LOGICA DINAMICA

María Manzano

Abstract

The proliferation of logics of programs, formal languages for logic programming and logics for Al makes it urgent to investigate their interrelationships in depth. The search for a logic into which all these others can be translated is a necessary and promising task.

Many-sorted logic (*MSL*) is a *natural* logic for reasoning about more than one kind of objetct; i.e., data and programs, data and time in computing, or states and programs in dynamic logic, *MSL* is also an *efficient* logic since its proof-theory is very powerful. Furthermore, many higher order logics (type theory, second order logic), infinitary logics, non-classical logics (modal logics, temporal logics, many-valued logics) and logics of programs (dynamic logic, Hoare logic) can, in principle, be translated into *MSL*. Thus, it provides a *unified framework* for all these logics.

This paper is about propositional dynamic logic (*PDL*), a meta-language designed to convey properties of computing programs. In it, I show how *PDL* can be unified by *MSL*. There are four sections:

PDL is presented as one of the logics of programs.

PDL is placed among modal logics.

PDL is defined; i.e., its formal language, its semantics and calculus.

PDL is shown to be equivalent to a many sorted theory that includes some comprehension sentences and some abstract conditions for axioms in propositional dynamic logic.

Estoy aquí para responder a la pregunta, ¿qué es la lógica dinámica? Caben varias respuestas:

¿Qué es la lógica dinámica?

1) Una de las lógicas de programas.
2) Una extensión de la lógica modal.
3) Esto. *(Especificando lenguaje, semántica* y *cálculo deductivo).*
4) Una teoría en lógica multivariada de primer orden *(O, también, una teoría en cierta lógica de segundo orden).*

AGORA 9 (1990): 31-43

Refrenaré mi impulso de centrarme exclusivamente en el punto 3) porque creo que ello os dejaría bastante insatisfechos. Al extenderme en el punto 1) espero situarla en el contexto de las lógicas de programas y que se entiendan las causas que motivaron su creación, qué necesidades satisface (el famoso ¿para qué sirve?, que a los que nos dedicamos a la lógica tanto nos indigna, fundamentalmente porque nunca podemos satisfacer a nuestros interlocutores ya que habitualmente no es una pregunta, sino una acusación de esterilidad). En el punto 2) la relaciono con otras lógicas filosóficas en cuyas aguas bebieron los creadores de la lógica dinámica. De esta forma la ubicaré históricamente. Finalmente, en el punto 4), la presentaré inmersa en una lógica que se caracteriza por su versatilidad y por ser la que, en el fondo, mejor le cuadra a la matemática y a la informática. Expondré aquí el resultado de mi trabajo personal sobre lógica dinámica sentencial. Este último planteamiento surge de la necesidad de unificación que abruma a los que nos dedicamos a la informática teórica.

La proliferación actual de lógicas de programas, de lenguajes de programación lógica y de lógicas para inteligencia artificial hace necesaria en este momento una investigacion profunda de las interrelaciones entre ellas. Encontrar una lógica suficientemente versátil a la que las demás lógicas puedan traducirse es tarea urgente, necesaria y prometedora por cuanto nos permitiría fundamentar, integrar y crear una teoría general de tan vasto campo.

La lógica multivariada se nos presenta a un amplio grupo de investigadores como una opción válida y con mucho futuro. La lógica multivariada es la lógica *natural* para razonar acerca de objetos de tipos diversos: vectores y escalares en matemáticas, datos y programas, o datos y tiempo, en programación (en especial, en la programación de sistemas expertos), estados y programas en lógica dinámica.

La lógica multivariada es no solamente una lógica natural para referirnos a objetos de diversos tipos, sino también una lógica *eficaz*, pues su teoría de la prueba es muy poderosa.

Sucede, además, que tanto las lógicas de orden superior (Teoría de tipos y lógica de segundo orden), como las lógicas infinitarias, o las no-clásicas (Lógica modal, temporal, multivaloradas), como las lógicas de programas (Lógica dinámica, Lógicas de Hoare) son, en principio, traducibles a la lógica multivariada. Por consiguiente, la lógica multivariada proporciona un *marco unificador* para estas lógicas.

