UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS MATEMÁTICAS
Departamento de Informática y Automática

TESIS DOCTORAL

Una lógica no estandar admisible para programas funcionales

MEMORIA PARA OPTAR AL GRADO DE DOCTOR
PRESENTADA POR

Ana Gil Luezas

Madrid, 2015

UNIVERSIDAD COMPLUTENSE DE MADRID
Facultad de Ciencias Matemáticas
Departamento de Informática y Automática

UNA LOGICA NO ESTANDAR ADMISIBLE PARA PROGRAMAS FUNCIONALES

Ana Gil Luezas

Madrid, 1990

UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS MATEMATICAS
DEPARTAMENTO DE INFORMATICA Y AUTOMATICA

UNA LOGICA NO ESTANDAR ADMISIBLE PARA PROGRAMAS FUNCIONALES

MEMORIA PRESENTADA POR
ANA GIL LUEZAS

Para la Obtención del
GRADO DE DOCTOR EN CIENCIAS MATEMATICAS
Por la Universidad Complutense de Madrid

Bajo la Dirección de
TERESA HORTALA GONZALEZ

Madrid, Diciembre de 1989.

Quiero dar las gracias en primer lugar y de forma muy especial a Teresa Hortalá, directora de la tesis, por su continuo apoyo desde el "primer día".

Por supuesto, a Mario Rodríguez y a Antonio Gavilanes, junto con los cuales hemos trabajado estos años.

A todas las personas integrantes de la Sección de Informática y Automática de la Universidad Complutense de Madrid, por las facilidades que me han dado en la realización de este trabajo.

A mi familia y a todos aquellos que de una forma u otra han hecho posible este trabajo.

INDICE

INTRODUCCION

Las lógicas de programas surgen para proveer de fundamentos lógicos a los métodos de verificación de programas. Durante los últimos años se han investigado diversas lógicas de programas, buscando cada vez una mayor capacidad de expresión. Paralelamente se han desarrollado técnicas para clasificar y comparar la capacidad de expresión de estas lógicas.

El interés de las lógicas de programas dentro de la informática teórica, se centra principalmente en la formalización de:

- La verificación de programas.
- La equivalencia de programas.
- La terminación de programas.

A un primer nivel podemos distinguir dos enfoques distintos en las lógicas de programas: las lógicas exógenas, que son aquellas en las que los programas se encuentran de forma explícita en el lenguaje, y las lógicas endógenas, en las que los programas son fijos y forman parte de la estructura. Entre el primer grupo se encuentran la lógica Algorítmica [Sal 70] y la lógica Dinámica, tanto en su versión proposicional [FL 77], como en su versión de primer orden [HMP 77]. Mientras que la lógica Temporal [Pnu 77] se encuentra entre las endógenas.

Por otro lado, como es sabido, las lógicas de primer orden de programas son fuertemente incompletas con respecto a la teoría de modelos estándar y además no pueden capturar algunas propiedades de programas, como por ejemplo la corrección total. De estos problemas, surgen las lógicas de programas no estándar [ANS 82], de la misma forma que surgió la lógica de segundo orden no estándar. Como único camino para "completar" teorías incompletas estándar, ya que solo en estos modelos puede precisarse la propiedad de "no demostrabilidad".

La mayoría de las lógicas de programas existentes están diseñadas para lenguajes de programación imperativos. Y actualmente, los lenguajes de programación declarativos (y en particular, los funcionales) están ganando popularidad por diversas razones [Eis 87], entre las que se encuentran:

- Acercamiento del código del progrma a la abstracción de los lenguajes de especificación y métodos matemáticos de verificación.
- Acercamiento a la realización de procesamiento paralelo real.
- Fusión de los conceptos de código y datos.
- Implementación de tipos abstractos de datos polimórficos.

Hasta el momento hay muy pocas publicaciones sobre lógicas de programas funcionales. En concreto, las principales referencias que conocemos son: Cartwright [Car 83], [Car 84], Cartwright y McCarthy [CM 79], Goerdt [Goe 85], [Goe 87] y Pazstor [Paz 86].

El trabajo de Goerdt [Goe 85], [Goe 87] trata las funciones de orden superior, pero se restringe a la corrección parcial de programas.

Las investigaciones de Cartwright y McCarthy [CM 79] [Car 83], [Car 84] y Pazstor [Paz 86] están en la vertiente no estándar de las lógicas de programas. Tanto Cartwright y McCarthy como Pazstor se basan en la semántica del menor punto fijo definible. Ninguno provee de facilidades sintácticas para construir programas funcionales estructurados (sus programas son más restrictivos que los que se proponen en este trabajo), ni para razonar sobre su comportamiento mediante composicionalidad.

La conocida lógica para funciones computables LCF [GMW 79] está orientada a un entorno de demostraciones interactivo que no se trata en este trabajo.

A partir de estos precedentes y motivaciones, creemos que el estudio de axiomas y reglas de inferencia composicionales específicas para el comportamiennto de operadores como: λ-abstracción, aplicación funcional y μ-operador, tienen cierto interés metodológico y puede ayudar a comprender mejor la lógica para las funciones definidas recursivamente.

En este trabajo presentamos un lenguaje de programación funcional abstracto, junto con una lógica de programas para él. El lenguaje permite funciones indeterministas, llamadas a funciones predefinidas y expresar λ-abstracciones, condicionales y definiciones mutuamente recursivas o μ-recursión.

Interesándonos principalmente como objetivo, centrar la investigación en un marco no estándar, inspirado en el de Andréka, Németi y Sain para la lógica Dinámica [ANS 82], puesto que provee de un marco lo suficientemente potente como para tratar los fenómenos de incompletitud, y caracterizar y comparar métodos de verificación de programas.

Para esto, en primer lugar hemos hecho un estudio de los problemas y posibles soluciones, presentados por la semántica estándar. El principal problema es que el concepto de menor punto fijo de un operador no es un concepto de primer orden, por lo que no existen cálculo finitarios correctos y completos para razonar sobre funciones definidas recursivamente. Hemos abordado este problema en primer lugar, mediante un cálculo infinitario que refleja la semántica operacional de la recursión. Y en segundo lugar, mediante un cálculo finitario orientado a la sintaxis, aritméticamente correcto y completo, que se fundamenta en la capacidad de expresión de las estructuras aritméticas, para poder definir mediante una fórmula de primer orden, el menor punto fijo de un operador continuo [Gil 89].

Partiendo de esta capacidad de expresión de las estructuras aritméticas, hemos definido la semántica no estándar usando estructuras de tres géneros: uno para los datos, uno para los números internos y otro para las secuencias internas de datos. A estas estructuras las hemos llamado precisamente pseudoaritméticas; restringiéndolas a aquellas que verifican ciertos axiomas de "admisibilidad" de primer orden, a fin de que los números y las secuencias internas se adecúen al cánon estándar, obtenemos una caracterización de la semántica no estándar del operador μ, en términos del menor punto fijo definible [GHR 89].

El trabajo está organizado por capítulos, y estos a su vez están divididos en secciones. Las definiciones, teoremas, etc de cada sección están numerados de forma consecutiva. La forma de referenciar por ejemplo el lema 2 de la sección 1 del capítulo 3 es: lema 2 de la sección 3.1.

Brevemente el contenido de los capítulos es el siguiente:

1- LOGICAS DE PROGRAMAS.

Se hace un resumen del estado actual de la investigación de las lógicas de programas, desarrollando más detalladamente los puntos tratados en esta introducción.

2- LA LOGICA LRF PARA PROGRAMAS FUNCIONALES.

Se define la sintaxis y la semántica de la lógica estándar para programas funcionales LRF. Y se obtienen resultados sobre la capacidad de expresión de LRF, en particular se demuestra que el μ-cálculo proposicional de Kozen PMC y la lógica dinamica para programas regulares de primer orden QDL pueden expresarse en LRF, que LRF puede expresarse en $L\omega_1\omega$ y que la terminación de programas no triviales no puede expresarse en primer orden, dentro del marco de la semántica de LRF.

3- CALCULOS DEDUCTIVOS PARA LRF.

Se presentan dos cálculos deductivos orientados a la sintaxis para LRF. Uno infinitario correcto y completo, y otro finitario aritméticamente correcto y completo. El primero se basa en el método de los tableaux para el problema de satisfactibilidad y el segundo, en la expresividad de las estructuras aritméticas. Como corolario obtenemos el teorema de Löwenheim-Skolem para LRF y que el problema de validez para fórmulas de LRF es Π^1_1-completo.

4- ENFOQUES SEMANTICOS NO ESTANDAR: LAS LOGICAS NLRF Y ALRF.

En primer lugar se definen las estructuras de tres géneros llamadas pseudoaritméticas, con un género para los datos, uno para los números internos y otro para las secuencias internas de datos. Adaptando la sintaxis de LRF a estas estructuras, obtenemos la versión no estándar NLRF de LRF, cuya semántica se define mediante una traducción sintáctica de las fórmulas de NLRF a fórmulas de primer orden sobre estructuras pseudoaritméticas.

A continuación, formulamos ciertos axiomas de admisibilidad en el lenguaje de primer orden sobre este tipo de estructuras, e introducimos la restricción ALRF de NLRF, mediante la exigencia de que los modelos sean admisibles; es decir, módelos de los axiomas de admisibilidad. Obtenemos asi, la lógica admisible para programas funcionales ALRF. Demostramos que la semántica de NLRF sobre modelos admisibles puede caracterizarse en términos del menor punto fijo definible y presentamos un cálculo finitario orientado a la sintaxis correcto y completo para ALRF. Mediante ejemplos, mostramos que en ALRF se puede expresar la terminación de programs no triviales, que es imposible tanto en LRF como en NLRF.

Terminamos este capítulo mostrando una técnica para trasferir deducciones de este cálculo a deducciones del cálculo aritmético para LRF, lo que permite obtener el teorema de completitud aritmética como corolario de la completitud admisible. Esta técnica que puede usarse para otras lógicas de programas, como la lógica dinámica.

CAPITULO 1:
LOGICAS DE PROGRAMAS.

Las lógicas de programas son sistemas formales cuyo propósito es proveer de una base matemática precisa para razonar sobre el comportamiento de los programas. Tradicionalmente, el interés se ha centrado en formalizar especificaciones de corrección y demostrar rigurosamente que un programa verifica ciertas especificaciones. Además, también cae dentro de estos sistemas estudiar la equivalencia de programas, comparar el poder de expresión de operadores, sintetizar programas a partir de especificaciones, etc. Paralelamente a esto, se han ido desarrollado técnicas formales para clasificar y comparar las lógicas de programas.

La idea de introducir e investigar un sistema formal para manejar las propiedades de programas en modelos abstractos, se debe a H. Thiele [Thi 66] e independientemente a E. Engeler [Eng 67]. Esta idea recogida por A. Salwicki [Sal 70] dio lugar al sistema formal llamado *Lógica Algorítmica*, precursora de la *Lógica Dinámica* introducida en [Pra 76] y [HMP 77]. Por otro lado surge la *Lógica Temporal*, como una aplicación de la lógica modal a la especificación y verificación de programas, introducida por A. Pnueli [Pnu 77].

Desde entonces, se han investigado sobre todo lógicas de programas dirigidas al estudio de lenguajes de programación imperativos. El principal problema es que las lógicas de programas de primer orden, son fuertemente incompletas con respecto a la teoria de módelos estándar, y además no pueden captar algunas propiedades de programas, como por ejemplo la corrección total, por lo que surgen las lógicas no estándar de programas imperativos, de la misma forma que surgió la lógica de orden superior no estándar, que aunque no son muy populares son el único camino para resolver los problemas presentados en las lógicas estándar.

Por otro lado, actualmente los lenguajes de programación declarativos (y más concretamente, los funcionales) estan ganando popularidad por varias razones, pero comparativamente hay muy pocas investigaciones acerca de lógicas de programas funcionales.

En este capítulo, hacemos un resumen del estado actual de las investigaciones en lógicas de programas y de su evolución. En la primera sección presentamos un breve resumen de los distintos tipos y resultados conocidos para las lógicas estándar de programas imperativos. En la segunda sección introducimos las lógicas de programas no estándar, así como algunos de los problemas resueltos a partir de estas. En la tercera y última sección del capítulo exponemos las diferencias entre los lenguajes de programación imperativos y declarativos, así como las principales referencias que conocemos sobre lógicas de programas funcionales.

1.- LOGICAS ESTANDAR DE PROGRAMAS IMPERATIVOS

Es generalmente aceptado que los primeros estudios serios para proveer de herramientas matemáticas para razonar sobre el comportamiento de los programas es el método de las aserciones invariantes para la corrección parcial de programas propuesto por Floyd [Flo 67] y Naur [Nau 66], al que siguio el sistema de axiomas de Hoare [Hoa 69] que incorpora este método.

La idea de usar lógicas como la de primer orden para razonar sobre programas es introducida simultánea pero independientemente por Thiele [Thi 66] y Engeler [Eng 67]. De esta primera idea surge la *Lógica Algorítmica*, que fue definida por Salwicki [Sal 60]. La lógica Modal de programas fue sugerida por Pratt [Pra 76] (a partir de la lógica de Floyd-Hoare) y posteriormente denominada *Lógica Dinámica* en [HMP 77] como una poderosa herramienta que fue primeramente estudiada por Fischer and Ladner [FL 77] en su versión proposicional. Estas dos lógicas son conceptualmente muy similares y forman parte de las lógicas denominadas Exógenas (esta terminología es usada en [KT]). No entramos en las lógicas de programas proposicionales al caer fuera del estudio de este trabajo (cfr. [Gil 85]).

Por otro lado, Pnueli en [Pnu 77] propone una aplicación alternativa de la lógica Modal para la verificación y especificación de programas llamada *Lógica Temporal* (cuyo precursor es el método de las aserciones inductivas), dando lugar a otro tipo de lógicas de programas denominadas Endógenas.

Las lógicas de programas imperativos se basan fundamentalmente en el concepto de *estado*. Un estado es una valoración de las variables, de forma que un programa puede verse como una transformación de estados. Dado un estado inicial o *input*, el programa recorrerá una serie de estados intermedios, llegando quizas a un estado final o *output*, dando lugar a una secuencia de ejecución del programa. El conjunto de todos los pares formados por el input y el output de una secuencia de ejecución para un programa es llamado la relación de input/output del programa.

Algunas lógicas de programas, como la lógica dinámica, interpretan los programas como su relación de input/output. Esto hace, que estas lógicas no sean apropiadas para estudiar el comportamiento de programas que conceptualmente no paran, como los sistemas operativos. Otras lógicas de programas, como la lógica temporal y la lógica de procesos, interpretan los programas como secuencias de ejecución (que pueden ser infinitas) con lo que se evita la anterior limitación. Sin embargo, para programas que conceptualmente paran, los criterios de corrección se dan tradicionalmente en forma de especificaciones de input/output, que consisten en una relación formal entre los estados input y output, que se supone que el programa debe mantener, junto con una descripción del conjunto de estados de input sobre el cual se supone que el programa debe parar. Para este caso la semántica de la relación input/output es suficiente para determinar si un programa es correcto con respecto a una especificación de este tipo.

