que formalizan cada uno de tales axiomas), entonces $AP \rightarrow G$ es válida (en sentido estándar) pero no demostrable en *CD2,* pues si fuera demostrable, entonces *G* sería demostrable a partir de *AP*, lo que no puede ser debido a la incompletitud esencial de la aritmética elemental, probada por Gödel
4. Respecto de la semántica estándar, también falla compacidad y el teorema de Löwenheim-Skolem.

Se presentan otras características de interés en el estudio de fundamentos de la matemática clásica, que omitimos para abreviar. A este respecto, en [4] se indica que si un investigador F adopta una lógica de segundo orden con semántica estándar y otro investigador H adopta esta lógica pero con semántica de Henkin, ambos para formalizar pruebas, no hallaríamos diferencias esenciales entre ellos. En cualquier caso, tiene sentido adoptar una lógica de segundo orden con semántica de Henkin, en la cual, como hemos dicho, todas las relaciones posibles que se consideran son precisamente las *definibles* haciendo uso del correspondiente lenguaje formal.

Consideremos un sistema *lógico-formal puro* de segundo orden, similar al establecido en [1] -modificando en parte la notación-. El vocabulario del lenguaje formal correspondiente no contiene variables individuales ni variables funcionales, aunque contenga constantes individuales así como constantes funcionales; las únicas variables son variables predicativas de aridad 1, 2, … Un modelo (en el sentido de la

semántica de Henkin) $M=(D,\{Dk\}k\geq 1, \mathbf{i})$, donde D es un dominio no vacío, $\{Dk\}k\geq 1$ es la familia de dominios relacionales, tal que, para todo $k\geq 1$, Dk es un conjunto de relaciones k-ádicas definidas en D (cerrado para las operaciones conjuntistas, como se indicó más arriba); $\mathbf{i}$ representa la función interpretación, de manera que para cada constante individual a, $\mathbf{i}(a)\in D$; para cada constante funcional de f aridad $k\geq 1$, $\mathbf{i}(f)$ es una función de esta aridad definida en D; y para cada constante predicativa de R de aridad $n\geq 1$, $\mathbf{i}(R)$ es un predicado de esta aridad definido en D. Para abreviar, anotaremos $M=(D, \mathbf{i})$ en lugar de $M=(D,\{Dk\}k\geq 1, \mathbf{i})$. Por otra parte, a cada variable Y de aridad $n\geq 1$, se asigna como valor semántico una relación de su misma aridad $\mathbf{R}\square Dn$, siendo Dn el conjunto relacional n-ádico; es decir, el conjunto de las relaciones (posibles) definidas en D; dada la función $\mathbf{i}$, a una variable Y se asigna $\mathbf{R}$ como valor semántico, y para una fórmula B anotaremos $B(\mathbf{R})$, para indicar que consideramos el valor de la fórmula B habiendo asignado $\mathbf{R}$ a Y. Para completar la semántica (de Henkin), definimos recursivamente cuándo un modelo $M=(D,\mathbf{i})$ satisface una fórmula (usando el símbolo |= para representar "satisfacción"),

1. $M \models Rt1t2...tn$ syss $\langle \mathbf{i}(t1),...,\mathbf{i}(tn)\rangle \in \mathbf{i}(R)$, si R es una constante predicativa n-ádica; si la fórmula es $Yt1t2...tn$, y el valor semántico de Y es $\mathbf{R}$, entonces la condición es que $\langle \mathbf{i}(t1),...,\mathbf{i}(tn)\rangle \in \mathbf{R}$

2. $M \models \neg A$ syss no es caso que $M \models A$

3. $M \models A\vee B$ syss $M \models A$ o bien $M \models B$ -asimismo, *mutatis mutandis*, se evalúan las demás conectivas-

4. M $\models \exists YB$ syss para algún $\mathbf{R}\in Dn$ ($n\geq 1$ es la aridad de la variable Y) $M \models B(\mathbf{R})$

5. M $\models \forall YB$ syss para todo $\mathbf{R}\in Dn$ ($n\geq 1$ es la aridad de la variable Y) $M \models B(\mathbf{R})$