1. La lógica dinámica es una de las lógicas de programas

En la teoría de la computación los objetos de interés prioritario son los programas. Un programa, o algoritmo, es básicamente un procedimiento que nos permite calcular ciertos valores a partir de ciertos datos. Por ejemplo, se puede escribir un programa que, dado un cierto n, calcula el valor F(n) de la serie de Fibonacci.

Cuando manejamos programas es conveniente usar variables que puedan ir tomando diversos valores sobre ciertos universos en cualquier momento de la computación. Sin embargo, a diferencia de lo que sucede en matemáticas, una misma variable puede tomar diversos valores a lo largo de una computación y, en particular, distinto valor inicial y final.

Un programa o algoritmo es un objeto dinámico, capaz de hacer pasar al computador de un estado a otro. Un estado puede concebirse como el contenido de los registros de memoria usados por el programa.

El objetivo de la lógica de programas es el de crear la base lógico matemática precisa para expresar nuestros razonamientos acerca de los programas de computación. Por supuesto, en esta nueva lógica se siguen los estándares de rigor y precisión que son comunes a la lógica y a la matemática actual.

¿Qué propiedades de los programas nos interesa formular en este metalenguaje que es la lógica de programas? En primer lugar, es de suma utilidad el poder expresar y demostrar que un programa es correcto; es decir, que se adecúa al fin para el que fue diseñado. Parece también interesante formular la propiedad de tener fin; es decir, que acaba en un momento determinado sin, por consiguiente, inducir una computación interminable. Otra propiedad fundamental es la de equivalencia de programas. Sería deseable tener, además, un cálculo deductivo para verificar dichos razonamientos.

¿Por qué no se utiliza la lógica clásica cuyos lenguajes y cálculos conocemos tan bien? Por principio, la lógica clásica es de naturaleza estática: la verdad de una sentencia depende, si estamos en lógica sentencial, de la asignación de valores de verdad que hagamos a las letras sentenciales y, si estamos en lógica de primer orden, de a qué relaciones y funciones se refieren los relatores y functores del lenguaje. Y cuando no son sentencias, sino fórmulas abiertas, también de la asignación de valor a las variables. No hay en ella necesidad de referirse a las consecuencias de ciertas acciones, no hay que contar con la posible alteración del valor de verdad de una fórmula como resultado de una acción (Por ejemplo, la ejecución de un programa).

Sin embargo, en teoría de la computación precisamos de una lógica de naturaleza dinámica. Es decir, en donde, además de poder expresar sentencias tales como: «La serie de los números primos es infinita» o «Todos los hombres son mortales», se puedan expresar otras tales como: «Después de accionar el interruptor se encenderá la luz» o «Después de ejecutar el programa a, φ será verdad». La estructura de estas últimas es del tipo: «Después de ejecutar la acción b sucederá que φ». Oraciones con estas estructuras aparecen continuamente en el discurso del razonamiento y, por lo tanto, las lógicas que las expresan son útiles no sólo en teoría de la computación, sino también en psicología, lingüística y lógica filosófica. No es, por consiguiente, de extrañar que la lógica propuesta para este fin, la denominada lógica dinámica, sea una generalización de la lógica modal.

2. La lógica dinámica es una extensión de la lógica modal

La lógica modal es la lógica de la necesidad y de la posibilidad. Comienza su historia con una etapa primitiva, no sistemática, que incluye los trabajos de: Aristóteles, los megáricos, los estoicos y algunos medievales. Aristóteles dedica dos capítulos de su *De Interpretatione* al estudio de la interdependencia entre necesidad y posibilidad.

Los lógicos modales necesitaron crear un formalismo capaz de captar situaciones dinámicas, de relativizar la verdad. Desde un principio se vio la conexión entre

nociones modales y temporales, siendo ya debatida por los megáricos y los estoicos. Concretamente, Diodoro Cronos la ve, según Boecio, así:

> «Diodoro define lo posible como aquello que o bien es o bien será; lo imposible como aquello que, siendo falso, no será verdadero; lo necesario como aquello que, siendo verdadero, no será falso; y lo no-necesario como aquello que o bien es ya, o será falso».