Otra diferencia entre las lógicas de programas es el concepto de lógica exógena y endógena. En las lógicas exógenas, como la lógica dinámica, los programas están explícitamente en el lenguaje. En la lógica dinámica, sintácticamente un programa es una expresión construida inductivamente a partir de programas primitivos usando un pequeño número de operadores de programas. Semánticamente, un programa es interpretado como su relación input/output, de forma que esta relación para un programa compuesto se determina a partir de las relaciones denotadas por sus componentes o partes. Este aspecto de composicionalidad permite hacer demostraciones por inducción estructural, y obtener sistemas de deducción formales orientados a la sintaxis (cfr. [Roe 85]). En las lógicas endógenas, como la lógica temporal, los programas son fijos y forman parte de la estructura. Esto hace necesaria una variable *contador* para guardar el punto por el que va ejecutandose el programa. Esta variable específica forma parte de los estados junto con las demás variables del programa. En lugar de operadores de programas hay operadores temporales que describen como las variables del programa incluida el contador cambian de valor en el tiempo. En estas lógicas se sacrifica la composicionalidad por un formalismo más general.

Hemos escogido la lógica estándar dinámica de primer orden como vehículo para presentar el estado actual de estas lógicas. Además, en el siguiente capítulo, usaremos una versión concreta de esta lógica para mostrar el poder de expresión de la lógica para programas recursivos propuesta en este trabajo.

LA LOGICA DINAMICA DE PRIMER ORDEN

Nos centramos ahora en la lógica dinámica de primer orden (DL), para resumir los resultados obtenidos para ella, así como los problemas que presenta y las soluciones que se han ido proponiendo. Para este propósito no es necesario explicitar una sintaxis concreta, basta decir que los operadores de construcción de programas son la asignación, composición, elección no determinista, pregunta, condicional e iteración. Y por supuesto, la interpretación de cada uno de ellos es la estándar. Este resumen es breve e informal, pero en [Har 79] se puede encontrar un excelente y detallado estudio monográfico sobre este tema, así como las distintas variantes (y su clasificación) que existen de esta lógica.

El primer resultado se debe a Harel, Meyer y Pratt [HMP 77], y establece la incompletitud de la lógica dinámica de primer orden DL, en el sentido de que no puede haber sistemas de deducción formales decidibles para DL. El concepto de decibilidad para un sistema de deducción se refiere a que sea decidible el conjunto de deducciones "correctas en el sistema".

COMPLEJIDAD DEL PROBLEMA DE VALIDEZ PARA DL

Dada una signatura Σ:

(1) Si Σ tiene al menos dos símbolos de función unarias y un símbolo de función binaria, entonces el conjunto de las fórmulas de DL válidas es Π^1_1-completo.

(2) Si Σ tiene al menos un símbolo de función unaria y un símbolo de función binaria, entonces el conjunto de fórmulas de DL para corrección parcial válidas es Π_2^0-completo.

(3) El conjunto de fórmulas de DL para corrección total válidas es Σ_1^0.

De este resultado se deduce que no hay cálculos finitarios correctos y completos, ni para fórmulas, ni para fórmulas de corrección parcial de DL válidas. Sin embargo si existe un cálculo finitario correcto y completo para fórmulas de corrección total, ya que solo es expresable en DL la terminación de programas triviales; como establece el siguiente resultado.

TERMINACION DE PROGRAMAS EN DL

Sea $\varphi \rightarrow \langle\alpha\rangle\psi$ una fórmula de DL válida con φ y ψ de primer orden. Existe una constante $k \in \omega$ tal que para toda interpretación $(\mathfrak{D}, \delta)$, si φ se satisface en $(\mathfrak{D}, \delta)$, entonces existe una secuencia de ejecuación de a lo sumo longitud k de p en $(\mathfrak{D}, \delta)$ cuyo estado final satisface ψ.

Este resultado es una versión del resultado de Kfoury y Park [KP 75].

COMPLETITUD PARA FORMULAS DE CORRECCION TOTAL DE DL

Existe un sistema de deducción formal finitario S tal que $\models \varphi \rightarrow \langle\alpha\rangle\psi$ sii $\vdash_S \varphi \rightarrow \langle\alpha\rangle\psi$ para todo programa α y fórmulas de primer orden φ y ψ.

Como ya hemos dicho no puede existir un resultado similar para fórmulas de corrección parcial. Una primera solución a esta dificultad fue propuesta por Cook [Coo 78]. Esta solución relativiza el sistema a modelos expresivos.

COMPLETITUD RELATIVA DE LAS FORMULAS DE CORRECCION PARCIAL DE DL

Un modelo $\mathfrak{D}$ es expresivo con respecto a las aserciones de primer orden si para todo programa α y para toda fórmula de primer orden φ, existe una fórmula de primer orden ψ tal que $\mathfrak{D} \models \psi \leftrightarrow [\alpha]\varphi$.

Ejemplos de modelos expresivos para muchos lenguajes de programación son los modelos finitos y los modelos aritméticos. Esta última clase de modelos fue introducida por Moschovakis [Mos 74] bajo el nombre de modelos aceptables (ver también [Har 79]).

El siguiente resultado se debe a Cook y establece la completitud relativa de un cálculo finitario correcto para fórmulas de corrección parcial de DL.

Existe un sistema de deducción formal finitario S tal que para todo modelo expresivo $\mathfrak{D}$, $\mathfrak{D} \models \varphi \rightarrow [\alpha]\psi$ sii $Th(\mathfrak{D}) \vdash_S \varphi \rightarrow [\alpha]\psi$ para todo programa α y fórmulas de primer orden φ y ψ, donde $Th(\mathfrak{D}) = \{\varphi \ / \ \mathfrak{D} \models \varphi$ y φ de primer orden$\}$.

Una generalización de este resultado a fórmulas válidas de DL se obtiene restringiendo aún más los modelos. El siguiente resultado establece la completitud aritmética de un cálculo finitario correcto para fórmulas de DL.

COMPLETITUD ARITMETICA PARA FORMULAS DE DL

Un modelo $\mathfrak{D}$ es aritmético si el modelo estándar de la aritmética es definible en $\mathfrak{D}$ en primer orden, y $\mathfrak{D}$ posee funciones definibles en primer orden que permiten codificar y decodificar secuencias finitas de elementos del dominio de $\mathfrak{D}$.

El siguiente resultado se debe a D. Harel [Har 79] y establece la completitud aritmética de un cálculo finitario correcto para fórmulas de DL

Existe un sistema de deducción formal finitario S_a tal que para todo modelo aritmético $\mathfrak{D}$, $\mathfrak{D} \models \varphi$ sii $Th(\mathfrak{D}) \vdash_{S_a} \varphi$ para todo fórmula de DL φ.

Este cálulo se puede obtener a partir de un cálculo para fórmulas de corrección total extendiendolo con dos nuevas reglas.

Una extensa recapitulación sobre los resultados de completitud relativa para corrección parcial puede encontrarse también en [Apt 81].

COMPLETITUD INFINITARIA PARA DL

Otra solución al problema de la no existencia de cálculos finitarios correctos y completos para DL son los sistemas de deducción formal infinitarios. Un cálculo de estas caracteristicas fue propuesto por Mirkowska para la lógica algorítmica [Mir 71].

Existe un sistema de deducción formal infinitario S_ω tal que $\models \varphi$ sii $\vdash_{S_\omega} \varphi$ para toda fórmula de DL.

Los cálculos infinitarios para fórmulas válidas son una extensión de los cálculos finitarios para fórmulas de corrección total con dos nuevos axiomas y una regla infinitaria (con infinitas premisas).

Observese que una demostración en S_ω es una secuencia posiblemente infinita de fórmulas de DL, cada una de las cuales es una instancia de algún axioma o se ha obtenido de aplicar una regla a fórmulas que aparecen previamente en la secuencia.

CAPACIDAD DE EXPRESION DE DL

En DL son expresables la corrección parcial y total de programas, la equivalencia de programas y el determinismo de programas, por medio del operador de fórmulas diamante.

Por otro lado, dada la diversidad de lógicas de programas que existen, es importante también el estudio de la capacidad de expresión de estas lógicas, en el sentido de la siguiente definición

Dadas dos lógicas L y L' se dice que L' es al menos tan expresiva como L, y se denota por $L' \leq L$, si para toda fórmula $\varphi \in L$ existe una fórmula $\psi \in L'$ tal que para toda toda interpretación $(\mathfrak{D}, \delta)$, $(\mathfrak{D}, \delta) \models \varphi$ sii $(\mathfrak{D}, \delta) \models \psi$. Donde se supone que L y L' son lógicas sobre un mismo lenguaje de primer orden.

A partir de esta definición se definen $L < L'$ y $L \equiv L'$ de forma usual. El siguiente resultado establece la relación de expresividad entre DL, la lógica de primer orden L_1 y la lógica de primer orden infinitaria $L_{\omega_1\omega}$:

$L_1 \leq DL < L_{\omega_1\omega}$.

Este resultado puede encontrarse en [Har 79], donde además se clasifican las distintas versiones de la lógica dinámica, como DL con asignación sobre arrays, DL con asignación indeterminista, DL determinista, etc. Todas las variantes verifican el resultado anterior. Resultados en esta línea pueden encontrarse en [HAR 84] y [KT].

2.- LOGICAS NO ESTANDAR DE PROGRAMAS IMPERATIVOS

Andréka, Németi y Sain relacionaron la situación de DL con la situación de la lógica de orden superior, la cual es fuertemente incompleta con respecto a su teoría de modelos estándar. Henkin ideo una teoría de modelos no estándar para la lógica de orden superior que la convertía en completa con respecto a esta teoría de modelos. Esto mismo es lo que han hecho Andréka, Németi y Sain para DL con la lógica dinámica no estándar (NDL), encontrándose una buena recopilación de sus investigaciones en esta línea en [ANS 82].

En concreto, la necesidad de introducir módelos no estándar en lógicas de programas, es el hecho de que en las lógicas de programas estándar, el principio de Tarski sobre teoría de módelos no se verifica (cfr. [Paz 84]). Este principio, permite probar el hecho de que una fórmula no es demostrable formalmente, encontrando un módelo en el cual esta fórmula sea falsa. Si una lógica no verifica este principio, entonces para probar que una fórmula no es demostrable formalmente, hay que probar que ninguna de las posibles demostraciones formales es una demostración de esta fórmula. NDL verifica el principio de Tarski.

LA LOGICA DINAMICA NO ESTANDAR NDL

La principal diferencia entre DL y NDL son los modelos. La idea fundamental es interpretar el operador de iteración de forma distinta (no estándar) a la estándar, para conseguir que sea expresable mediante una fórmula de primer orden.

Para esto usan estructuras de tres géneros *d*, *t* e *i* con un símbolo de función binaria *ext*, de forma que un modelo consiste de tres partes disjuntas, los datos, el tiempo o números internos (con signatura aritmética) y las secuencias internas de datos (con signatura vacía) y

además *ext* asigna a cada secuencia *s* y número interno *n* el dato de la *n*-esima componente de *s*. Estos modelos permiten interpretar el operador de iteración codificando las secuencias de ejecución de los programas mediante las secuencias internas de datos.

Observese la relación entre estos modelos y los modelos aritméticos, en los que se exige que el tiempo sea el modelo estándar de la aritmética y que las secuencias sean finitas con respecto a esta aritmética.

COMPLETITUD PARA FORMULAS DE NDL

Existe un sistema de deducción formal finitario N tal que $\models\varphi$ sii $\vdash_N\varphi$ para toda fórmula de NDL.

En la primera demostración de este resultado no se da directamente un cálculo específico para NDL, sino que se reduce NDL, mediante una función de traducción computable total, a un lenguaje de primer orden de tres géneros completo y compacto. De hecho, esta traducción puede considerarse como la definición de la semántica de NDL.

De esta demostración también se deduce el siguiente resultado:

NDL es compacto.

En [Sai 84] se presenta un cálculo al estilo Hilbert con programas estructurados para NDL. Un sistema análogo para una versión no estándar de un género de DL que necesitaba los axiomas de la aritmética de Peano fue dado por Hajek [Haj 86].

NDL RAZONABLE

NDL resuelve el problema de incompletitud de DL y además, la semántica del operador de iteración, coincide con la estándar sobre aquellos modelos en los que el tiempo es la aritmética estándar. Sin embargo, muchos de los modelos de NDL no reflejan el significado intuitivo de la ejecución de un programa. Restringiendo estos modelos para que los números y secuencias internas se comporten adecuadamente, mediante la exigencia de que ciertos axiomas se verifiquen en ellos, definen RDL (lógica dinámica no estándar razonable, en [Sai 84]), como un marco adecuado para razonar sobre el comportamiento de los programas.

Estos axiomas son decidibles, y por tanto añadiendolos al cálculo anteriormente citado para NDL, se obtiene un cálculo finitario correcto y completo para RDL. Además, los modelos estándar del tiempo vefifican estos axiomas. T. Hortalá y M Rodríguez [HR 85] y [HR 89], estudian una lógica de Hoare para programas regulares indeterministas desde el punto de vista de la lógica dinámica no estándar. Se establece un teorema de completitud del cálculo de Hoare con respecto a la semántica continua.

Sain en [Sai 84] estudia la minimización de estos axiomas, en el sentido de obtener el conjunto de axiomas más débil que obligue a los modelos de NDL a ser razonables.

CAPACIDAD DE EXPRESION DE NDL

Además, NDL es un poderoso marco para caracterizar y comparar métodos de verificación de programas y sus resultados de completitud. Este tipo de resultados para el método de Floyd-Hoare y la lógica temporal de Pnueli para corrección parcial de programas, el método de Manna-Cooper para corrección total de programas [Man 74], y la axiomatización aritmética de Harel para DL, pueden encontrarse en [Paz 86]. La técnica para la caracterización de un método M para razonar sobre el comportamiento de programas en NDL, consiste en encontrar un conjunto de axiomas Ax decidible, de forma que las fórmulas demostrables formalmente en M sean exactamente aquellas que lo son en NDL a partir de Ax.

Posteriormente Sain en [Sai 87] demuestra que la terminación (corrección total) de programas es un concepto de primer orden vía lógicas no estándar de programas, en contraste con el resultado de Kfoury-Park [KP 75]. A la vez demuestra que NDL es estrictamente más potente que el método para corrección total de programas de Manna-Cooper [Man 74].

Para finalizar esta sección, decir que existen otras lógicas no estándar de programas propuestas por otros autores como Berman [Ber 83], Cartwright [Car 84], Hajek [Haj 86], etc., pero estan menos relacionadas con este trabajo.

3.- LOGICAS DE PROGRAMAS FUNCIONALES

Como ya hemos dicho, los lenguajes de programción funcional están ganando popularidad por varias razones. En esta sección queremos exponer las características de estos lenguajes y sus diferencias con los imperativos, así como los precedentes conocidos en lógicas de programas funcionales.