Nótese que, en general, si $A(Y)$ es una fórmula con la variable Y libre, entonces $M\models A(Y)$ si se verifica que $M\models A(\mathbf{R})$, para todos los valores $\mathbf{R}$ que puedan ser asignados a Y; es decir, para todos los valores posibles (que son las relaciones definibles) asignables a Y. Asimismo, podemos definir las nociones de *validez* y *consecuencia lógica*. Si $\$$ es un conjunto de fórmulas, $M \models \$$ indica que M satisface todas las fórmulas de $\$$.

Definición 2.1 *Dada una sentencia A, diremos que es universalmente válida en el sentido de la semántica de Henkin, en símbolos |= A, syss para todo modelo M, M |= A.*

Definición 2.2 *Dados un conjunto de sentencias $ y una sentencia A, diremos que A es consecuencia lógica de $ en el sentido de la semántica de Henkin, en símbolos $ |= A, syss para todo modelo M, si M |= $, entonces M |= A.*

Para completar la descripción del sistema lógico-formal puro de segundo orden, mediante *CDP2* nos referimos al mecanismo deductivo, que será un cálculo representado mediane |– y semejante al de primer orden, al cual se han de añadir las reglas (en su caso, axiomas) que rigen el comportamiento de los cuantificadores, cuyos únicos sufijos son variables predicativas, de acuerdo con el carácter del lenguaje adoptado. Teniendo en cuenta los cálculos definidos en [2], fijamos reglas de eliminación de ∀ y de introducción de ∃, sin restricciones, como en las reglas de primer orden; en concreto, la eliminación de ∃ y la introducción de ∀, se expresan en los esquemas

$$\frac{\exists Y A(Y);\ (A(Y) \vdash K)}{K}$$

donde $(A(Y)|\!-K)$ representa una deducción subsidiaria (respecto de la principal, la cual comienza con una hipótesis auxiliar $A(Y)$, y concluye en K, cerrándose entonces tal hipótesis), siempre que K sea una fórmula en la que no ocurra libre la variable Y y que tampoco ésta ocurra libre en ningún supuesto previo o provisional (ya sea premisa o hipótesis auxiliar no cancelada) del cual dependa K, excepto la propia $A(Y)$.

Asimismo

$$\frac{A(Y)}{\forall Y A(Y)}$$

siempre que Y no ocurra libre en ningún supuesto previo o provisional del cual $A(Y)$ dependa.

Definición 2.3 *Dada una fórmula A, diremos que es deducible en CDP2, en símbolos |– A, syss existe una sucesión de sentencias A1,...An, n≥1, tal que para cada j≤n, Aj es un axioma o una consecuencia inmediata de fórmulas precedentes, y An es A. Dado el conjunto de fórmulas $, diremos que A es deducible desde $ syss existe una sucesión de sentencias A1,...An, n≥1, tal que para cada j≤n, Aj es un axioma, o Aj∈$, o Aj es consecuencia inmediata de fórmulas precedentes, y An es A.*

Teorema 2.4 *(Adecuación/corrección) Para cada conjunto de fórmulas $ y la fórmula A, si $ |–A, entonces $ |= A.*

CDP2 es una extensión del correspondiente cálculo de primer orden, que es adecuado. Las reglas específicas de los cuantificadores, como se comprueba fácilmente, garantizan, en cada modelo, el paso de fórmulas verdaderas a fórmulas verdaderas (si el antecedente es verdadero, el consecuente también), de manera que *CDP2* es también correcto.