La etapa sistemática se sitúa en este siglo y comienza, más o menos independientemente, con Lewis, Lukasiewicz y Carnap. La etapa siguiente es calificada por Segerberg de «revolución industrial» y corresponde a la de Kripke, Prior, Kanger y Hintikka. Aunque se atribuye a Kripke la paternidad de los mundos posibles, parece ser que no fue ni el primero ni, por supuesto, el único que definió a los modelos modales en la forma en la que nos han llegado.

Modelos de Kripke

$$A = \langle W, R, \langle P^A \rangle_{P \in \Sigma} \rangle$$

1) W es un conjunto no vacío de mundos.
2) $R \subseteq W \times W$ es una relación de accesibilidad entre mundos.
3) Para cada $P \in \Sigma$, $P^A \subseteq W$ es un subconjunto del conjunto de mundos, asociado a cada variable sentencial.

Los modelos de la lógica modal son sistemas formados por un conjunto *W* de mundos posibles o de estados, una relación de accesibilidad entre mundos y una representación adecuada de los signos del lenguaje clásico.

Lenguaje modal

Alfabeto
1) Σ es el conjunto de variables sentenciales.
2) Todos los signos lógicos de la lógica sentencial clásica.
3) Operadores modales: [ ] y $\prec\ \succ$ (necesariedad y posibilidad, o siempre y alguna vez).
Las fórmulas se construyen inductivamente utilizando estos signos.

Semántica

$A, w \models P$ syss $w \in P^A$ (P es verdadera en el mundo w del modelo A)

$A, w \models [\,]\varphi$ syss $A, t \models \varphi$ en todo $t \in W$ tal que $\langle w,t \rangle \in W$

En el lenguaje de esta lógica, que contiene los signos [ ] y < > para la necesariedad y la posibilidad, escribimos [ ] φ . Esta fórmula expresa que φ es necesariamente verdadera, que es verdadera siempre. [ ] φ será verdadera en un mundo w siempre que φ sea verdadera en todo mundo accesible desde w. Por otra parte, < > φ es verdadera en un mundo w si φ lo es en al menos un mundo accesible desde w.

La relación de accesibilidad recibe en lógica modal multitud de tratamientos originándose diversas teorías modales: KT, KB, KT5, etc. Si pretendemos captar como noción de necesariedad una noción fuerte, leibniziana, la relación de accesibilidad ha de ser de equivalencia y la teoría modal correspondiente será S5. Según Leibniz necesarias son las sentencias que valen en todo mundo posible y posibles lo son las que valen en alguno. Si la noción de necesario no es tan fuerte, podremos tener relaciones de accesibilidad que sean simplemente reflexivas, o simétricas, o transitivas, y estaremos en las teorías KT, KB o K4. Por otra parte, cuando la necesariedad se interpreta temporalmente, la relación de accesibilidad debe ser un orden.

En la actualidad, la lógica modal se ha diversificado enormemente y a los objetivos tradicionales han venido a sumarse: 1) problemas filosóficos muy elaborados (Fine), 2) una ampliación de la teoría de modelos con nociones modales (Mortimer), 3) la interpretación de los operadores modales como demostrabilidad (Boolos, Solovay), 4) nuestra lógica dinámica (Andréka, Harel, Kozen, Meyer, Németi, Pratt) y 5) los planteamientos de la denominada «gramática de Montague».

3. Presentación de la lógica dinámica

¿Cómo se modifica en la lógica dinámica la semántica de los mundos posibles? Como veremos, el esquema abstracto de la lógica modal, y en particular, los modelos de Kripke, le sientan perfectamente a la lógica dinámica. W será ahora el conjunto de los estados relevantes para los programas en consideración. Con cada programa b asociamos una relación binaria de accesibilidad de forma que el par $<s, t>$ está en ella si hay un estado t al que se llega desde s mediante el programa b; es decir, una computación que transforma el estado s en el estado t. Así, un programa es concebido como un conjunto de pares de estados: inicial y final. Puesto que vamos a admitir programas no deterministas -es decir, en donde el estado inicial no determina unívocamente el estado final-, la relación de accesibilidad no necesita ser una función.