EVOLUCION DE LOS LENGUAJES DE PROGRAMACION

Existe una gran proliferación de lenguajes de programación tanto de propósito específico como general, debido fundamentalmente a dos razones: optimizar la eficiencia de los programas por medio de lenguajes orientados a la máquina, en el sentido de que es el programador quien, haciendo uso de las intrucciones de control, debe hacer estos programas eficientes, y optimizar la sencillez, elegancia y claridad de los programas, por medio de lenguajes de alto nivel orientados al programador.

Los distintos lenguajes surgen del estudio de los problemas existentes tanto de Hardware como de Software, como posibles soluciones. Hagamos un breve repaso a los problemas que han ido apareciendo y a las soluciones propuestas.

PROBLEMAS DE HARDWARE

Hoy en día los procesadores son rápidos y proporcionalmente baratos gracias a las tecnologías LSI y VLSI, que se usan en la construcción tanto de la memoria como de los procesadores. Sin embargo todavía no se ha explotado suficientemente el procesamiento real en paralelo. Algunas máquinas paralelas están diseñadas para problemas específicos que son

especialmente sencillos de tratar, como las que consisten en un array de procesadores cada uno de los cuales realiza la misma tarea para distintos datos. Pero el paralelismo consistente en una tubería de procesadores, de forma que cada uno realice una parte del cómputo sincronizado con los demás procesadores no es fácil de explotar, puesto que no todos los problemas son fácilmente descomponibles en esta forma.

Una alternativa a este problema son los lenguajes que explícitamente controlen la ejecución paralela de los procesos, como Modula2 y ADA. Normalmente estos lenguajes se usan con un solo procesador que comparten los distintos procesos concurrentes, pero no hay motivo para que no se usen con varios procesadores paralelos, sobre todo cuando hay pocas tareas que se realizan a la vez.

Lo ideal sería un lenguaje en el que se puedan escribir problemas complejos de forma sencilla y que posteriormente, de forma automática, se descomponga en procesos concurrentes. De esta forma, se puede ir aumentando la velocidad sencillamente aumentando el número de procesadores, sin necesidad de especificar el control sobre ellos.

PROBLEMAS DE SOFTWARE

En este campo, muchas investigaciones se han dirigido a la productividad de los programadores, especialmente cuando se descubrió que el costo de emplear un equipo de programadores era mayor que el de obtención y mantenimiento del ordenador. En este sentido, se pueden distinguir dos etapas: (A) la producción propiamente del programa y (B) su mantenimiento.

(A) Se comprobó que independientemente del lenguaje de programación usado, el número de líneas de código producidas (escritas, comprobadas, corregidas y documentadas) por un programador era el mismo. De lo que se concluyó que cuanto más potente sea el lenguaje que se use mayor será la productividad. Esto motivó el estudio de aquellos elementos de un lenguaje que sean más propensos a provocar errores en el programador para poder así eliminarlos

en un posterior diseño del lenguaje. Entre estos elementos se encuentran:

- El acceso directo a la memoria, que produjo el salto de los lenguajes ensamblador a los lenguajes de alto nivel, que exigen una organización de los datos y por tanto del programa.
- Las instrucciones de salto incondicional (goto), que da lugar a los lenguajes estructurados.
- Las variables globales, que producen efectos colaterales, que dio lugar a los lenguajes modulares con paso de parámetros.

(B) En segundo lugar se comprobó que un 50% del esfuerzo de los programadores se invertía en el mantenimiento de los programas. De lo que se deduce que puede ganarse productividad por medio de especificaciones más precisas de los programas y con lenguajes más cercanos a las especificaciones. En este punto hay que resaltar la importancia de la especificación del problema previa al programa. Una herramienta para hacer las especificaciones más precisas es realizarlas en un lenguaje no ambiguo como el matemático: métodos matemáticos de descripción y comprobación.

La mayor complicación en el uso de estos métodos es el operador de asignación. La asignación distancia el concepto de variable matemática del de variable de un programa. Una variable en un programa con asignación denota una localización en la memoria cuyo contenido (valor) puede modificarse mediante asignaciones, de forma que una variable tiene asociada su historia de cómputo, y para conocer su valor hay que saber exactamente el punto de ejecución del programa. En contraste con esto, las variables matemáticas tiene un valor o están indefinidas sí todavía no se ha computado este: esta propiedad se conoce como *Transparencia Referencial* y a los lenguajes que la poseen *Lenguajes Declarativos* (sin asignación).

La transparencia referencial no solo elimina el distanciamento entre las variables de programa y las matemáticas, sino que además elimina la secuencialidad explícita del código, lo que permite procesamientos en paralelo.

LENGUAJES DE PROGRAMACION FUNCIONALES

Los lenguajes de programación funcionales (o aplicativos) son lenguajes declarativos que surgen del λ-cálculo y cuya componente fundamental es la función. El primer lenguaje funcional LISP se debe a McCarthy [McC 60] que lo propuso fundamentalmente para su uso en el campo de la inteligencia artificial como herramienta para el procesamiento simbólico. Posteriormente han surgido otros lenguajes funcionales como FP diseñado por Backus en 1977, HOPE diseñado por MacQueen, Burstall y Sannella en 1980 y ML que fue diseñado como meta lenguaje para LCF en 1974 [GMW 79]. Los lenguajes declarativos en los que la componente fundamental es el predicado se denominan normalmente lenguajes lógicos, pero caen fuera del alcance de este trabajo.

En los lenguajes funcionales la transparencia referencial tiene otras implicaciones además de las expuestas. En primer lugar la fusión de los conceptos de variable (dato) y función (código), ya que una variable indefinida no es más que una llamada a una función sin evaluar. Esta fusión produce de forma natural las funciones de orden superior. En segundo lugar, las estructuras de datos estáticas, como los arrays, etc, deben reemplazarse por estructuras de datos dinámicas, puesto que los datos se van guardando a medida que se van produciendo. Algunos lenguajes de programación como Pascal y C implementan las estructuras dinámicas de datos por medio de punteros, mientras que en los lenguajes funcionales puede hacerse de forma abstracta.

Inicialmente, los lenguajes funcionales se han ejecutado en ordenadores convencionales, diseñados para los lenguajes imperativos, por lo que se les ha acusado de lentos. Pero ahora ya han surgido diseños de harware especialmente para los lenguajes funcionales, basados en el λ-cálculo o en la teoría de los combinadores, diseñadas para un código intermedio al que previamente hay que reducir el lenguaje de alto nivel. La Máquina abstracta SECD, debida a Landin [Lan 63], se basa en la implementación de β-reducciones del λ-cálculo. La máquina SK, debida a Turner [Tur 79], se basa en la implementación de los grafos de reducción que representan la

definición de una función en términos de los combinadores S y K. Una descripción detallada de la máquina SECD puede encontrarse en [Hen 80] y de la máquina SK en [Pey 87].

En resumen, los lenguajes funcionales reunen las siguientes características (cfr. [Els 87]):

1- Acercamiento del código del programa a la abstración de los lenguajes de especificación y métodos matemáticos de comprobación.
2- Acercamiento a la realización de procesamiento paralelo real.
3- Fusión de los conceptos de código y datos.
4- Implementación de tipos abstractos de datos polimórficos.

LOGICAS DE PROGRAMAS FUNCIONALES

Hasta el momento hay pocas investigaciones sobre lógicas de programas funcionales, las principales referencias de las que disponemos son: Cartwright [Car 83], [Car 84], Cartwright y McCarthy [CM 79], Goerdt [Goe 85], [Goe 87] y Pazstor [Paz 86].

Dentro del marco de las lógicas de Hoare, Goerdt [Goe 85], [Goe 87] ha obtenido un cálculo al estilo de Hoare correcto y relativamente completo para el $\lambda\mu$-cálculo con tipos. Pero su trabajo se restringe a la corrección parcial de programas.

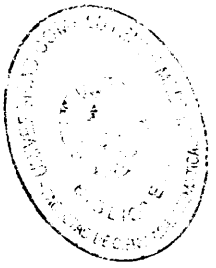
Las otras investigaciones están en la línea no estándar de las lógicas de programas. Cartwright y McCarthy [CM 79] [Car 83], [Car 84] han considerado sistemas de definiciones de funciones mutuamente recursivas en el marco de la lógica de primer orden. Han demostrado que la lógica de primer orden es correcta y completa con respecto a una semántica no estándar para razonar sobre el comportamiento de funciones definidas recursivamente, basandose en el menor punto fijo definible. La existencia de este menor punto fijo definible se garantiza por medio de técnicas de codificación complejas.

Pazstor [Paz 86] ha investigado una lógica dinámica para programas recursivos no estándar, usando estructuras de dos géneros. Da una semántica del menor punto fijo definible para los programas recursivos y prueba la existencia de un cálculo finitario correcto y completo para su lenguaje dinámico. Sus programas están restringidos de forma que solo se permiten funciones recursivas unarias y en su formalismo no es expresable ni la composición ni la recursión mutua.

Ni Cartwright y McCarthy ni Pazstor proveen de facilidades sintácticas para construir programas funcionales estructurados y para razonar sobre su comportamiento mediante composicionalidad. La cuestión de la composicionalidad ha sido un punto importante en muchas investigaciones sobre métodos formales de verificación de programas (cfr. [Roe 85]).

Por último decir que la conocida lógica para funciones computables LCF [GMW 79] está orientada a un entorno de demostraciones interactivo que no se trata en este trabajo.

A partir de estos precedentes y motivaciones, creemos que el estudio de axiomas y reglas composicionales específicas para el comportamiennto de operadores como: λ-abstracción, aplicación funcional y μ-operador, tienen cierto interés metodológico y puede ayudar a comprender mejor la lógica para las funciones definidas recursivamente. Por otro lado pensamos en emplazar la investigación dentro de la lógica no estándar de Andréka, Németi y Sain [ANS 82], puesto que provee de un marco lo suficientemente potente como para comprender los fenomenos de incompletitud y caracterizar y comparar métodos de verificación de programas, como hemos expuesto en este capítulo.



CAPITULO 2:
LA LOGICA LRF PARA PROGRAMAS FUNCIONALES

En este capítulo presentamos la lógica LRF para programas funcionales. El lenguaje de programación abstracto asociado a LRF, permite funciones indeterministas, llamadas a funciones predefinidas y expresar λ-abstracciones, condicionales y definiciones mutuamente recursivas o μ-recursión. La sintaxis de este lenguaje de programación está orientada a la simplificación y claridad de la notación necesaria a lo largo de este trabajo, y no a su uso como lenguaje de programación. Las fórmulas de LRF están formadas como las de primer orden junto con un nuevo símbolo lógico para expresar la pertenencia de un elemento o término al conjunto de valores asociado a una expresión o programa no determinista.

En la sección 1, usando una técnica inspirada por Meyer y Mitchell [MM 83], extendemos la lógica de primer de orden con facilidades para razonar sobre funciones predefinidas e indeterministas.

En la sección 2, se hace un breve estudio del dominio de las funciones indeterministas, donde aparecen conceptos y resultados necesarios para el estudio semántico de las funciones definidas recursivamente. En particular, damos una caracterización del menor punto fijo para funciones indeterministas recursivas, que será parte esencial en los cálculos finitarios para LRF.

En la sección 3 definimos la sintaxis y la semántica de LRF. De hecho, se presentan tres variantes sintácticas de LRF cada una con distintas propiedades sobre el comportamiento semántico de las funciones.

Finalmente, en la sección 4 hacemos un estudio de la capacidad de expresión de LRF comparándola con otras lógicas, como la lógica dinámica de primer orden para programas regulares de Harel, el μ-cálculo proposicional de Kozen y la lógica infinitaria de primer orden. Y además comprobamos, adaptando un resultado de Kfoury y Park [KP 75], que las propiedades de terminación de programas no triviales en el marco de LRF no son de primer orden.

1.- UNA LOGICA DE PRIMER ORDEN EXTENDIDA

Presentamos una extensión sintáctica de la lógica de primer orden clásica (que denotamos por EL_1) obtenida al añadir un nuevo tipo de variables para las funciones predefinidas, que no pueden ser cuantificadas, y un nuevo símbolo lógico $\ni$, para expresar la posibilidad de que un individuo sea valor de una función indeterminista.

La sintaxis y la semántica son muy similares a la lógica de primer orden clásica. De hecho esta extensión no es necesaria, ya que en su lugar podríamos asociar a las funciones indeterministas símbolos de predicado para sus grafos. La conveniencia de definir esta extensión de la lógica de primer orden clásica, es precisamente adecuarla al uso que vamos a hacer de ella en este trabajo. Una técnica similar es usada por Meyer y Mitchell en [MM 83].

1- DEFINICION *(Sintaxis)*

Una signatura de primer orden $\Sigma = \Sigma_{fn} \cup \Sigma_{pd}$ está formada por un conjunto Σ_{fn} de símbolos de función con aridad asociada, f, g, etc. y un conjunto Σ_{pd} de símbolos de predicado con aridad asociada, p, q, etc. Ambos conjuntos de cardinal a lo sumo numerable. Los símbolos de función, resp. símbolos de predicado, 0-arios se consideran de forma usual como símbolos de constantes, resp. símbolos de proposición.

Un conjunto de variables $V = VAR \cup FVAR$ consta de dos conjuntos disjuntos: VAR para las variables de individuo, (o datos) x,y,z y FVAR para las variables de función con aridad asociada, X,Y,Z. Consideramos una cantidad numerable de variables de individuo y una cantidad numerable de variables de función para cada posible aridad.

Los Σ-términos ($t \in TER_\Sigma$) y fórmulas ($E \in EFOR_\Sigma$) de EL_1 se definen mediante las siguientes reglas BNF:

$t ::= x \mid f(t_1,\ldots,t_k)$

$E ::= p(t_1,\ldots,t_m) \mid (t_1=t_2) \mid X(t_1,\ldots,t_n)\ni t \mid \neg E_1 \mid (E_1 \vee E_2) \mid \exists x E_1$

donde la aridad asociada a los símbolos f, p y X es k, m y n respectivamente.

A los tres primeros tipos de fórmulas las llamamos atómicas. ■

Siempre supondremos que las variables de función y los símbolos de función y predicado están usados con su correspondiente aridad. Además usaremos las abreviaturas usuales para los símbolos lógicos no primitivos y omitiremos paréntesis en las fórmulas de acuerdo con los convenios estándar.

El concepto de variable libre es análogo al de la lógica de primer orden clásica para las variables de individuo. Además en esta extensión toda aparición de una variable funcional es libre. Denotamos al conjunto de variables libres de Θ por $Lib(\Theta)\subseteq V$, donde Θ es un término o una fórmula.

La sustitución de una variable de individuo x por un término t en Θ (donde Θ es un término o una fórmula) se define como en la lógica de primer orden clásica, y lo denotamos por $\Theta[t/x]$. Ambos conceptos serán definidos en la sección 3 para la lógica LRF que incluye como fragmento propio a EL_1.