Teorema 2.5 *(Completitud) Para cada conjunto de fórmulas $ y cada fórmula A, si $|=A, entonces $|–A.*

La demostración, en este caso, se puede hacer por reducción al absurdo. Supuesto que $|=A, se supone después que no es el caso que $|–A; entonces, $ junto con ¬A constituyen un conjunto consistente; en efecto, en otro caso, $,¬A|–⊥, es decir "lo falso" se deduciría de $ y ¬A, de donde tendríamos que $|–¬¬A y por lo tanto $|–A, pero ello contradice lo anteriormente supuesto. De acuerdo con el *lema de Lindembaun*, dicho conjunto consistente se puede extender a máximamente consistente y existencialmente saturado; a partir de éste, interpretando cada constante individual como ella misma y asignando a cada constante predicativa de aridad $n \geq 1$ las n-tuplas de constantes individuales que son sus argumentos en las correspondientes fórmulas atómicas de tal conjunto, se define un modelo de Henkin (que es a lo sumo numerable) que lo satisface; por tanto también al conjunto consistente inicial. Ahora bien, de acuerdo con el punto de partida, todo modelo de Henkin que satisfaga a $ debe satisfacer a A; pero el modelo definido satisface $ y satisface ¬A, o, lo que es lo mismo, no satisface A. Estamos, pues, ante una contradicción, de donde el segundo supuesto es de negar, de manera que finalmente $|– A.

Como corolario de la adecuación y completitud en sentido fuerte, en el sentido de la semántica de Henkin, tenemos los mismos resultados para el sentido débil: si |–A, entonces |=A. En efecto, si una fórmula es demostrable (sin premisas), entonces es válida, lo cual se verifica como caso particular del primer teorema cuando el conjunto $ es vacío. De manera similar, en el caso de la completitud.

3. Programas moleculares

Un *modelo formal de computación molecular* consta, básicamente, de (a) una *estructura de datos* que se denominan *tubos* (abstracciones de los tubos de ensayo) asociados a un alfabeto prefijado; (b) una *sintaxis* que precisa las estructuras formales de las instrucciones básicas (*instrucciones moleculares*) y los procedimientos mecánicos del modelo (*programas moleculares*); y (c) una *función semántica* que describe la forma en que se ejecutan los programas moleculares cuando se le suministran datos de entrada.

Formalmente, un tubo T es un multiconjunto (es decir, un conjunto cuyos elementos pueden aparecer repetidos) finito de cadenas sobre el alfabeto $SADN$ =

{*A,C,G,T*} y notaremos T□(*S*ADN),* siendo *S*ADN* el conjunto de todas las palabras sobre el alfabeto *SADN*. Las instrucciones moleculares básicas son abstracciones de operaciones bioquímicas que se pueden realizar en la actualidad, con bastante precisión, en un laboratorio de Biología Molecular, y los programas moleculares son sucesiones finitas de instrucciones moleculares junto con instrucciones estándares (bucles FOR, WHILE y asignación) que proporcionan la secuencia de ejecución de las instrucciones moleculares. La especificación o instanciación de las instrucciones moleculares básicas proporcionarán distintos modelos de computación molecular. La *función semántica* del modelo se define de manera natural, siendo para las instrucciones moleculares, la abstracción del resultado correspondiente de la operación bioquímica que representa. Las entradas de los programas moleculares son tubos y las salidas estarán codificadas en el tubo final del programa: diremos que el resultado de la computación es sí en el caso de que exista alguna molécula en el tubo final, y el resultado computación es no en el caso de que el tubo final esté vacío. En el primer caso, diremos que el programa molecular *acepta* la instancia del problema codificada por el tubo inicial, y en el segundo caso diremos que *rechaza* dicha instancia.

Los modelos básicos de computación molecular están basados en procedimientos de *filtrado*; es decir, las computaciones en dichos modelos parten de un tubo inicial que contiene cadenas (eventualmente repetidas) que codifican todas las posibles soluciones del problema que trata de resolver y, mediante un determinado criterio, se procede a realizar sucesivos filtrados hasta obtener un tubo de salida que debe contener todas las soluciones correctas del mismo, y sólo esas. En dichos modelos, las computaciones no alteran la estructura interna de las moléculas que aparecen en los tubos; es decir, la información que codifica cada molécula del tubo inicial es invariable a lo largo de todo el proceso de ejecución. Por ello, se suele decir que estos modelos de computación carecen de memoria de acceso aleatorio.