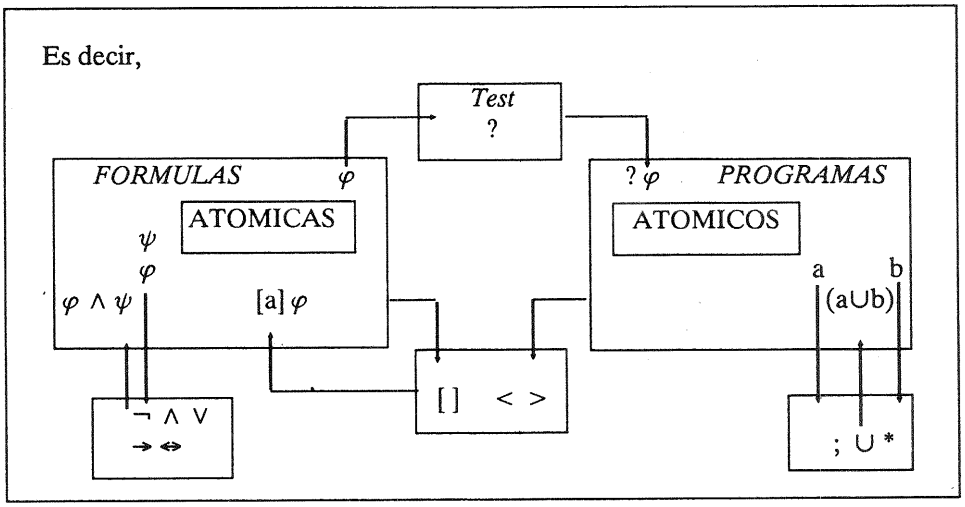
Como es de suponer, asociando a cada programa una relación de accesibilidad, se tienen que manejar simultáneamente muchas de ellas y lo que haremos es colocar dentro de [ ] y de < > el programa concreto al que nos referimos; es decir, escribiremos $[b]$ y $<b>$. Además de esta extensión de la lógica modal, consistente en considerar conjuntamente diversas relaciones de accesibilidad, en lógica dinámica está permitido formar programas compuestos a partir de programas simples.

El lenguaje de la lógica dinámica que voy a presentar es proposicional, su cálculo es correcto y completo, pero en sentido débil, pues el teorema de compacidad falla. Es decir, es completo para validez ($\models \varphi \Rightarrow \vdash \varphi$ es siempre cierto) pero no es completo para consecuencia ($\Gamma \models \varphi \Rightarrow \Gamma \vdash \varphi$ no siempre es cierto).

Lógica dinámica sentencial (LDS)

Sintaxis
$\varnothing_0 = <P_i>_{i \in I}$ *fórmulas atómicas,*
$P_0 = <Q_j>_{j \in J}$ *programas atómicos.*
Inductivamente se construyen $\varnothing$ y Π usando: $\varnothing_0$ y Π_0
Conectores: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$
Operadores modales: [ ] y < >
Operadores de programas: $\cup$, ;, *
Operador de test: ?

Con las fórmulas y programas atómicos se construyen inductivamente las fórmulas y programas de la dinámica usando para ello los conectores, los operadores modales, los operadores de programas y el operador de test. El procedimiento es el habitual en estos casos.



Como muestra el dibujo, a partir de fórmulas cualesquiera, los conectores nos permiten formar nuevas fórmulas (el procedimiento es el normal). Mientras que los programas se combinan mediante sus operadores para formar nuevos programas (la unión y la composición de programas son operadores binarios mientras que el *loop* lo es monario). A partir de una fórmula se forma un programa que es el *test* de esa fórmula. Y con una fórmula, un programa y un operador modal se forma una nueva fórmula. Son estas últimas las fórmulas características de la lógica modal, las que nos permiten expresar propiedades de programas.

El **significado intuitivo** de estos programas y fórmulas es:

(a ; b)	significa: «haz *a* seguido de *b*»
$(a \cup b)$	significa: «haz *a* o *b*»
a^*	significa: «repite *a* un número finito, pero no precisado de veces»
$[a]\varphi$	significa: «*a* es parcialmente correcto respecto del OUTPUT φ»
$<a>\varphi$	significa: «*a* es totalmente correcto respecto del OUTPUT φ»

Aunque sea éste el significado intuitivo de estas expresiones, la definición, mucho más precisa, no es ésta, sino la expuesta en el cuadro siguiente:

Semántica

$$A = <W, <P^A>_{P \in \varnothing_0}, <Q^A>_{Q \in \Pi_0}>$$

Inductivamente se define:
$A(\varphi)$ conjunto de estados donde φ es verdad.
A(a) pares de estados iniciales y finales.