2- DEFINICION *(Estructuras, Valoraciones, Contextos)*

Dado un conjunto no vacio D y una aridad $n\geq 0$, consideramos los conjuntos $\mathfrak{F}_D^{(n)}$ de las funciones n-arias sobre D, $\mathfrak{NF}_D^{(n)}$ de las funciones indeterministas sobre D y $\mathfrak{P}_D^{(n)}$ de los predicados n-arios, definidos por:

$\mathfrak{F}_D^{(n)} = \{ f \mid f:D^n \to D \}$

$\mathfrak{NF}_D^{(n)} = \{ X \mid X:D^n \to \mathfrak{P}(D) \}$

$\mathfrak{P}_D^{(n)} = \{ p \mid p:D^n \to \{true,false\} \}$.

(a) Una Σ-estructura es un sistema de la forma

$$\mathfrak{D} = \langle D,\ (f^{\mathfrak{D}})_{f\in\Sigma_{fn}},\ (p^{\mathfrak{D}})_{p\in\Sigma_{pd}}\rangle$$

donde D es un conjunto no vacío, $f^{\mathfrak{D}}\in\mathfrak{F}_D^{(n)}$ and $p^{\mathfrak{D}}\in\mathfrak{P}_D^{(m)}$, siendo n y m su respectiva aridad.

(b) Una valoración sobre $\mathfrak{D}$ es una aplicación $\delta: VAR \longrightarrow D$. Para $x\in D$ y $x\in VAR$, denotamos por $\delta[x/x]$ a la valoración δ' tal que $\delta'(x)=x$ y $\delta'(y)=\delta(y)$ para toda variable $y\neq x$.

(c) Un contexto sobre $\mathfrak{D}$ es una aplicación $\Delta: FVAR \longrightarrow \cup_{n\in\mathbb{N}}\mathcal{P}\mathfrak{F}_D^{(n)}$ que preserva las aridades. Para $X\in FVAR$ y $X\in\mathcal{P}\mathfrak{F}_D^{(n)}$, $\Delta[X/X]$ se define similarmente a $\delta[x/x]$ en (b). ■

3- DEFINICION *(Semántica Denotacional)*

Sea $\mathfrak{D}$ una Σ-estructura, δ una valoración sobre $\mathfrak{D}$ y Δ un contexto sobre $\mathfrak{D}$. El valor de los términos (denotado por $t^{\mathfrak{D}}(\delta)\in D$) y de las fórmulas de primer orden extendido (denotado por $E^{\mathfrak{D}}(\Delta,\delta)\in\{true,\ false\}$) se define recursivamente como sigue:

$$x^{\mathfrak{D}}(\delta) = \delta(x)$$

$$f(t_1,\ldots,t_n)^{\mathfrak{D}}(\delta) = f^{\mathfrak{D}}(t_1^{\mathfrak{D}}(\delta),\ldots,t_n^{\mathfrak{D}}(\delta))$$

$$p(t_1,\ldots,t_n)^{\mathfrak{D}}(\delta) = p^{\mathfrak{D}}(t_1^{\mathfrak{D}}(\delta),\ldots,t_n^{\mathfrak{D}}(\delta))$$

$$(t_1 = t_2)^{\mathfrak{D}}(\delta) = \begin{cases} true & \text{si } t_1^{\mathfrak{D}}(\delta) = t_2^{\mathfrak{D}}(\delta) \\ false & \text{en otro caso} \end{cases}$$

$$(X(t_1,\ldots,t_n)\ni t)^{\mathfrak{D}}(\Delta,\delta) = \begin{cases} true & \text{si } t^{\mathfrak{D}}(\delta)\in\Delta(X)(t_1^{\mathfrak{D}}(\delta),\ldots,t_n^{\mathfrak{D}}(\delta)) \\ false & \text{en otro caso} \end{cases}$$

$$(\neg E_1)^{\mathfrak{D}}(\Delta,\delta) = \begin{cases} false & \text{si } E_1^{\mathfrak{D}}(\Delta,\delta)=true \\ true & \text{en otro caso} \end{cases}$$

$$(E_1 \vee E_2)^{\mathfrak{D}}(\Delta,\delta) = \begin{cases} \text{false} & \text{si } E_1^{\mathfrak{D}}(\Delta,\delta)=E_2^{\mathfrak{D}}(\Delta,\delta)=\text{false} \\ \text{true} & \text{en otro caso} \end{cases}$$

$$(\exists x E_1)^{\mathfrak{D}}(\Delta,\delta) = \begin{cases} \text{true} & \text{si } E_1^{\mathfrak{D}}(\Delta,\delta[x/x])=\text{true para alguna } x\in D \\ \text{false} & \text{en otro caso} \end{cases}$$ ■

4- DEFINICION *(Satisfactibilidad, Validez, Consecuencia Lógica)*

Para $\mathfrak{C}\cup\{E\}\subseteq EFOR_\Sigma$:

(a) $\mathfrak{D}$ satisface E bajo Δ,δ (en símbolos: $\mathfrak{D} \models E(\Delta,\delta)$) si $E^{\mathfrak{D}}(\Delta,\delta)=true$.

(b) $\mathfrak{D}$ satisface $\mathfrak{C}$ bajo Δ,δ (en símbolos: $\mathfrak{D} \models \mathfrak{C}(\Delta,\delta)$) si $\mathfrak{D} \models E(\Delta,\delta)$ para toda $E\in\mathfrak{C}$.

(c) E (resp. $\mathfrak{C}$) es satisfactible si $\mathfrak{D} \models E(\Delta,\delta)$ (resp. $\mathfrak{D} \models \mathfrak{C}(\Delta,\delta)$) para algún $\mathfrak{D}$, Δ, δ.

(d) E es consecuencia lógica de $\mathfrak{C}$ (en símbolos: $\mathfrak{C}\models E$) si $\mathfrak{D}\models\mathfrak{C}$ implica $\mathfrak{D}\models E$ para toda posible $\mathfrak{D}$.

(e) E es lógicamente válida (en símbolos: $\models E$) si $\varnothing \models E$ ■

Los siguientes resultados son adaptaciones de sus análogos en lógica de primer orden clásica (pueden encontrarse por ejemplo en [Smu 86], [EFT 84], por lo que omitimos sus demostraciones.

5- LEMA *(Sustitución)*

Dada una Σ-estructura $\mathfrak{D}$, $x\in VAR$ y $t\in TER_\Sigma$. Para toda $\Theta\in TER_\Sigma$, resp. $\Theta\in EFOR_\Sigma$, valoración δ y contexto Δ sobre $\mathfrak{D}$:

$$(\Theta[t/x])^{\mathfrak{D}}(\Delta,\delta) = \Theta^{\mathfrak{D}}(\Delta,\delta[t^{\mathfrak{D}}(\delta)/x])$$

Demostración

Por inducción sobre la estructura de los términos y las fórmulas. ■

6- TEOREMA *(Cálculo Correcto y Completo)*

La lógica de primer orden extendida tiene un cálculo finitario correcto y completo.

Demostración

Cualquier cálculo correcto y completo para la lógica de primer orden clásica puede ser transformado, mediante pequeñas adaptaciones sintácticas, en un cálculo correcto y completo para EL_1. En la sección 1 del capítulo 3 damos un cálculo para la lógica LRF, la cual incluye como fragmento a la lógica de primer orden extendida, por esto no especificamos ahora ningún cálculo concreto. La demostración de completitud se reduce, de forma usual, a demostrar la satisfactibilidad de los conjuntos consistentes de fórmulas. Esto puede hacerse mediante la técnica de los conjuntos de Hintikka y el método de los tableaux. (cfr. [Smu 86]). En la demostración de satisfactibilidad de un conjunto de Hintikka se construye una estructura $\mathfrak{D}$ a partir del conjunto cociente de los términos, una valoración y un contexto adecuados. La interpretación de una variable funcional $\Delta(X)$, se define en relación con la pertenencia de las fórmulas $X(t_1,\ldots,t_n)\partial t$ al conjunto de Hintikka. De hecho, esta definición es la análoga a la lógica de primer orden clásica, si las variables funcionales de aridad n son vistas como símbolos de relación de aridad n+1. La técnica de los tableaux y los conjuntos de Hintikka será usada en la sección 1 del capítulo 3 para LRF. ■

Como corolario de este teorema se deduce que EL_1 satisface el teorema de compacidad y el de Löwenheim-Skolem.

7- TEOREMA *(Compacidad)*

Un conjunto de fórmulas $\mathfrak{C}$ de EL_1 es satisfactible sii todo subconjunto finito de $\mathfrak{C}$ es satisfactible. ■

8- TEOREMA *(Löwenheim-Skolem)*

Una fórmula de EL_1 satisfactible tiene un modelo de dominio a lo más numerable. ■

Estos resultados recalcan que la lógica de primer orden extendida tiene el mismo poder de expresión que la lógica de primer orden clásica.

2.- EL RETICULO DE LAS FUNCIONES INDETERMINISTAS: CARACTERIZACIONES DEL MENOR PUNTO FIJO

Como hemos dicho, el lenguaje funcional de LRF permite definiciones recursivas de funciones indeterministas. Por esto, en su semántica haremos uso del concepto de menor punto fijo. En particular, nos interesa conocer las condiciones de existencia del menor punto fijo de un operador $T: \mathfrak{N}\mathfrak{F}_D^{(n)} \longrightarrow \mathfrak{N}\mathfrak{F}_D^{(n)}$.

Además, los cálculos que presentamos para LRF en este trabajo, están basados en dos posibles caracterizaciones del menor punto fijo de un operador continuo T.

Para esto, en primer lugar definimos un orden parcial $\ll$ sobre $\mathfrak{N}\mathfrak{F}_D^{(n)}$, y demostramos que $(\mathfrak{N}\mathfrak{F}_D^{(n)}, \ll)$ es un retículo algebraico y completo (cfr. [Sto 77]).

1- DEFINICION *(Orden Parcial)*

Sean $X, Y \in \mathfrak{N}\mathfrak{F}_D^{(n)}$:

(a) $X \ll Y$ sii para todo $z_1, \ldots, z_n \in D$: $X(z_1, \ldots, z_n) \subseteq Y(z_1, \ldots, z_n)$.

(b) Un conjunto $\mathcal{X} \subseteq \mathfrak{N}\mathfrak{F}_D^{(n)}$ es dirigido si para todo $X, Y \in \mathcal{X}$ existe $Z \in \mathcal{X}$ tal que $X \ll Z$ e $Y \ll Z$.

(c) $grafo(X) = \{\langle x_1, \ldots, x_n, y\rangle \in D^{n+1} \ / y \in X(\vec{x})\}$. ■

2- LEMA *(Retículo de las Funciones Indeterministas)*

$(\mathfrak{N}\mathfrak{F}_D^{(n)}, \ll)$, es un retículo algebraico y completo. En concreto, se verifica:

(a) $X \sqsubset Y$ sii grafo(X) $\subseteq$ grafo(Y).

(b) El elemento mínimo $\perp_D^{(n)}$ de $\Pi\mathfrak{I}_D^{(n)}$ corresponde al grafo vacío, y lo llamaremos función indefinida.

(c) Los elementos finitos de $\Pi\mathfrak{I}_D^{(n)}$ son precisamente aquellas funciones cuyos grafos son finitos, y los llamamos funciones finitas.

(d) El supremo $\sqcup\mathcal{X}$ de un subconjunto $\mathcal{X}\subseteq\Pi\mathfrak{I}_D^{(n)}$ tiene como grafo la unión conjuntista de la familia { grafo(X) / $X\in\mathcal{X}$ }.
Dualmente, el ínfimo $\sqcap\mathcal{X}$ tiene como grafo la intersección de la misma familia de conjuntos.

Demostración

(a), (b) y (d) se demuestran fácilmente. Para (c), recordar que un elemento $X\in\Pi\mathfrak{I}_D^{(n)}$ es finito si para todo conjunto dirigido $\mathcal{X}\in\Pi\mathfrak{I}_D^{(n)}$ tal que $X\sqsubset\sqcup\mathcal{X}$ existe $\chi\in\mathcal{X}$ con $X\sqsubset\chi$. Es fácil demostrar que las funciones finitas son los elementos finitos y que estos forman base, trabajando con los grafos. ■

Este lema nos permitirá aplicar el conocido resultado de Knaster y Tarski [Tar 55] sobre el menor punto fijo de aplicaciones monótonas y la caracterización de Kleene [Kle 67] para continuas sobre retículos completos. Pero además, obtendremos otra caracterización del menor punto fijo para operadores continuos sobre $\Pi\mathfrak{I}^{(n)}$, a la que también queremos dar significado para el caso de operadores monótonos sobre $\Pi\mathfrak{I}^{(n)}$. Para este proposito introducimos las siguientes definiciones y propiedades.

3- DEFINICION

Sea $T:\Pi\mathfrak{I}_D^{(n)} \longrightarrow \Pi\mathfrak{I}_D^{(n)}$

(a) T es monótono si para toda $X,Y\in\Pi\mathfrak{I}_D^{(n)}$ tal que $X\sqsubset Y$: $T(X)\sqsubset T(Y)$.

(b) T es continuo si para todo $\mathcal{X}\subseteq\Pi\mathfrak{I}_D^{(n)}$ dirigido

$$T(\sqcup\mathcal{X})=\sqcup\{\ T(X)\ /\ X\in\mathcal{X}\ \}$$ ■

4- DEFINICION

A todo operador $T:\mathfrak{N}\mathfrak{I}_D^{(n)} \longrightarrow \mathfrak{N}\mathfrak{I}_D^{(n)}$ le asociamos el operador denotado por $\tilde{T}:\mathfrak{N}\mathfrak{I}_D^{(n)} \longrightarrow \mathfrak{N}\mathfrak{I}_D^{(n)}$ definido por:

$$\tilde{T}(X)=\sqcup\{T(X_0) \;/\; X_0 \ll X \text{ finita}\} \text{ para toda } X\in\mathfrak{N}\mathfrak{I}^{(n)}. \quad \blacksquare$$

5- PROPIEDADES

Dado un operador $T:\mathfrak{N}\mathfrak{I}_D^{(n)} \longrightarrow \mathfrak{N}\mathfrak{I}_D^{(n)}$ monótono:

(i) $\tilde{\tilde{T}}=\tilde{T}$ (Idempotente).
(ii) $\tilde{T}$ es monótono
(iii) T es continuo <=> $T=\tilde{T}$
(iv) $T(Z)=\tilde{T}(Z)$ para toda $Z\in\mathfrak{N}\mathfrak{I}_D^{(n)}$ finita ■

6- DEFINICION

Dado un operador $T:\mathfrak{N}\mathfrak{I}_D^{(n)} \longrightarrow \mathfrak{N}\mathfrak{I}_D^{(n)}$, una función $Y\in\mathfrak{N}\mathfrak{I}_D^{(n)}$ es:

(a) El casi menor punto fijo de T, denotado por fic(T), si verifica las siguientes dos condiciones:

$\tilde{T}(Y)\leq Y\leq T(Y)$
$Y\ll Z$ para toda $Z\in\mathfrak{N}\mathfrak{I}_D^{(n)}$ y tal que $T(Z)=Z$

(b) El menor punto fijo de T, denotado por fix(T), si verifica las siguientes dos condiciones:

$T(Y)=Y$
$Y\ll Z$ para toda $Z\in\mathfrak{N}\mathfrak{I}_D^{(n)}$ y tal que $T(Z)=Z$ ■

7- LEMA

Dado $T:\mathfrak{N}\mathfrak{J}_D^{(n)} \rightarrow \mathfrak{N}\mathfrak{J}_D^{(n)}$ se verifica:

(a) Si T es monótono, el menor punto fijo de T, fix(T), existe.