A la hora de trabajar con programas moleculares hay que distinguir claramente una *fase de inicialización o precomputación* y una *fase de ejecución*. En la primera, se construye un tubo de ensayo inicial con la condición de que toda posible solución del problema a resolver está codificada mediante una molécula de dicho tubo y, en la segunda, se van filtrando del tubo inicial las moléculas que satisfacen ciertas condiciones, de tal manera que en el tubo final aparezcan las cadenas que codifican las soluciones correctas del problema (y sólo esas).

Este tipo de programas moleculares permiten resolver *problemas de decisión*. No obstante, antes de hablar de este tipo de problemas, vamos a introducir unos problemas que son muy usuales en la vida real: los *problemas de optimización*.

Definición 3.1 *Un* problema de optimización*, X, es una tupla (I_X, s_X, f_X) en donde: (a) I_X es un lenguaje sobre un alfabeto, cuyos elementos se denominan* instancias *del problema; (b) s_X es una función cuyo domino es I_X y para cada instancia a de X el conjunto s_{X(a)} es finito y sus elementos se denominan* soluciones candidatas *asociadas a esa instancia; y (c) f_X es una función, denominada* función objetivo*, que asigna a cada instancia a del problema y a cada solución candidada asociada c_a, un número racional positivo.*

El número racional *f_X(a,c_a)* es el valor que la función objetivo le asigna a esa solución candidata y viene a ser un indicativo de la *bondad* de esa solución. La función objetivo *f_X* proporciona el *criterio* para determinar una *solución óptima*.

Un caso particularmente importante de problemas de optimización son los *problemas de decisión*. Informalmente, un problema de decisión es un problema abstracto que sólo admite como respuesta o salida sí o no.

Definición 3.12 *Un* problema de decisión*, X, es un par ordenado (EX, ZX), donde EX es un lenguaje sobre un cierto alfabeto SX y ZX es un predicado booleano ("...es verdadero", "...es falso") sobre EX.*

Los elementos del conjunto *EX* se denominan *instancias* o *datos de entrada del problema*. Para cada instancia *a* de *EX,* existe un conjunto finito *S(a)* cuyo elementos se denominan *soluciones candidatas* que permiten responder afirmativa o negativamente a la cuestión planteada por el problema.

Es interesante hacer notar algunas consideraciones. Los problemas de decisión son mucho más simple de tratar que los problemas de optimización. Ahora bien, si nos centramos en la resolución de problemas de decisión y análisis de su complejidad computacional, entonces no estamos restringiendo sustancialmente nuestro estudio. Esto quiere decir lo siguiente: a cada problema de optimización X es posible asocciarle un problema de decisión D_X de tal manera que, en primer lugar, la construcción de D_X a partir de X se puede realizar consumiento pocos recursos computacionales; y en segundo lugar que para cada solución del problema de decisión D_X se puede construir una solución del problema de optimización X de tal manera que esa construcción también consuma pocos recursos computacionales. Dicho con otras palabras, es posible resolver de manera indirecta un problema de optimización X reduciéndolo a la resolución de un problema de decisión D_X de tal manera que una *buena solución* de éste, nos proporciona indirectamente una buena solución del problema de optimización X. Así pues, a la hora de resolver problemas importantes de la vida real, es suficiente que nos restrinjamos al estudio y resolución de problemas de decisión.

Sea P un programa molecular diseñado para resolver un problema de decisión X=(*EX, ZX*). Para cada instancia *a* de *EX,* un *tubo inicial con dato de entrada a* es un multiconjunto de cadenas de *SADN* que contiene exactamente aquellas cadenas que representan o codifican todas las soluciones candidatas del problema que están asociadas a esa instancia. Notaremos *I(P,a)* el conjunto de todos los posibles tubos iniciales del programa *P* con dato de entrada *a*. Teniendo presente que cada tubo inicial es un elemento de *M*(*S*ADN*) resulta que *I(P,a)* es un subconjunto del *conjunto de partes finitas de M*(*S*ADN*), es decir del conjunto ***P***(*M_F*(*S*ADN*)). Si T es un tubo inicial del programa *P* con dato de entrada *a,* entonces notaremos *P(a,T)*=1 (respectivamente, *P(a,T)*=0) para indicar que la ejecución del programa *P* a partir del tubo inicial T proporciona como resultado sí (respectivamente, no).