En particular,
$A(a^*) = \{<s, t> \in W \times W / \exists k \exists s_0 \ldots s_k (s_0 = s \wedge s_k = t \wedge \forall i_{1 \leq i \leq k} <s_{i-1}, s_i> \in A(a))\}$
$A([a]\varphi = \{s \in W / \forall t(<s, t> \in A(a) \rightarrow t \in A(\varphi))\}$

También definimos:

$A, s \models \varphi$	(φ es verdadera en el estado *s* de *A*)
$A \models \varphi$	(φ es válida en *A*)
$\models \varphi$	(φ es válida)
$A, s \models \Gamma$	(Γ es satisfacible en el estado *s* de *A*)
$A \models_\exists \Gamma$	(Γ es satisfacible en algún estado de *A*)
$A \models_\forall \Gamma$	(Γ es satisfacible en todo estado de *A*)
$G \models_L \varphi$	(φ es una consecuencia local de Γ)
$G \models_G \varphi$	(φ es una consecuencia global de Γ)

La semántica es, como dije, una ampliación de la modal. Los modelos contienen un universo de mundos o de estados, una familia de subconjuntos del universo de estados y una familia de relaciones sobre mundos. La idea es exactamente la misma que en la modal; cada fórmula se interpreta como el conjunto de los estados en donde es verdad, empezando por las atómicas. En vez de tener una relación de accesibilidad, tenemos una para cada programa atómico ya que los programas son concebidos como objetos dinámicos que permiten que el computador pase de un estado a otro. Al interpretar las fórmulas y programas del lenguaje se hace de forma que toda fórmula represente

el conjunto de los estados en donde es verdad y que cada programa se interprete como el conjunto de pares de estados iniciales y finales entre los que el programa nos lleva.

En particular, la interpretación del *loop* es la menor clausura reflexiva y transitiva de la relación sobre la que se aplica.

Usando los programas básicos y los operadores de programas se pueden construir programas nuevos. Por ejemplo:

Programas compuestos

IF φ THEN a ELSE b $\equiv_{Df}$ $((\varphi\,?\,;a) \cup (\neg\,\varphi\,?\,;b)$
WHILE φ DO a $\equiv_{Df}$ $((\varphi\,?\,a)^*;\neg\,\varphi\,?)$

Propiedades de programas

$[a]\varphi$	*a* es parcialmente correcto, en sentido fuerte.
$<a>\varphi$	*a* es totalmente correcto, en sentido débil.
$<a>\varphi \wedge [a]\varphi$	*a* es totalmente correcto, en sentido fuerte.
$\varphi \rightarrow [a]\psi$	*a* es parcialmente correcto respecto INPUT φ y OUTPUT ψ .
$<a>\varphi \leftrightarrow <b>\varphi$	*a* y *b* son equivalentes por lo que respecta a la condición φ.

Una noción semántica que utilizaré después es la de relación definible en una estructura dinámica. Se trata sencillamente de admitir que los programas y las fórmulas definen relaciones o conjuntos de estados; a saber, el de los pares de estados que conecta, o el de los estados en los que son verdaderas. Diremos que un conjunto o relación es definible cuando existe una fórmula o programa que lo define.

Para la lógica dinámica sentencial hay un cálculo completo en sentido débil. Entre los axiomas de este cálculo se cuentan los de la lógica sentencial no modal, los que regulan el funcionamiento de la composición de programas y alguno de los de la lógica modal. De entre ellos cabe destacar estos dos, el segundo de los cuales es conocido como axioma de inducción.

Cálculo

A5) $<a^*>\varphi \leftrightarrow (\varphi \vee <a><a^*>\varphi)$
A8) $\varphi \wedge [a^*](\varphi \rightarrow [a]\varphi) \rightarrow [a^*]\varphi$

Este cálculo es incompleto en sentido fuerte, tanto si consideramos el concepto de consecuencia local como el global.