(b) Además, si T es continuo fix(T) se puede caracterizar como el supremo de la cadena creciente de aproximaciones obtenida al iterar las aplicaciones del operador, empezando por la función indefinida:

$$fix(T)= \bigsqcup_{i\in\mathbb{N}} X_i \quad \text{donde } X_0=T^0(\perp)=\perp_D^{(n)},\ X_{i+1}=T^{i+1}(\perp)=T(X_i)$$

Llamaremos aproximación i-ésima de T a $T^i(\perp)$.

Demostración

Como $\mathfrak{N}\mathfrak{J}_D^{(n)}$ es un retículo algebraico completo, basta con aplicar para (a) el teorema de Knaster y Tarski [KT 55], y para (b) el teorema de Kleene [Kle 67]. ■

Por tanto, para $\vec{x}\in D^{(n)}$ $fix(T)(\vec{x})= (\bigcup_{i\in\mathbb{N}} T^i(\perp))(\vec{x})= \bigcup_{i\in\mathbb{N}}(T^i(\perp)(\vec{x}))$, es decir, $y\in fix(T)(\vec{x})$ sii existe $j\in\mathbb{N}$ tal que $y\in T^j(\perp)(\vec{x})$. Sin embargo obsérvese que al estar trabajando con funciones indeterministas, aunque el dominio de la función $T^i(\perp)$ fuese finito, esta función puede ser infinita (ver el ejemplo de abajo). Por tanto, no podremos construir la tupla $\langle\vec{x},y\rangle$ de $T^j(\perp)$ finitamente. De ahí que, esta caracterización no nos sea suficiente para obtener cálculos finitarios, aunque sí lo será para obtener cálculos infinitarios para LRF, como veremos en el capítulo 3.

Por último vamos a introducir otra caracterización para el menor punto fijo de un operador continuo $T:\mathfrak{N}\mathfrak{J}_D^{(n)}\rightarrow\mathfrak{N}\mathfrak{J}_D^{(n)}$, ya mencionada. Esta caracterización será fundamental en la obtención de cálculos finitarios para LRF en el cápitulo 3, y para definir la semántica no estándar de las funciones de LRF definidas recursivamente en el cápitulo 4.

8- DEFINICION *(T-Derivación)*

(a) Dado un operador $T:\mathfrak{N}\mathfrak{F}_D^{(n)}\to\mathfrak{N}\mathfrak{F}_D^{(n)}$, decimos que una (n+1)-tupla $\langle\vec{s},s\rangle=\langle s_1,\ldots,s_n,s\rangle$ de secuencias finitas de elementos de D:

$$s_i=\langle s_i(0),\ldots,s_i(k)\rangle\in D^{k+1} \quad (1\le i\le n)$$

$$s=\langle s(0),\ldots,s(k)\rangle\in D^{k+1}$$

todas de la misma longitud k+1 ($k\ge 0$), es una T-derivación de longitud k+1 si para todo $0\le j\le k$:

$$s(j)\in T(fun_j(\langle\vec{s},s\rangle))(s_1(j),\ldots,s_n(j)),$$

donde $fun_j(\langle\vec{s},s\rangle)\in\mathfrak{N}\mathfrak{F}^{(n)}$ es la función finita cuyo grafo es el conjunto $\{\langle s_1(1),\ldots,s_n(1),s(1)\rangle \;/\; 0\le 1<j\}$.

(b) Dado un operador $T:\mathfrak{N}\mathfrak{F}_D^{(n)}\to\mathfrak{N}\mathfrak{F}_D^{(n)}$, decimos que una (n+1)-tupla $\langle\vec{x},y\rangle=\langle x_1,\ldots,x_n,y\rangle\in D^{n+1}$ es producida por la T-derivación $\langle\vec{s},s\rangle$ de longitud k+1 si:

$$\langle\vec{x},y\rangle = \langle\vec{s}(k),s(k)\rangle. \qquad \blacksquare$$

9- DEFINICION

Dado un operador $T:\mathfrak{N}\mathfrak{F}_D^{(n)}\to\mathfrak{N}\mathfrak{F}_D^{(n)}$ denotamos por $\min(T)\in\mathfrak{N}\mathfrak{F}_D^{(n)}$ a la función cuyo grafo es el conjunto de todas las (n+1)-tuplas $\langle\vec{x},y\rangle\in D^{n+1}$ producidas por alguna T-derivación.

10- LEMA

Si $T:\mathfrak{N}\mathfrak{F}_D^{(n)}\to\mathfrak{N}\mathfrak{F}_D^{(n)}$ es monótono, entonces min(T) verifica las siguientes condiciones:

(i) $\min(T)=fix(\tilde{T})$
(ii) $\min(T)\le Z$ para toda $Z\in\mathfrak{N}\mathfrak{F}_D^{(n)}$ y tal que $\tilde{T}(Z)\le Z$
(iii) $\tilde{T}(\min(T))\le\min(T)\le T(\min(T))$
(iv) $\min(T)\le Z$ para toda $Z\in\mathfrak{N}\mathfrak{F}_D^{(n)}$ y tal que $T(Z)\le Z$

Demostración

(i) Como $\bar{T}$ es continuo (por la propiedades S (i)-(iii)), sabemos que

$$fix(\bar{T}) = \bigsqcup_{i\in N} X_i \text{ donde } X_0 = \perp_D^{(n)},\ X_{i+1} = \bar{T}(X_i).$$

Veamos las dos inclusiones de la igualdad.

$\underline{fix(\bar{T}) \subseteq min(T)}$: Si $\langle \vec{x}, y\rangle \in grafo(fix(\bar{T}))$ entonces $\langle \vec{x}, y\rangle \in grafo(X_i)$ para algún $i<\omega$. Procedemos por inducción sobre i para demostrar que $\langle \vec{x}, y\rangle$ es producida por alguna T-derivación.

El caso i=0 se cumple, puesto que el $grafo(X_0)=\emptyset$.

Supongamos i>0. Entonces $y \in \bar{T}(X_{i-1})(\vec{x})$. Puesto que $\bar{T}$ es continuo y X_{i-1} es el supremo de sus aproximaciones finitas, existe $Z \subset X$ finita tal que $y \in \bar{T}(Z)(\vec{x})$. Por hipótesis de inducción, cada tupla de grafo(Z) es producida por alguna T-derivación. Como grafo(Z) es finito, todas estas T-derivaciones pueden concatenarse para obtener una T-derivación que produzca $\langle \vec{x}, y\rangle$. Notar que la monotonía de T es necesaria para garantizar que esta concatenación sea una T-derivación.

$\underline{min(T) \subseteq fix(\bar{T})}$: Supongamos ahora que $\langle \vec{x}, y\rangle$ es producida por una T-derivación $\langle \vec{s}, s\rangle$ de longitud k+1, ($k \geq 0$). Por inducción sobre k, demostramos que $\langle \vec{x}, y\rangle \in grafo(fix(\bar{T}))$. De la definición de T-derivación, tenemos que $y \in T(fun_k(\langle \vec{s}, s\rangle))(\vec{x})$. Por construcción de $fun_k(\langle \vec{s}, s\rangle)$, cada tupla del grafo de esta función es producida por una T-derivación de longitud estrictamente menor que k (más precisamente, por un prefijo de $\langle \vec{s}, s\rangle$). Por hipotesis de inducción, podemos asegurar que cada tupla de $grafo(fun_k(\langle \vec{s}, s\rangle))$ pertenece a algún X_j. Tomando el máximo de estos j, obtenemos un i tal que $fun_k(\langle \vec{s}, s\rangle) \subset X_j$. Luego podemos concluir haciendo uso de que $\bar{T}$ es monótono que:

$$\langle \vec{x}, y\rangle \in grafo(T(fun_k(\langle \vec{s}, s\rangle))) = grafo(\bar{T}(fun_k(\langle \vec{s}, s\rangle))) \subseteq grafo(\bar{T}(X_i)) = grafo(X_{i+1}) \subseteq grafo(fix(\bar{T})).$$

(ii) Supongamos que $\tilde{T}(Z)\ll Z$ y que $\langle\vec{x},y\rangle\in$grafo(min(T)), es decir que $\langle\vec{x},y\rangle$ es producida por una T-derivación $\langle\vec{s},s\rangle$ de longitud k+1, ($k\geq 0$). Por inducción sobre k, se demuestra que $\langle\vec{x},y\rangle\in$grafo(Z) haciendo uso de la monotonía de $\tilde{T}$ y la propiedad S (iv).

(iii) Veamos las dos inclusiones:

$\tilde{T}(\min(T))\subseteq\min(T)$: como $fix(\tilde{T})=\min(T)$, $\tilde{T}(\min(T)=\min(T)$.

$\min(T)\subseteq T(\min(T))$: sea $\langle\vec{x},y\rangle\in$grafo(min(T)), entonces $\langle\vec{x},y\rangle$ es producida por una T-derivación $\langle\vec{s},s\rangle$ de longitud k+1, ($k\geq 0$). Por inducción sobre k, usando la monotonía de T, se demuestra que $\langle\vec{x},y\rangle\in$grafo(T(min(T))).

(iv) Supongamos que $T(Z)\ll Z$ y que $\langle\vec{x},y\rangle\in$grafo(min(T)), es decir, que $\langle\vec{x},y\rangle$ es producida por una T-derivación $\langle\vec{s},s\rangle$ de longitud k+1, ($k\geq 0$). Por inducción sobre k, se demuestra que $\langle\vec{x},y\rangle\in$grafo(Z) haciendo uso de la monotonía de T. ■

Como corolario de este teorema, se obtienen los siguientes resultados.

11- TEOREMA

Si $T:\mathfrak{N}\mathfrak{V}_D^{(n)}\rightarrow\mathfrak{N}\mathfrak{V}_D^{(n)}$ es monótono, entonces min(T) es:

(a) el menor punto fijo del operador $\tilde{T}$ asociado a T.
(b) el casi menor punto fijo de T.

Demostración

(a) es (i) del teorema 10.
(b) se deduce del teorema 10 (iii) y (iv). ■

12- TEOREMA

Si $T:\mathfrak{N}\mathfrak{I}_D^{(n)} \rightarrow \mathfrak{N}\mathfrak{I}_D^{(n)}$ es continuo, min(T) es el menor punto fijo de T.

Demostración

Se deduce del teorema 11 (a) y la propiedad 5 (iii). ■

13- EJEMPLO

La función indeterminista SP: $\mathbb{N} \rightarrow \mathcal{P}(\mathbb{N})$ definida recursivamente por las ecuaciones:

SP(0) = 2*N
SP(1) = 2*(N+1)
SP(n+2) = 2+SP(n)

que hace corresponder a cada número natural el conjunto de sus sucesores pares:

SP(n) = { m∈N / m≥n y par(m) }

es el menor punto fijo del operador continuo (asociado a las ecuaciones)

$T: \mathfrak{N}\mathfrak{I}_{\mathbb{N}}^{(1)} \rightarrow \mathfrak{N}\mathfrak{I}_{\mathbb{N}}^{(1)}$

definido por:

T(X)(0) = 2*N
T(X)(1) = 2*(N+1)
T(X)(n+2) = 2+X(n)

Obsérvese que todas las aproximaciones de T son infinitas aunque solo esten definidas para un número finito de elmentos.

En concreto:

$T(\perp)(0) = \mathbb{N}$
$T(\perp)(1) = \mathbb{N}\setminus\{0\}$
$T(\perp)(2) = \emptyset$
$T^i(\perp)(n) = SP(n)$ si $n \leq i$ para todo $i>0$
$T^i(\perp)(n) = \emptyset$ si $n>i$ para todo $i>0$.

Una T-derivación de longitud dos que produce el par $\langle 2,2\rangle \in grafo(min(T)) = grafo(SP)$ puede ser:

$\langle\langle 0,2\rangle,\langle 0,2\rangle\rangle = \langle s_1, s\rangle$ ya que verifica las condiciones:

$\langle 2,2\rangle = \langle s_1(1), s(1)\rangle$
$s(0)=0 \in T(\perp)(0)$ donde $fun_0 = \perp$ y $s_1(0)=0$
$s(1)=2 \in T(fun_1)(2)$ donde $grafo(fun_1)=\{\langle 0,0\rangle\}$ y $s_1(1)=2$

Evidentemente $\langle 2,2\rangle \in grafo(T^2(\perp))$, pero como las T-derivaciones son finitas, no podemos hacer uso directamente de las aproximaciones de T para obtener T-derivaciones, sino que tenemos que coger solamente la parte necesaria para producir el elemento en cuestión (que como se ha demostrado en el teorema anterior es finita). Así, en este ejemplo de T-derivación, la parte finita de $T(\perp)$ que se ha escogido es fun_1 y la parte finita de $T^2(\perp)$ es la propia T-derivación.

Además, como veremos en el siguiente capítulo, esta definición es expresable en nuestro lenguaje. ■

3.- SINTAXIS Y SEMANTICA DE LRF

Vamos ahora a definir nuestra lógica para programas funcionales a la que hemos llamado LRF por ser las iniciales de Logic for Recursive Functions.

De hecho, como ya hemos dicho, definiremos tres variantes de LRF, que serán restricciones sintácticas de la siguiente definición. Estas restricciones son necesarias para el estudio semántico del menor punto fijo.

SINTAXIS DE LRF

El lenguaje de LRF consta de expresiones funcionales, expresiones y fórmulas. Las expresiones funcionales y las expresiones hacen el papel de los programas funcionales mientras que las fórmulas posibilitan el razonamiento sobre el comportamiento de los programas.

1- DEFINICION *(Sintaxis)*

Sea Σ una signatura fija. Las Σ-expresiones funcionales de aridad n $(F, G \in FEXP_{\Sigma}^{(n)})$ Σ-expresiones $(M, N \in EXP_{\Sigma})$ y Σ-fórmulas $(\varphi, \psi \in FFOR_{\Sigma})$ de LRF se definen mediante las siguientes reglas BNF:

(a) Expresiones funcionales de aridad $n \geq 0$

$F^{(n)} ::=$ f (símbolos de función u operaciones primitivas) |
X (variables funcionales o procedimientos globales) |
$\lambda x_1 \ldots x_n . M$ (λ-abstracciones) |
$\mu X . G$ (definiciones recursivas o μ-operador)

(b) Expresiones

$M::= t$ (términos de primer orden) |
$F(N_1,\ldots,N_n)$ (aplicación) |
$(N_1 \cup N_2)$ (elección no determinista) |
$(\varphi \rightarrow N)$ (expresión condicional) |
$\varepsilon x\varphi$ (ε-abstracción)

(c) Fórmulas

$\varphi::= (t_1 = t_2)$ (ecuación) |
$p(M_1,\ldots,M_k)$ (predicados) |
$M \ni t$ (pertenencia) |
$\neg\psi$ (negación) |
$(\psi_1 \vee \psi_2)$ (disyunción) |
$\exists x\psi$ (cuantificación existencial)

donde $t, t_1, t_2 \in TER_\Sigma$ y n, m y k son las aridades asociadas a X, F y P respectivamente. ■

Tanto en las expresiones funcionales como en las expresiones y fórmulas de LRF, las variables de función no siempre aparecen libres.