Definición 3.3 *Un programa molecular P resuelve un problema de decisión* X=(*EX,ZX*) *si para cada instancia a de EX se tiene verifica: (1) si ZX(a)=1, entonces P(a,T)=1, para cada tubo inicial T de P con dato de entrada a; y (2) si ZX(a)=0, en*tonces *P(a,T)*=0, *para cada tubo inicial T de P con dato de entrada a.*

En este trabajo se trata de justificar que todo programa molecular diseñado para resolver un determinado problema de decisión se puede describir a través de un sistema formal de segundo orden, de tal manera que la verificación formal del programa molecular para ese problema (es decir, la prueba de que realmente el programa resuelve todas las instancias del problema) sea equivalente a establecer la corrección y completitud del sistema formal asociado.

Recuérdese que para *verificar* formalmente *un programa molecular* en relación con un problema de decisión, hay que establecer dos resultados relativos a cualquier tubo inicial *T* del programa asociado a un dato de entrada *a* del problema:

(a) Toda cadena del tubo de salida que proporciona la ejecución del programa a partir del tubo inicial *T* representa una solución correcta del problema asociada a la instancia *a* (*corrección o adecuación del programa molecular*); es decir, si el tubo de salida no está vacío, entonces cada cadena de dicho tubo codifica una solución correcta del problema.

(b) Toda cadena del tubo inicial *T* asociada a la instancia *a* que codifica una solución correcta del problema asociada a esa instancia, debe estar en el tubo de salida que proporciona la ejecución de dicho programa a partir del tubo inicial *T* (*completitud del programa molecular*); es decir, si el tubo de salida está vacío, entonces no existe ninguna solución correcta del problema.

Definición 3.4 *Un sistema molecular es un par ordenado (X,P), en donde X es*

un problema de decisión y P un programa molecular.

Verificar un sistema molecular *(X,P)* consiste en demostrar que el programa molecular *P* resuelve el problema de decisión *X*. Por tanto, para verificar un sistema molecular *(X,P)* hay que probar que para cada instancia del problema, $a \in EX$, y cada tubo inicial $T \in I(P,a)$, se tiene que $P(a,T)=1$ syss $ZX(a)=1$; es decir, que la salida del programa coincide con la respuesta del problema. La implicación "si $P(a,T)=1$, entonces $ZX(a)=1$" recibe el nombre de *corrección* del programa P para el problema X, y la implicación recíproca "si $ZX(a)=1$, entonces $P(a,T)=1$" recibe el nombre de *completitud* del programa *P* para el problema *X*. Esto justifica la siguiente definición:

Definición 3.5 *Diremos que el sistema molecular (X,P) está verificado syss para cada* $a \in EX$ *y cada* $T \in I(P,a)$ *se tiene que* $P(a,T)=1$ *syss* $ZX(a)=1$.

Así pues, un sistema molecular *(X,P)* está verificado syss el programa *P* es correcto y completo para el problema *X*. La verificación de un sistema molecular *(X,P)* equivale a que el programa molecular *P* resuelva el problema de decisión *X*, de acuerdo con la definición 3.3.

4. Verificación de programas moleculares y CDP2

La formalización introducida tiene por objeto caracterizar la verificación formal de programas moleculares diseñados para resolver problemas de decisión, en términos de corrección y completitud de un cierto sistema formal asociado, como en el ejemplo que se propone a continuación. De acuerdo con lo que hemos indicado en la sección anterior, una metodología similar puede ser usada para resolver problemas de optimización mediante programas moleculares, debido a que a todo problema de este tipo se le puede asociar un problema de decisión de *manera natural* y de tal forma que la cantidad de recursos necesarios para implementar dicha construcción sea *razonable*.