La demostración de la incompletud para consecuencia es como sigue:

Sea $\Sigma = \{P, [Q]\,P, [Q;Q]\,P, ..., [Q^n]\,P, ...\}$ y sea $\varphi \equiv [Q^*]\,P$

Es evidente que $\Sigma \models [Q^*]\,P$ pero para ningún subconjunto finito, $\Sigma_o \subseteq \Sigma$, se cumple que $\Sigma_o \models \varphi$. Para demostrarlo, sea Σ_o un subconjunto finito de Σ.

Sea n el mayor de los exponentes de las fórmulas de Σ_o, y sea $A = <N, P^A, Q^A>$ con $P^A = \{0, 1, ..., n\}$ y Q^A la función del siguiente.

En el estado 0 el conjunto Σ_o es satisfacible, pero φ no es verdadera en el estado 0.

4. La lógica dinámica sentencial es una teoría (SOLO^2) en lógica multivariada

Lo que quiero hacer ahora es presentar la LDS como una teoría en lógica multivariada (LMV). Traduciré las fórmulas de la LDS en fórmulas de la LMV y convertiré los modelos dinámicos en multivariados. Procediendo así demostraré la equivalencia semántica y sintáctica de la LDS y de SOLO^2, una teoría en lógica multivariada. Me he tomado este trabajo, no porque me desagrade la lógica dinámica o discrepe del planteamiento que allí se hace, sino por colaborar en la unificación de las lógicas que se usan en informática.

¿Por qué se eligió la LMV?

En muchas ramas de la matemática y de la informática se utilizan explícita o implícitamente estructuras multivariadas; es decir, estructuras que tienen más de un Universo o dominio (sobre cada uno de los cuales toman valores distintos conjuntos de variables).

Hay muchísimos ejemplos:

Ejemplos de estructuras multivariadas

Geometría (*puntos, líneas, ángulos...*).
Teoría de Grupos (*elementos del grupo, subgrupos*).
Teoría de Anillos (*elementos del anillo, subanillos ideales, filtros*).
Espacios vectoriales (*vectores, escalares*).
Teoría de tipos simple (*individuos, conjuntos de individuos, conjuntos de conjuntos de individuos...*).
Lógica de segundo orden (*individuos, relaciones monarias, binarias...*).
Cuando se computa (*datos, números naturales...*).
Para razonar acerca de programas de computación (*estados y programas).*

¿Cuál es la lógica que cubre nuestras necesidades mejor?

La lógica de primer orden es muy bonita y tiene muchas propiedades interesantes: se puede definir un cálculo correcto y completo, la lógica es compacta y verifica el

teorema de Löwenheim-Skolem, por sólo mencionar las propiedades más importantes.

La lógica de primer orden, obviamente, no es multivariada: las estructuras que se utilizan en lógica de primer orden sólo tienen un universo o dominio. El lenguaje de primer orden sólo tiene una clase de variables. Por consiguiente, podemos intentar meter en estructuras de primer orden, con una sola variedad, nuestras estructuras multivariadas. De hecho, éste es el tratamiento estandard de la LMV: la reducción a la lógica de primer orden (LPO). Este es el tratamiento que dio a la lógica multivariada Hao Wang[1], uno de los pioneros en el estudio de esta disiciplina.

Por consiguiente, que la lógica multivariada se reduce a la de primer orden es un resultado conocido, y éste es el tratamiento que recibe en los libros de texto. La reducción se realiza a dos niveles: una traducción sintáctica de las fórmulas multivariadas en fórmulas de primer orden, conocida como relativización de cuantificadores, y una conversión de las estructuras multivariadas en monovariadas, conocida como Unificación de dominios.

Lo que no se dice en los libros de texto es que por la reducción de la lógica multivariada a la monovariada hay que pagar un alto precio:

Precio de la conversión

1) Se pierde naturalidad.
2) El teorema de interpolación se debilita.
3) No siempre se preserva la interpretabilidad de una teoría en otra.
4) Su teoría de la prueba es peor.