2- DEFINICION *(Variables Libres)*

Dada una signatura Σ, el conjunto de variables libres de una expresión funcional, expresión y fórmula se define recursivamente por:

(a) $Lib(F) \subseteq V$, para cualquier $F \in FEXP_\Sigma$:

$Lib(f) = \emptyset$
$Lib(X) = \{X\}$
$Lib(\lambda x_1 \ldots x_n.M) = Lib(M) \setminus \{x_1,\ldots,x_n\}$
$Lib(\mu X.G) = Lib(G) \setminus \{X\}$

(b) $Lib(M) \subseteq V$, para cualquier $M \in EXP_\Sigma$:

$Lib(t) = var(t)$ se define como en la lógica de primer orden clásica
$Lib(F(N_1,\ldots,N_n)) = Lib(F) \cup \{Lib(N_i) \;/\; 1 \leq i \leq n\}$
$Lib(N_1 \cup N_2) = Lib(N_1) \cup Lib(N_2)$
$Lib(\varphi \rightarrow N) = Lib(\varphi) \cup Lib(N)$
$Lib(cx\varphi) = Lib(\varphi) \setminus \{x\}$

(c) $Lib(\varphi) \subseteq V$, para cualquier $\varphi \in FFOR_\Sigma$:

$Lib(t_1 = t_2) = Lib(t_1) \cup Lib(t_2)$
$Lib(p(M_1,\ldots,M_n)) = \{Lib(M_i) \;/\; 1 \leq i \leq n\}$
$Lib(M \ni t) = Lib(M) \cup Lib(t)$
$Lib(\neg\psi) = Lib(\psi)$
$Lib(\psi_1 \vee \psi_2) = Lib(\psi_1) \cup Lib(\psi_2)$
$Lib(\exists x\psi) = Lib(\psi) \setminus \{x\}$ ■

Además, denotamos al conjunto de variables que aparecen en Θ por $var(\Theta)$, para Θ fórmula, expresión o expresión funcional.

La siguientes dos definiciones se usan a continuación para definir las variantes de LRF.

3- DEFINICION *(Expresión Simple)*

$M \in EXP_\Sigma$ es una expresión simple, si toda fórmula φ que aparece en una subexpresión $\varphi \rightarrow N$ o $cx\varphi$ de M, es de primer orden extendido. ■

4- DEFINICION *(Función positiva, existencial)*

(a) $F \in FEXP_\Sigma^{(n)}$ (resp. $M \in EXP_\Sigma$, $\varphi \in FFOR_\Sigma$) es positiva (resp. negativa) en la variable funcional X sii cualquier aparición libre de X en F (resp. M, φ) esta dentro del alcance de un número par (resp. impar) de símbolos de negación.

(b) $F \in FEXP_{\Sigma}^{(n)}$ (resp. $M \in EXP_{\Sigma}$, $\varphi \in FFOR_{\Sigma}$) es existencial (resp. universal) en la variable funcional X sii F (resp. M, φ) es positiva (resp. negative) en X y cualquier aparición de una fórmula $\exists x\psi$ en F (resp. M, φ) que este dentro del alcance de un número impar (resp. par) de símbolos de negación verifica que $X \notin Lib(\psi)$. ■

5- LEMA

Las expresiones funcionales F^{eX} y F^{uX}, expresiones M^{eX} y M^{uX} y fórmulas φ^{eX} y φ^{uX} definidas por las siguientes reglas BNF son exactamente las expresiones funcionales existenciales y universales en X, las expresiones existenciales y universales en X y las fórmulas existenciales y universales en la variable funcional X respectivamente.

$$F^{eX} ::= f \mid X \mid Y \mid \lambda x_1 \ldots x_n.M^{eX} \mid \mu Y.G^{eX} \mid \mu X.G^{eX}$$

$$M^{eX} ::= t \mid F^{eX}(N_1^{eX},\ldots,N_n^{eX}) \mid (N_1^{eX} \cup N_2^{eX}) \mid (\varphi^{eX} \to N^{eX}) \mid cx\varphi^{eX}$$

$$\varphi^{eX} ::= (t_1 = t_2) \mid p(M_1^{eX},\ldots,M_n^{eX}) \mid M^{eX} \ni t \mid \neg\psi^{uX} \mid (\psi_1^{eX} \vee \psi_2^{eX}) \mid \exists x\psi^{eX}$$

$$F^{uX} ::= f \mid Y\ (Y \neq X) \mid \lambda x_1 \ldots x_n.M^{uX} \mid \mu Y.G^{uX} \mid \mu X.G^{uX}$$

$$M^{uX} ::= t \mid F^{lX}(N_1^{lX},\ldots,N_n^{lX}) \mid (N_1^{uX} \cup N_2^{uX}) \mid (\varphi^{uX} \to N^{uX}) \mid cx\varphi^{uX}$$

$$\varphi^{uX} ::= (t_1 = t_2) \mid p(M_1^{lX},\ldots,M_n^{lX}) \mid M^{lX} \ni t \mid \neg\psi^{eX} \mid (\psi_1^{uX} \vee \psi_2^{uX}) \mid \exists x\psi^{lX}$$

donde F^{lX}, M^{lX} y φ^{lX} denota que X es una variable ligada; es decir $X \notin lib(F)$, $X \notin lib(M)$ y $X \notin lib(\varphi)$ y $X \neq Y$. ■

El significado estándar de una definición recursiva $\mu X.F$ es el menor punto fijo de la ecuación X=F(X). Como veremos en el lema 15, este menor punto fijo existe siempre que F sea positiva en X. Además este puede ser construido como el supremo de una cadena de aproximaciones, siempre que F sea existencial en X. Por esta razón, restringimos la especificación sintáctica de LRF (cfr. Definición 1) de la siguiente forma:

6- DEFINICION

(a) LRF_M (LRF *Monótono*) es la restricción sintáctica de LRF obtenida al exigir que una expresión funcional $\mu X.G$ solo se pueda formar en el caso de que G sea positiva en X.

(b) LRFc (LRF *Continuo*) es la restricción sintáctica de LRF obtenida al exigir que una expresión funcional $\mu X.G$ solo se pueda formar en el caso de que G sea existencial en X.

(c) LRFs (LRF *Simple*) es la restricción sintáctica de LRFc obtenida al exigir que todas las expresiones de LRFc sean simples. ■

Obsérvese que $LRFs \subseteq LRFc \subseteq LRF_M$, y que por lo tanto todos los resultados que se obtengan para LRF_M lo son también para LRFc y LRFs, y análogamente para LRFc y LRFs.

SEMANTICA DENOTACIONAL LRF

Como ya hemos dicho, el significado estándar de una definición recursiva $\mu X.F$ es el menor punto fijo de la ecuación $X=F(X)$. Por otro lado, el significado de una expresión M es un subconjunto del dominio de la estructura, que representa los posibles valores de M con una evaluación indeterminista.

7- DEFINICION (*Semántica Denotacional*)

Sea $\mathfrak{D}$ una Σ-estructura, δ una valoración y Δ un contexto. El valor de las expresiones funcionales, expresiones y fórmulas de LRF_M queda definido por:

(a) $F^{\mathcal{D}}(\Delta,\delta)\in\mathfrak{N}\mathfrak{I}_{\mathcal{D}}^{(n)}$, para toda $F\in FEXP_{\Sigma}^{(n)}$, donde

$f^{\mathcal{D}}(\Delta,\delta) = f^{\mathcal{D}}$

$X^{\mathcal{D}}(\Delta,\delta) = \Delta(X)$

$((\lambda x_1\ldots x_n.M)^{\mathcal{D}}(\Delta,\delta))(x_1,\ldots,x_n) = M^{\mathcal{D}}(\Delta,\delta[x_1/x_1,\ldots,x_n/x_n])$.

para todo $x_1,\ldots,x_n\in D$.

$(\mu X.G)^{\mathcal{D}}(\Delta,\delta) = fix(T(\mathcal{D},\Delta,\delta,X,G)) = fix(T)$ es la menor $X\in\mathfrak{N}\mathfrak{I}_{\mathcal{D}}^{(n)}$ tal que $X = G^{\mathcal{D}}(\Delta[X/X],\delta)$ es decir, el menor punto fijo del operador $T=T(\mathcal{D},\Delta,\delta,X,G):\mathfrak{N}\mathfrak{I}_{\mathcal{D}}^{(n)}\longrightarrow\mathfrak{N}\mathfrak{I}_{\mathcal{D}}^{(n)}$ definido por $T(X) = G^{\mathcal{D}}(\Delta[X/X],\delta)$. (cfr. definición 6 (b) de la sección 2)

(b) $M^{\mathcal{D}}(\Delta,\delta)\subseteq D$, para toda $M\in EXP_{\Sigma}$, donde

$t^{\mathcal{D}}(\Delta,\delta) = \{t^{\mathcal{D}}(\delta)\}$

$F(N_1,\ldots,N_n)^{\mathcal{D}}(\Delta,\delta) = \{ F^{\mathcal{D}}(\Delta,\delta[x_1/x_1,\ldots,x_n/x_n]) \;/\; x_i\in N_i^{\mathcal{D}}(\Delta,\delta), 1\le i\le n \}$

$(N_1\cup N_2)^{\mathcal{D}}(\Delta,\delta) = N_1^{\mathcal{D}}(\Delta,\delta)\cup N_2^{\mathcal{D}}(\Delta,\delta)$

$(\varphi\to N)^{\mathcal{D}}(\Delta,\delta) = \begin{cases} N^{\mathcal{D}}(\Delta,\delta) & \text{si } \varphi^{\mathcal{D}}(\Delta,\delta) = true \\ \varnothing & \text{e.o.c.} \end{cases}$

$(\varepsilon x\varphi)^{\mathcal{D}}(\Delta,\delta) = \{ x\in D \;/\; \varphi^{\mathcal{D}}(\Delta,\delta[x/x]) = true \}$

(c) $\varphi^{\mathcal{D}}(\Delta,\delta)\in\{true, false\}$, para toda $\varphi\in FFOR_{\Sigma}$, donde

$(t_1=t_2)^{\mathcal{D}}(\Delta,\delta) = true$ sii $t_1^{\mathcal{D}}(\delta)=t_2^{\mathcal{D}}(\delta)$

$p(M_1,\ldots,M_n)^{\mathcal{D}}(\Delta,\delta)=true$ sii existen $x_i\in M_i^{\mathcal{D}}(\Delta,\delta)$, $1\le i\le n$ tal que $p^{\mathcal{D}}(x_1,\ldots,x_n)=true$

$(M\ni t)^{\mathcal{D}}(\Delta,\delta) = true$ sii $t^{\mathcal{D}}(\delta)\in M^{\mathcal{D}}(\Delta,\delta)$

$(\neg\psi)^{\mathcal{D}}(\Delta,\delta)$, $(\psi_1\vee\psi_2)^{\mathcal{D}}(\Delta,\delta)$ y $(\exists x\psi)^{\mathcal{D}}(\Delta,\delta)$ se definen como en lógica de primer orden. ■

Satisfactibilidad, validez y consecuencia lógica se definen como en la definición 4 de la sección 1, para la lógica de primer orden extendida.

Como veremos, el operador T de la definición 7 (a) asociado a una expresión funcional $\mu X.G$, será monótono si G es positiva en X y continuo si G es existencial en X. Por lo que la semántica estará bien definida para LRF$_M$.

8- NOTACION

De acuerdo con la semántica que acabamos de definir, tenemos: $(\exists x\ x{=}x)^{\mathcal{D}}(\Delta,\delta)$=true, $(\exists x\ \neg\ x{=}x)^{\mathcal{D}}(\Delta,\delta)$=false para toda $\mathcal{D},\Delta,\delta$. Denotaremos estas fórmulas por tt y ff respectivamente. Además denotaremos por $\Omega^{(n)}$ a la expresión funcional $\lambda x_1 \ldots x_n.(ff \rightarrow x_1)$, cuyo valor es la función n-aria totalmente indefinida para toda $\mathcal{D},\Delta,\delta$. ■

9- DEFINICION *(Símbolos Definidos)*

Los símbolos < >, [ ] y $\downarrow$ son usados como sigue:

<M>yφ suple a $\exists z(M \ni z \wedge \varphi[z/y])$ (donde z es diferente de y y no aparece ni en M ni en φ).

[M]yφ suple a $\neg\langle M\rangle y \neg\varphi$.

M$\downarrow$N suple a $\exists z(M \ni z \wedge N \ni z)$ (donde z no aparece ni en M ni en N).

< > y [ ] juegan el mismo papel que en la lógica dinámica [Har 84], mientras que $\downarrow$ actua como un tipo de "igualdad existencial" entre expresiones indeterministas. ■

10- EJEMPLO

Las siguientes expresiones funcionales son ejemplos de como se pueden expresar funciones recursivas y mutuamente recursivas en nuestro formalismo.

(a) La función factorial Fact:$\mathbb{N} \rightarrow \mathbb{N}$ definida por

$$\text{Fact}(n) = \begin{cases} 1 & \text{si } n{=}0 \\ n{*}\text{Fact}(n{-}1) & \text{si } n{>}0 \end{cases}$$

puede expresarse por la expresión funcional

$$\text{Fact} = \mu X.\ \lambda x.(x{=}0 \rightarrow 1\ \cup\ \neg\ x{=}0 \rightarrow x{*}X(x{-}1))$$

(b) Las funciones caracteristicas de los conjuntos pares e impares admiten la siguiente definición mutuamente recursiva

$$PAR(n) = \begin{cases} 1 & \text{si } n=0 \\ IMPAR(n-1) & \text{si } n>0 \end{cases}$$

$$IMPAR = \begin{cases} 0 & \text{si } n=0 \\ PAR(n-1) & \text{si } n>0 \end{cases}$$

podemos expresar, por ejemplo, la función PAR mediante una expresión funcional anidando el operador μ

$$PAR = \mu X.\ \lambda x.(x=0 \longrightarrow 1 \ \cup \ \neg\, x=0 \longrightarrow (\mu Y.\ \lambda y.(y=0 \longrightarrow 0 \ \cup \ \neg\, y=0 \longrightarrow X(y-1)))(x-1))$$

(c) La función no determinista de los sucesores pares, usada en el ejemplo 13 de la sección 2, puede expresarse mediante la expresión funcional:

$$SP = \mu X.\ \lambda x.(x=0 \longrightarrow 2*UNI \ \cup \ x=1 \longrightarrow 2*(UNI+1) \ \cup \ x>1 \longrightarrow 2+ SP(x-2))$$

donde UNI es la expresión definida por: $UNI = \varepsilon z\ (z=z)$. ■

Ahora vamos a extender el concepto de sustitución permitiendo la sustitución de una variable funcional por una expresión funcional de la misma aridad.