Como se indicó más arriba, CDP2 representa el cálculo puro de predicados de segundo orden, que consta de axiomas y reglas establecidas como extensión del de primer orden. Mediante D nos referimos al conjunto de sus axiomas.

Definición 4.1 *Dado un sistema molecular (X,P), se le asocia un sistema formal SCDP2(X,P), de acuerdo con lo siguiente:*

(a) El lenguaje de SCDP2(X,P) viene dado por $EX \cup EADN$ *y tiene como*

*constantes funcionales: f(a,T)=P(a,T), siendo a una instancia del problema X, T∈M(E*ADN) un tubo inicial asociado a la instancia a y P(a,T) la ejecución del programa P con entrada el tubo inicial T.*

(b) El conjunto de fórmulas de SCDP2(X,P) viene dado por los pares (a,T) en donde a∈EX y T∈I(P,a).

(c) Una fórmula (a,T) del sistema es válida en SCDP2(X,P) syss ZX(a)=1 (lo que proporciona un concepto de validez).

(d) Una fórmula (a,T) del sistema es demostrable en SCDP2(X,P) syss P(a,T)=1 (lo que proporciona un concepto de deducción).

Proposición 4.2 *Sea (X,P) un sistema molecular. Se verifica:*

a) *El sistema formal SCDP2(X,P) es adecuado syss para cada instancia del problema a∈EX y cada tubo incial del programa asociado a esa instancia T∈I(P,a) se tiene que si P(a,T)=1, entonces ZX*(*a*)=1.

b) El sistema formal SCDP2(X,P) es completo syss para cada instancia del problema a∈EX y cada tubo incial del programa asociado a esa instancia T∈I(P,a) se tiene que si ZX(*a*)=1*, entonces P(a,T)=1.*

Demostración

a) En primer lugar, supongamos que el sistema *SCDP2(X,P)* es adecuado; es decir, que si una fórmula del sistema es demostrable, entonces es válida. Sea *a* una instancia del problema y *T∈I(P,a)* un tubo inicial asociado a esa instancia de tal manera que se verifique *P(a,T)*=1. En tal situación, la fórmula *(a,T)* es demostrable. Teniendo presente que el sistema *SCDP2(X,P)* es adecuado, resulta que la fórmula *(a,T)* es válida. Por tanto, *ZX*(*a*)=1.

Recíprocamente, supongamos ahora que para cada instancia del problema a∈EX y cada tubo incial del programa asociado a esa instancia *T∈I(P,a)* se tiene que si *P(a,T)=1,* entonces *ZX*(*a*)=1. Vamos a probar que el sistema *SCDP2(X,P)* es adecuado. Caso contrario, existiría una fórmula *(a,T)* de *SCDP2(X,P)* que sería demostrable y que, en cambio, no sería válida. Puesto que la fórmula *(a,T)* es demostrable resultaría que *P(a,T)*=1, pero al no ser válida se tendría que *ZX*(*a*)=0. Lo cual sería una contradicción.

b) En primer lugar, supongamos que el sistema *SCDP2(X,P)* es completo; es decir, que si una fórmula del sistema es válida, entonces es demostrable. Sea *a* una instancia del problema y *T∈I(P,a)* un tubo inicial asociado a esa instancia de tal manera que se verifique *ZX*(*a*)=1. En tal situación, la fórmula *(a,T)* es válida. Teniendo presente que el sistema *SCDP2(X,P)* es completo, resulta que la fórmula *(a,T)* es demostrable. Por tanto, *P(a,T)*=1.

Recíprocamente, supongamos ahora que para cada instancia del problema a∈EX y cada tubo incial del programa asociado a esa instancia *T∈I(P,a)* se tiene que si *ZX*(*a*)=1, entonces *P(a,T)=1*. Vamos a probar que el sistema *SCDP2(X,P)* es completo. Caso contrario, existiría una fórmula *(a,T)* de *SCDP2(X,P)* que sería válida y que, en cambio, no sería demostrable. Puesto que la fórmula *(a,T)* es válida resultaría que *ZX*(*a*)=1, pero al no ser demostrable se tendría que *P(a,T)*=0. Lo cual sería una contradicción.