1) Uno de los objetivos al estudiar la LMV era el de encontrar una lógica que se adecuara al tipo de estructuras que queríamos estudiar. Resulta que los supuestos sobre los que se asienta la LPO, por lo que hace referencia al tipo de estructuras a estudiar, son inadecuados. Por consiguiente, se pierde naturalidad al pasar a la lógica de primer orden sin variedades.

2) Feferman[2] señaló, en 1967, que, por lo que hace referencia al teorema de interpolación de Craig, es mejor el lenguaje multivariado porque se puede probar para él una versión mejorada de dicho teorema. Puesto que el teorema de interpolación sirve para determinar la eficacia de la teoría de la prueba de una cierta lógica, cabe esperar que la de la multivariada supere a la monovariada, como de hecho sucede.

3) En 1985 Julian Hook[3] demostró que una teoría multivariada puede ser interpretable en otra teoría multivariada sin serlo sus correspondientes traducciones.

4) Desde el punto de vista de la prueba automática de teoremas, es mejor la multivariada porque las deducciones son más cortas y se ahorra uno conclusiones

---

[1] «Logic of many-sorted theories», *JSL*, vol. 17, núm. 2, June, 1952.

[2] *Lectures on proof theory. Lecture Notes in Mathematics 70*, Springer-Verlag, 1968.

[3] «A note on interpretations of many-sorted theories», *JSL*, vol. 50, núm. 2, June, 1985.

inútiles. Es decir, teoremas de la lógica de primer orden que no son traducción de ninguna fórmula multivariada. La razón es que, al relativizar cuantificadores, añadimos nuevos relatores al lenguaje para expresar la propiedad de ser miembro de un cierto universo. Por consiguiente, desde el punto de vista de los que se dedican a la prueba automática de teoremas, entre cuyos objetivos está el de obtener conclusiones lógicas con eficacia y rapidez y evitar resultados inútiles, la reducción a la lógica monovariada es inaceptable.

Estas, y me imagino que también otras razones, han forzado a un grupo considerable de personas a elegir la lógica multivariada como marco en donde situar otras lógicas, relacionadas o no con la informática, y a utilizarla para estudiar las estructuras, evidentemente multivariadas, que queremos investigar.

Todo esto sirve de justificación a por qué he querido pasar de la lógica dinámica, presentada al estilo modal, a la multivariada. Vereis que, pese a lo que dije en un principio al presentar modalmente la dinámica, el punto de vista multivariado sigue siendo una forma muy natural de abordar el tema, que tiene la ventaja adicional de incluirse en un planteamiento unificador. Por ejemplo, la lógica de segundo orden con la semántica general de Henkin, la que posee un cálculo completo, no es más que una teoría en lógica multivariada.

Para traducir la LDS en LMV necesitamos un lenguaje con tres variedades que representen individuos, conjuntos y relaciones.

El alfabeto contiene los programas y fórmulas atómicas de LDS, pero ahora como relatores monarios y binarios. Tendremos tres tipos de variables y los cuantificadores, pero prescindiremos de los operadores modales y de programas. En cambio, añadiremos unos relatores de pertenencia.

LDS *versus* LMV

Alfabeto
1) Los signos de $\varnothing_0$ y Π_0 junto con igualador y conectores.
2) Variables de tres tipos (*individuales, relacionales monarias y binarias*).
3) Cuantificadores.
4) Nuevos relatores de pertenencia.

Para interpretar este lenguaje formal utilizaremos: marcos construidos sobre estructuras dinámicas, estructuras generales en el sentido de Henkin (esto es posible porque la lógica de segundo orden es lógica multivariada disfrazada), estructuras estándar y lo que denomino estructuras dinámicas extendidas.

Estructuras

Marcos construidos sobre estructuras dinámicas
Sea
$A = <W, <P^A>P\in \varnothing_0, <Q^A>Q \in \Pi_0>$

Definamos
$AF = \langle W, W', W'', \langle P^A \rangle_{P \in \varnothing_0}, \langle Q^A \rangle_{Q \in \Pi_0} \rangle$
donde:
1) $W' \subseteq PW$ y $P^A \in W'$, para cada $P \in \varnothing_0$
2) $W'' \subseteq PW \times W$ y $Q^A \in W''$, para cada $Q \in \Pi_0$
Estructuras generales.
Estructuras estándar de segundo orden.
Estructuras dinámicamente extendidas.