11- DEFINICION *(Sustitución)*

(a) La sustitución de una variable libre de individuo x por un término t en una expresión funcional F, expresión M o fórmula φ (denotada pnr F[t/x], M[t/x] y φ[t/x] respectivamente) se define recursivamente como sigue:

(a.1) $F[t/x]$

$f[t/x] = f$

$X[t/x] = X$

$$(\lambda x_1 \ldots x_n . M)[t/x] = \begin{cases} \lambda x_1 \ldots x_n . M[t/x] & \text{si } (x \neq x_1 \wedge x_1 \notin lib(t)) \ (1 \leq i \leq n) \\ (\lambda y_1 \ldots y_n . M[\vec{y}/\vec{x}])[t/x] & \text{si } x \neq x_i \ (1 \leq i \leq n) \ \wedge \text{ ex. } j, \\ & x_j \in lib(t) \text{ con } y_1 \notin lib(t) \cup lib(F) \end{cases}$$

$(\mu X.G)[t/x] = \mu X.G[t/x]$

(a.2) $M[t/x]$

$t_1[t/x]$ se define como en lógica de primer orden.

$(F(N_1,\ldots,N_n))[t/x] = F[t/x](N_1[t/x],\ldots,N_n[t/x])$

$(N_1 \cup N_2)[t/x] = N_1[t/x] \cup N_2[t/x]$

$(\varphi \rightarrow N)[t/x] = \varphi[t/x] \rightarrow N[t/x]$

$$(cy\varphi)[t/x] = \begin{cases} cy\varphi[t/x] & \text{si } (x \neq y \wedge y \notin lib(t)) \\ cz\varphi[z/y, t/x] & \text{si } (x \neq y \wedge y \in lib(t)) \text{ con } z \notin lib(t) \cup lib(\varphi) \end{cases}$$

(a.3) $\varphi[t/x]$

$(t_1 = t_2)[t/x] = (t_1[t/x] = t_2[t/x])$

$(p(M_1,\ldots,M_n))[t/x] = p(M_1[t/x],\ldots,M_n[t/x])$

$(M \ni t_1)[t/x] = M[t/x] \ni t_1[t/x]$

$(\neg\psi)[t/x] = \neg\psi[t/x]$

$(\psi_1 \vee \psi_2)[t/x] = \psi_1[t/x] \vee \psi_2[t/x]$

$$(\exists y\psi)[t/x] = \begin{cases} \exists y\psi[t/x] & \text{si } (x \neq y \wedge y \notin lib(t)) \\ \exists z\psi[z/y, t/x] & \text{si } (x \neq y \wedge y \in lib(t)) \text{ con } z \notin lib(t) \cup lib(\psi) \end{cases}$$

(b) La sustitución de una variable libre de función X por una expresión funcional G de la misma aridad, en una expresión funcional F, expresión M o fórmula φ (denotada por $F[G/X]$, $M[G/X]$ y $\varphi[G/X]$ respectivamente) se define recursivamente como sigue:

(b.1) $F[G/X]$

$f[G/X] = f$

$$Y[G/X] = \begin{cases} Y & \text{si } X \neq Y \\ G & \text{si } X = Y \end{cases}$$

$(\lambda x_1 \ldots x_n . M)[G/X] = \lambda x_1 \ldots x_n . (M[G/X])$

$$(\mu Y . G_1)[G/X] = \begin{cases} \mu Y . G_1[G/X] & \text{si } (X \neq Y \wedge Y \notin lib(G)) \\ \mu Z . G_1[Z/Y, G/X] & \text{si } (X \neq Y \wedge Y \in lib(G)) \text{ con } Z \notin lib(G) \cup lib(G_1) \end{cases}$$

(b.2) $M[G/X]$

$t[G/X] = t$

$(F(N_1, \ldots, N_n))[G/X] = F[G/X](N_1[G/X], \ldots, N_n[G/X])$

$(N_1 \cup N_2)[G/X] = N_1[G/X] \cup N_2[G/X]$

$(\varphi \rightarrow N)[G/X] = \varphi[G/X] \rightarrow N[G/X]$

$(\varepsilon x \varphi)[G/X] = \varepsilon x (\varphi[G/X])$

(b.3) $\varphi[G/X]$

$(t_1 = t_2)[G/X] = (t_1 = t_2)$

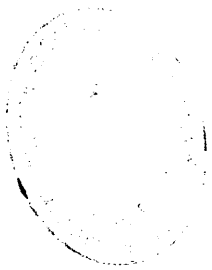
$(p(M_1, \ldots, M_n))[G/X] = p(M_1[G/X], \ldots, M_n[G/X])$

$(M \ni t)[G/X] = M[G/X] \ni t$

$(\neg \psi)[G/X] = \neg \psi[G/X]$

$(\psi_1 \vee \psi_2)[G/X] = \psi_1[G/X] \vee \psi_2[G/X]$

$(\exists x \psi)[G/X] = \exists x \psi[G/X]$



(c) La sustitución de una variable ligada de individuo x (resp. de función X) por un término (o por una expresión funcional respectivamente) en Θ (expresión funcional, expresión o fórmula) denotada como an (a) (resp. en (b)) se define por: $\Theta[t/x] = \Theta$ y $\Theta[G/X] = \Theta$ repectivamente. ■

El siguiente lema técnico será usado más adelante.

12- LEMA

Sea Θ cualquier expresión funcional, expresión o fórmula de LRF. Si Θ es positiva (resp. negativa, existencial, universal) y F es una expresión funcional de LRF positiva (resp. negatiava, existencial, universal) en una variable de función dada Y, entonces $\Theta[F/X]$ tiene la misma propiedad.

Demostración

Por inducción simultánea sobre las expresiones funcionales, expresiones y fórmulas de LRF. ■

13- LEMA *(Coincidencia)*

Sean $\mathfrak{D}_1$ y $\mathfrak{D}_2$ dos Σ-estructuras con el mismo dominio, δ_1 y δ_2 dos valoraciones, Δ_1 y Δ_2 dos contextos y Θ una Σ-expresión funcional, Σ-expresión o Σ-fórmula.

Si $\mathfrak{D}_1$ y $\mathfrak{D}_2$ coinciden en la interpretación de los símbolos que aparecen en Θ y δ_1, Δ_1 y δ_2, Δ_2 coinciden en la valoración de las variables libres de Θ entonces $\Theta^{\mathfrak{D}_1}(\Delta_1, \delta_1) = \Theta^{\mathfrak{D}_2}(\Delta_2, \delta_2)$.

Demostración

Intuitivamente el Lema afirma que el significado de las expresiones funcionales, expresiones y fórmulas solo depende de la interpretación de los símbolos que aparecen en ellas y de sus variables libres. Se demuestra por inducción simultánea sobre las expresiones funcionales, expresiones y fórmulas, de forma análoga a como en [EFT 84] se demuestra este Lema para la lógica de primer orden clásica. ■

14-LEMA *(Sustitución)*

Para toda Θ (expresión funcional, expresión o fórmula), contexto Δ y valoración δ:

(a) $(\Theta[t/x])^{\mathfrak{D}}(\Delta, \delta) = \Theta^{\mathfrak{D}}(\Delta, \delta[t^{\mathfrak{D}}(\delta)/x])$ p.t. $x \in VAR$ y $t \in TER$.

(b) $(\Theta[F/X])^{\mathfrak{D}}(\Delta, \delta) = \Theta^{\mathfrak{D}}(\Delta[F^{\mathfrak{D}}(\Delta, \delta)/\Delta], \delta)$ p.t. $X \in FVAR$ y $F \in FEXP$.

Demostración

Por inducción simultánea sobre las expresiones funcionales, expresiones y fórmulas. Veamos los casos mas significativos:

Para (a) con $\Theta=\lambda\vec{x}.M$

Supongamos que $x\neq x_i$ y que $x_i\notin lib(t)$ para toda $1\leq i\leq n$.
Entonces $\Theta[t/x] = \lambda\vec{x}.M[t/x]$ y por tanto, para todo $\vec{x}\in D^n$:
$\Theta[t/x]^{\mathcal{D}}(\Delta,\delta)(\vec{x}) = M[t/x]^{\mathcal{D}}(\Delta,\delta[\vec{x}/\vec{x}])$ por hipótesis de inducción
$= M^{\mathcal{D}}(\Delta,\delta[\vec{x}/\vec{x},\ t^{\mathcal{D}}(\delta[\vec{x}/\vec{x}])/x])$ por el lema de coincidencia
$= M^{\mathcal{D}}(\Delta,\delta[\vec{x}/\vec{x},t^{\mathcal{D}}(\delta)/x]) = \Theta^{\mathcal{D}}(\Delta,\delta[t^{\mathcal{D}}(\delta)/x])(\vec{x})$ como se quería demostrar.

Supongamos que $x\neq x_i$ para todo $1\leq i\leq n$ y que para algún $1\leq j\leq n$ $x_j\in lib(t)$. Entonces $\Theta[t/x] = \lambda\vec{y}.M[\vec{y}/\vec{x},t/x]$ donde $\vec{y}$ son variables nuevas, y por lo tanto, para todo $\vec{x}\in D^n$:
$\Theta[t/x]^{\mathcal{D}}(\Delta,\delta)(\vec{x}) = M[\vec{y}/\vec{x},t/x]^{\mathcal{D}}(\Delta,\delta[\vec{x}/\vec{y}])$ por hipótesis de inducción
$= M^{\mathcal{D}}(\Delta,\delta[\vec{x}/\vec{y},\ t^{\mathcal{D}}(\delta[\vec{x}/\vec{y}])/x,\ \vec{x}/\vec{x}])$ por el lema de coincidencia
$= M^{\mathcal{D}}(\Delta,\delta[t^{\mathcal{D}}(\delta)/x,\ \vec{x}/\vec{x}]) = \Theta^{\mathcal{D}}(\Delta,\delta[t^{\mathcal{D}}(\delta)/x])(\vec{x})$ como se quería demostrar.

Para (b) con $\Theta=\mu Y.G$:

Supongamos que $X\neq Y$ y que $Y\notin lib(G)$. Entonces $\Theta[F/X] = \mu Y.G[F/X]$ y $\Theta[F/X]^{\mathcal{D}}(\Delta,\delta) = fix(T')$, donde para toda $Y\in\mathfrak{M}^{(n)}$:
$T'(Y) = G[F/X]^{\mathcal{D}}(\Delta[Y/Y],\delta)$ por hipótesis de inducción
$= G^{\mathcal{D}}(\Delta[Y/Y,\ F^{\mathcal{D}}(\Delta[y/Y],\delta)/X],\delta)$ por el lema de coincidencia
$= G^{\mathcal{D}}(\Delta[Y/Y,\ F^{\mathcal{D}}(\Delta,\delta)/X],\delta) = T(Y)$ y por tanto
$fix(T') = fix(T) = \Theta^{\mathcal{D}}(\Delta[F^{\mathcal{D}}(\Delta,\delta)/X],\delta)$ como se quería demostrar.

Supongamos que $X\neq Y$ y que $Y\notin lib(F)$. Entonces $\Theta[F/X] = \mu Z.G[Z/Y,\ F/X]$ donde Z es una variable nueva y $\Theta[F/X]^{\mathcal{D}}(\Delta,\delta) = fix(T')$, donde para toda $Y\in\mathfrak{M}^{(n)}$:
$T'(Y) = G[Z/Y,\ F/X]^{\mathcal{D}}(\Delta[Y/Z],\delta)$ por hipótesis de inducción
$= G^{\mathcal{D}}(\Delta[Y/Z,\ F^{\mathcal{D}}(\Delta[Y/Z],\delta)/X,\ Y/Y),\delta)$ por el lema de coincidencia
$= G^{\mathcal{D}}(\Delta[F^{\mathcal{D}}(\Delta,\delta)/X,\ Y/Y) = T(Y)$ y por tanto
$fix(T') = fix(T) = \Theta^{\mathcal{D}}(\Delta[F^{\mathcal{D}}(\Delta,\delta)/X],\delta)$ como se quería demostrar. ■

Ahora podemos demostrar la presupuesta existencia del menor punto fijo del operador T de la definición 7 (a). Para esto usaremos los resultados de la sección 2, y a continuación, vamos a comprobar la monotonía y continuidad de estos operadores.

15- LEMA *(Monotonía)*

Consideremos $X \in FVAR$, $F \in FEXP_{\Sigma}^{(n)}$, $M \in EXP_{\Sigma}$, $\varphi \in FFOR_{\Sigma}$, una Σ-estructura $\mathcal{D}$, una valoración δ sobre $\mathcal{D}$, un contexto Δ sobre $\mathcal{D}$ y dos funciones $X_1, X_2 \in \mathfrak{N}\mathfrak{V}_{\mathcal{D}}^{(n)}$ con $X_1 \ll X_2$. Entonces :

(a.1) Si F es positiva en X: $F^{\mathcal{D}}(\Delta[X_1/X],\delta) \ll F^{\mathcal{D}}(\Delta[X_2/X],\delta)$.

(a.2) Si M es positiva en X: $M^{\mathcal{D}}(\Delta[X_1/X],\delta) \subseteq M^{\mathcal{D}}(\Delta[X_2/X],\delta)$.

(a.3) Si φ es positiva en X: $\varphi^{\mathcal{D}}(\Delta[X_1/X],\delta)$=true implica $\varphi^{\mathcal{D}}(\Delta[X_2/X],\delta)$=true.

(b.1) Si F es negativa en X: $F^{\mathcal{D}}(\Delta[X_2/X],\delta) \ll F^{\mathcal{D}}(\Delta[X_1/X],\delta)$.

(b.2) Si M es negativa en X: $M^{\mathcal{D}}(\Delta[X_2/X],\delta) \subseteq M^{\mathcal{D}}(\Delta[X_1/X],\delta)$.

(b.3) Si φ es negativa en X: $\varphi^{\mathcal{D}}(\Delta[X_2/X],\delta)$=true implica $\varphi^{\mathcal{D}}(\Delta[X_1/X],\delta)$=true.

Demostración

Por inducción simultánea sobre la estructura de F, M y φ. Veamos los casos mas relevantes:

Caso para (a.1):
Supongamos que $F=(\mu Y.G)$ es positive en X y que X,Y son variables distintas (en otro caso, la situación es trivial). Por hipótesis de inducción para (a.1), sabemos que

(i) $G^{\mathcal{D}}(\Delta[X_1/X,Y/Y],\delta) \ll G^{\mathcal{D}}(\Delta[X_2/X,Y/Y],\delta)$ para toda Y.

(ii) T_j definido como $T_j(Y) = G^{\mathcal{D}}(\Delta[X_j/X,Y/Y],\delta)$ es monótono para j=1,2.

Luego podemos concluir que $F^{\mathcal{D}}(\Delta[X_j/X],\delta) = fix(T_j)$ existe para j=1,2, y que $F^{\mathcal{D}}(\Delta[X_1/X],\delta) \ll F^{\mathcal{D}}(\Delta[X_2/X],\delta)$.