Corolario 4.3 *Son equivalentes las proposiciones siguientes:*

1. *El sistema molecular (X,P) está verificado*
2. *El sistema forma asociado SCDP2(X,P), es adecuado y completo*

Demostración: En primer lugar, veamos que (1) implica (2). En efecto: si el sistema molecular *(X,P)* está verificado, entonces para cada instancia *a∈EX* y cada tubo inicial *T∈I(P,a)* asociado a esa instancia se tiene que *P(a,T)*=1 syss *ZX*(*a*)=1. Por tanto, de la proposición 4.2 se deduce que el sistema *SCDP2(X,P)* es adecuado y completo.

A continuación, veamos que (2) implica (1). En efecto: si el sistema *SCDP2(X,P)* es adecuado y completo, entonces de la proposición 4.2 se deduce que para cada instancia *a∈EX* y cada tubo inicial *T∈I(P,a)* asociado a esa instancia se tiene que *P(a,T)*=1 syss *ZX*(*a*)=1. En consecuencia, el sistema molecular *(X,P)* está verificado.

5. Consideraciones finales

El sistema formal de segundo orden CDP2 cuenta con los elementos e ingredientes necesarios para definir un sistema formal asociable a un programa molecular. De hecho, tanto el conjunto de constantes individuales, como el de las constantes funcionales y el de las constantes predicativas, son finitos o, a lo sumo, numerables, mientras que el conjunto de las variables predicativas es a lo sumo numerable. La semántica correspondiente es una semántica de Henkin, con lo que estamos ante una lógica correcta y completa, lo que permite una buena aplicabilidad, por así decir, en particular para establecer un conjunto de fórmulas que representan los pares "instancia del problema de que se trate y (la formalización de) tubo inicial asociado a la misma". Así, si el conjunto de tubos iniciales asociado a una instancia *a* del problema, en la notación usada *I(P,a)*, del programa *P* relativos al dato de entrada *a*, es representado por la fórmula ***F****(a)* del lenguaje (de segundo orden) de CDP2,

$\exists Y\mathbf{F}(a)$ expresa que existe Y tal que es un tubo inicial del programa P relativo al dato de entrada a.

Es interesante hacer notar que el sistema formal asociado a un programa molecular, de acuerdo con lo descrito en el presente trabajo, puede describir la verificación formal del mismo en relación con un cierto problema de decisión, en el sentido siguiente: dado un programa molecular que ha sido diseñado para resolver un problema de decisión, es posible verificar dicha propiedad (es decir, que el programa realmente decide correctamente *todas* las instancias del problema) estudiando el sistema formal asociado; más concretamente, estableciendo la adecuación y completitud del sistema.

Agradecimientos

El primer autor ha realizado este trabajo en el marco del proyecto FFI2011-29609-C02-00 del Ministerio de Ciencia e Innovación, así como del proyecto de excelencia P2010-HUM-5844 de la Junta de Andalucía. El segundo autor quiere agradecer la ayuda recibida del proyecto TIN2009-13192 del Ministerio de Ciencia e Innovación, así como del Proyecto de Excelencia con Investigador de Reconocida Valía P08-TIC-04200, para poder realizar este trabajo de investigación.

Referencias

[1] Denyer, N.: "Pure Second-Order Logic", *Notre Dame of Formal Logic,* vol. 33, n. 2, 1991: 220-224

[2] Nepomuceno-Fernández A.: "Sistemas de cálculo como formas de logicismo", *Crítica*, vol. XXV, n. 73, 1993: 15-35

[3] Pérez-Jiménez, M. J.: "Métodos formales en computación bioinspirada", en *Lógica e Filosofia da Ciência*, O. Pombo, A. Nepomuceno (eds.), Centro de Filosofia das Ciências Universidade de Lisboa, 2009: 185-212

[4] Väänänen, J.: "Second order logic and foundations of mathematics", *The Bulletin of Symbolic Logic*, vol. 7, n. 4, 2001: 504-520