De entre todas estas estructuras son las dinámicamente extendidas las que nos permitirán demostrar la equivalencia entre la lógica dinámica y la multivariada.

La idea es tener en los universos a todos los conjuntos y relaciones que se puedan definir dinámicamente en A. Es decir:

Estructuras dinámicamente extendidas

$AD = \langle W, W', W'', \langle P^A \rangle_{P \in \varnothing_0}, \langle Q^A \rangle_{Q \in \Pi_0} \rangle$
es una estructura dinámicamente extendida a partir de A
syss
1) $\varnothing \in W'$ y $W \in W'$
2) $\varnothing_0^A \subseteq W'$ y $\Pi_0^A \subseteq W''$
3) $\forall RS\,(R, S \in W' \rightarrow R \cup S, W - R, R \cap S, \ldots \in W')$
4) $\forall R\,(R \in W'' \rightarrow R \cup S, R \circ S, R^* \in W'')$
5) $\forall R\,(R \in W' \rightarrow I_R \rightarrow W'')$
6) $\forall RS\,(R \in W'' \wedge S \in W' \rightarrow Dom(R \cap (W \times S)) \in W')$

A cada fórmula de LDS le asociamos una en el nuevo lenguaje multivariado. Lo que se pretende es pasar a una fórmula que defina, en una estructura multivariada construida sobre A, el mismo conjunto que la original definía en A, y lo mismo para programas. Puesto que se pueden utilizar diversas clases de estructuras, el resultado varía según cual sea la utilizada.

Función traductora

LDS		LMV
$\varphi \in \varnothing$	$\longrightarrow$	$\varphi'(x)$
$a \in \Pi$	$\longrightarrow$	$a'(x, y)$

$P'(x) \equiv Px$
$(\neg \varphi)'(x) \equiv \neg (\varphi'(x))$
.
.
$([a]\varphi)'(x) \equiv \forall z(a'(x, z) \rightarrow \varphi'(z))$ con z nueva

$Q'(x, y) \equiv Qxy$

.

.

$(a^*)'(x, y) \equiv \forall X^2 (a' \subseteq X^2 \wedge \text{Reflexiva } X^2 \wedge \text{Transitiva } X^2 \rightarrow \varepsilon\, xyX^2)$
(Es decir, la menor relación reflexiva y transitiva que contiene a la original)

Como veis, la interpretación del *loop* es la menor relación reflexiva y transitiva que contiene a la original. Pero pudiera no ser la menor de las posibles, puesto que pudiera suceder que la estructura no la tuviera en su dominio de relaciones.

En las estructuras estándar de segundo orden el *loop* se interpreta de manera estándar, y lo mismo sucede en las estructuras definidas dinámicamente.

He definido también una teoría, $SOLO^2$, cuyos axiomas son los siguientes:

$SOLO^2$

1) Axiomas de definición de clases y relaciones para las traducciones de fórmulas y programas.

$\exists X^2 \forall xy\, (\varepsilon_2\, xyX^2 \leftrightarrow a'(x, y))$ cada $a \in \Pi$

$\exists X \forall y(\varepsilon_1\, yX \leftrightarrow \varphi'(x))$ cada $\varphi \in \varnothing$

2) Condiciones abstractas relativas a los axiomas A5 y A8 de la dinámica.

He podido demostrar unos cuantos teoremas que me permiten cerrar el ejercicio con el siguiente resultado (Donde D es la clase de las estructuras dinámicamente extendidas y F la de los marcos contruidos sobre estructuras dinámicas):

CONCLUSIÓN

$$\begin{array}{ccccc} \models \varphi & \longleftarrow & \models_D \forall x\varphi'(x) & \longleftarrow & SOLO^2 \models_F \forall x\varphi'(x) \\ \updownarrow & & & & \updownarrow \\ \vdash \varphi & & \longrightarrow & & SOLO^2 \vdash \forall x\varphi'(x) \end{array}$$

María MANZANO
Universidad de Barcelona