Caso para (a.3):
Supongamos que $\varphi=(\neg\psi)$ es positiva en X y $\varphi^{\mathcal{D}}(\Delta[X_1/X],\delta)$=true. Entonces, ψ es negativa en X y $\psi^{\mathcal{D}}(\Delta[X_1/X],\delta)$=false. Y por hipótesis de inducción para (b.3) $\psi^{\mathcal{D}}(\Delta[X_2/X],\delta)$=false. Luego, $\varphi^{\mathcal{D}}(\Delta[X_2/X],\delta)$=true. ■

16- LEMA *(Continuidad)*

Sea $X \in FVAR$, $F \in FEXP_{\Sigma}^{(n)}$, $M \in EXP_{\Sigma}$, $\varphi \in FFOR_{\Sigma}$, $\mathcal{D}$ una Σ-estructura, δ una valoración, Δ un contexto y $X \in \mathfrak{N}\mathfrak{V}_{\mathcal{D}}^{(n)}$:

(a.1) Si F es existencial en X: $F^{\mathcal{D}}(\Delta,\delta) = \bigsqcup \{F^{\mathcal{D}}(\Delta[X_0/X],\delta) \,/\, X_0 \ll X,\ X_0 \text{ finita}\}$

(a.2) Si M es existencial en X: $M^{\mathcal{D}}(\Delta,\delta) = \cup\{M^{\mathcal{D}}(\Delta[X_0/X],\delta) \,/\, X_0 \ll X,\ X_0 \text{ finita}\}$

(a.3) Si φ es existencial en X: $\varphi^{\mathcal{D}}(\Delta,\delta)$=true sii $\varphi^{\mathcal{D}}(\Delta[X_0/X],\delta)$=true para alguna $X_0 \ll X$ finita.

(b.1) Si F es universal en X: $F^{\mathcal{D}}(\Delta,\delta) = \sqcap \{F^{\mathcal{D}}(\Delta[X_0/X],\delta) \,/\, X_0 \ll X,\ X_0 \text{ finita}\}$

(b.2) Si M es universal en X: $M^{\mathcal{D}}(\Delta,\delta) = \cap\{M^{\mathcal{D}}(\Delta[X_0/X],\delta) \,/\, X_0 \ll X,\ X_0 \text{ finita}\}$

(b.3) Si φ es universal en X: $\varphi^{\mathcal{D}}(\Delta,\delta)$=true sii $\varphi^{\mathcal{D}}(\Delta[X_0/X],\delta)$=true para toda $X_0 \ll X$ finita.

Demostración

Por inducción simultánea sobre la estructura de F, M y φ. Veamos los casos mas relevantes:

Caso para (b.3):
Supongamos que $\varphi = (\psi_1 \vee \psi_2)$ es universal en X. Entonces tenemos $\varphi^{\mathcal{D}}(\Delta[X/X],\delta)$=false sii $\psi_1^{\mathcal{D}}(\Delta,\delta)$=false y $\psi_2^{\mathcal{D}}(\Delta,\delta)$=false sii existen $X_1, X_2 \ll X$ tal que $\psi_1^{\mathcal{D}}(\Delta[X_1/X],\delta)$=false y $\psi_2^{\mathcal{D}}(\Delta[X_2/X],\delta)$=false (por hipótesis de inducción para (b.3)) sii existe $X_0 \ll X$ tal que $\psi_1^{\mathcal{D}}(\Delta[X_0/X],\delta)$=false y $\psi_2^{\mathcal{D}}(\Delta[X_0/X],\delta)$=false (tomando $X_0 = X_1 \sqcup X_2$ y aplicando el lema 15 (b.3)). ■

Por tanto, si F es positiva en X el operador T asociado a F y X tiene menor punto fijo. Y si Además F es existencial en X este menor punto fijo de T puede construirse, segun se vió en lema 4 y el teorema 5 de la sección 2. Luego la semántica de LRF_M (y por tanto la de LRFc y LRFs) está bien definida.

17- EJEMPLO

Con este ejemplo se quiere mostrar que la condición de positividad en una expresión funcional de LRF no es suficiente para garantizar la continuidad del operador asociado.

Sea $G = \lambda x.\ \varepsilon y\ ((\forall z\ X(x) \ni z) \wedge z=y)$

el operador asociado T(X,G) aplicado a una función X es la función:

$$T(X,G)(X)(x) = \begin{cases} \{z\} & \text{si } X(x) = D \\ \emptyset & \text{e.o.c.} \end{cases}$$

G es positiva pero no existencial en X y como puede observarse, T es monótono pero no es continuo en X (recuerdese la definición 3 de la sección 2) ya que $\sqcup\{T(X_0)\ /\ X_0 \subset X \text{ finita}\}$ es $\perp$ para toda X, y T(X) no es la función indefinida para toda X. ∎

APROXIMACIONES SINTACTICAS

Como ya hemos dicho, para LRFc, el menor punto fijo del operador T asociado a $\mu X.G$ se caracteriza como el supremo de la cadena creciente de sus aproximaciones:

$$fix(T) = \bigsqcup_{i \in N} T^i(\perp)\ ,\ \text{donde } T^0 = \text{identidad y } T^{i+1} = T(T^i)$$

Además, para LRFs se verifica que los elementos de esta cadena se pueden denotar mediante ciertas expresiones funcionales, llamadas aproximaciones sintácticas de $\mu X.G$. Estas aproximaciones serán usadas en la obtención de un cálculo infinitario para LRFs en el siguiente capítulo.

18- DEFINICION *(Aproximaciones Sintácticas)*

Para $F \in FEXP_\Sigma^{(n)}$, $M \in EXP_\Sigma$ de LRFs, la i-ésima aproximación sintáctica $F^{(i)}$ y $M^{(i)}$ de F y M respectivamente se define recursivamente como sigue:

(a) $F^{(i)}$

$f^{(i)} = f$

$X^{(i)} = X$

$(\lambda\vec{x}.M)^{(i)} = \lambda\vec{x}.M^{(i)}$

$(\mu X.G)^{(i)}$ se define por inducción sobre i:

$$(\mu X.G)^{(0)} = \Omega^{(n)}$$

$$(\mu X.G)^{(i+1)} = G^{(i+1)}[(\mu X.G)^{(i)}/X]$$

(b) $M^{(i)}$

$t^{(i)} = t$

$(N_1 \cup N_2)^{(i)} = N_1^{(i)} \cup N^{(i)}$

$(\varphi \to N)^{(i)} = \varphi \to N^{(i)}$

$F(N_1,\ldots,N_n)^{(i)} = F^{(i)}(N_1^{(i)},\ldots,N_n^{(i)})$

$(c x \varphi)^{(i)} = c x \varphi$ ■

Obsérvese que según esta definición, en las aproximaciones sintácticas no aparece el operador μ, ya que al $(\mu X.G)$ primero se aproxima G. Además si en F (resp. M) no aparece el operador μ, entonces $F^{(i)}=F$ (resp. $M^{(i)}=M$).

Alternativamente se podría haber definido $(\mu X.G)^{[i]}$ por:

$$(\mu X.G)^{[0]} = \Omega^{(n)} \text{ y } (\mu X.G)^{[i]} = G^{[i]}[G^{[i]}/X]^{i}[\Omega/X]$$

en donde queda claro que $(\mu X.G)^{[i]}$ se obtiene al iterar i veces la sustitución en $G^{[i]}$ de X por $G^{[i]}$, para acabar sustituyendo X por Ω.

Esencialmente, las dos definiciones son análogas, la diferencia puede observarse en este ejemplo para i=3:

$$(\mu X.G)^{[3]} = G^{[3]}[G^{[3]}/X]^{3}[\Omega/X]$$

$$(\mu X.G)^{(3)} = G^{(3)}[G^{(2)}/X][G^{(1)}/X][\Omega/X]$$

es decir, se podria demostrar $(\mu X.G)^{(i)} \leq (\mu X.G)^{[i]}$ usando el siguiente lema.

La elección hecha, se debe a que la definición recursiva facilita las demostraciones.

19- LEMA

Para toda $F \in FEXP_{\Sigma}^{(n)}$ y $M \in EXP_{\Sigma}$ de LRFs positivas en la variable funcional X sus aproximaciones sintácticas $F^{(i)}$ y $M^{(i)}$ también son positivas en X.

Demostración

Por inducción simultánea sobre F y M. En el caso de $F=(\mu X.G)^{(i)}$ se procede por inducción sobre i, usando el lema 12 y el hecho de que Ω es positiva en todas las variables funcionales, ya que en ella no aparece ninguna variable funcional. ■

20- LEMA

Consideremos $F \in FEXP_{\Sigma}^{(n)}$, $M \in EXP_{\Sigma}$ de LRFs, $\mathcal{D}$ una Σ-estructuray Δ y δ un contexto y una valoración sobre $\mathcal{D}$:

(a) $F^{(i)\mathcal{D}}(\Delta,\delta) \ll F^{(i+1)\mathcal{D}}(\Delta,\delta)$

(b) $M^{(i)\mathcal{D}}(\Delta,\delta) \subseteq M^{(i+1)\mathcal{D}}(\Delta,\delta)$

Demostración

Por inducción simultánea sobre las expresiones funcionales y expresiones.

Veamos el caso $\mu X.G$ para (a): por inducción sobre $i \in N$

i=0, se verifica por ser $\Omega^{\mathcal{D}}(\Delta,\delta)$ la función totalmente indefinida.

i>0, $(\mu X.G)^{(i)\mathcal{D}}(\Delta,\delta)$

$= G^{(i)\mathcal{D}}(\Delta[(\mu X.G)^{(i-1)\mathcal{D}}(\Delta,\delta)/X],\delta)$ por hipótesis de inducción sobre i y por ser $G^{(i)}$ positiva en X

$\subseteq G^{(i)\mathcal{D}}(\Delta[(\mu X.G)^{(i)\mathcal{D}}(\Delta,\delta)/X],\delta)$ por hipótesis de inducción

$\subseteq G^{(i+1)\mathcal{D}}(\Delta[(\mu X.G)^{(i)\mathcal{D}}(\Delta,\delta)/X],\delta) = (\mu X.G)^{(i+1)\mathcal{D}}(\Delta,\delta)$ como se quería demostrar. ■

21- TEOREMA

Consideremos $F \in FEXP_{\Sigma}^{(n)}$, $M \in EXP_{\Sigma}$ de LRFs, $\mathfrak{D}$ una Σ-estructuray Δ y δ un contexto y una valoración sobre $\mathfrak{D}$:

(a) $F^{\mathfrak{D}}(\Delta,\delta) = \bigsqcup_{i\in N} F^{(i)\mathfrak{D}}(\Delta,\delta)$

(b) $M^{\mathfrak{D}}(\Delta,\delta) = \bigsqcup_{i\in N} M^{(i)\mathfrak{D}}(\Delta,\delta)$

Demostración

Por inducción simultánea sobre las expreesiones funcionales y expresiones de LRFs.

Veamos el caso de $\mu X.G$ para (a):

$(\mu X.G)^{\mathfrak{D}}(\Delta,\delta) = fix(T) = \bigsqcup_{i\in N} T^i(\perp)$ donde $T(X) = G^{\mathfrak{D}}(\Delta[X/X],\delta)$.

Demostremos las dos inclusiones de

$$\bigsqcup_{i\in N} T^i(\perp) = \bigsqcup_{i\in N} (\mu X.G)^{(i)\mathfrak{D}}(\Delta,\delta)$$

(1) $\bigsqcup_{i\in N} (\mu X.G)^{(i)\mathfrak{D}}(\Delta,\delta) \subseteq \bigsqcup_{i\in N} T^i(\perp)$:

Es suficiente demostrar $(\mu X.G)^{(i)\mathfrak{D}}(\Delta,\delta) \subseteq T^i(\perp)$ por inducción sobre $i\in N$.

$i=0$, por ser $\Omega^{\mathfrak{D}}(\Delta,\delta)$ la función totalmente indefinida $\perp$.

$i>0$, $(\mu X.G)^{(i+1)\mathfrak{D}}(\Delta,\delta)$

$= G^{(i+1)\mathfrak{D}}(\Delta[(\mu X.G)^{(i)\mathfrak{D}}(\Delta,\delta)/X],\delta)$ por hipótesis de inducción

$\subseteq G^{\mathfrak{D}}(\Delta[(\mu X.G)^{(i)\mathfrak{D}}(\Delta,\delta)/X],\delta)$ por hipótesis de inducción sobre i y por ser G positiva en X

$\subseteq G^{\mathfrak{D}}(\Delta[T^i(\perp)/X],\delta) = T^{(i+1)}(\perp)$ como se quería demostrar.

(2) $\bigsqcup_{i\in N} T^i(\perp) \leq \bigsqcup_{i\in N} (\mu X.G)^{(i)\mathfrak{D}}(\Delta,\delta)$:

Es suficiente demostrar $T^i(\perp) \leq \bigsqcup_{i\in N} (\mu X.G)^{(i)\mathfrak{D}}(\Delta,\delta)$ por inducción sobre $i\in N$.

$i=0$, por ser $T^0(\perp)=\perp$ la función totalmente indefinida.

$i>0$, $T^{i+1}(\perp) = G^{\mathfrak{D}}(\Delta[T^i(\perp)/X],\delta)$ por hipótesis de inducción

$= \bigsqcup_{j\in N} G^{(j)\mathfrak{D}}(\Delta[T^i(\perp)/X],\delta)$ por hipótesis de inducción sobre i

$\leq \bigsqcup_{j\in N} G^{(j)\mathfrak{D}}(\Delta[\bigsqcup_{i\in N} (\mu X.G)^{(i)\mathfrak{D}}(\Delta,\delta)/X],\delta)$ por ser $G^{(j)}$ existencial en X

$= \bigsqcup_{j\in N} \bigsqcup_{i\in N} G^{(j)\mathfrak{D}}(\Delta[(\mu X.G)^{(i)\mathfrak{D}}(\Delta,\delta)/X],\delta)$

$= \bigsqcup_{i\in N} (\mu X.G)^{(i)\mathfrak{D}}(\Delta,\delta)$ como se queria demostrar. ■

El motivo para restringir estos resultados a LRFs, es que en LRFc tendríamos que definir de forma natural aproximaciones sintácticas para las fórmulas, con lo que a los resultados anteriores habría que añadir:

$\varphi^{(i)\mathfrak{D}}(\Delta,\delta)$ = true => $\varphi^{(i+1)\mathfrak{D}}(\delta,\delta)$ =true
$\varphi^{\mathfrak{D}}(\Delta,\delta)$ = true sii existe $i\in N$ tal que $\varphi^{(i)\mathfrak{D}}(\Delta,\delta)$ = true

lo cual es falso, como puede observarse para la fórmula de LRFc:

$\varphi = \exists z\ \neg(\mu X.\ \lambda x.\ x\cup X(x+1))\ (0) \ni z$

suponiendo que el dominio es N, $\varphi^{\mathfrak{D}}(\Delta,\delta)$ = false y para todo $i\in N$ $\varphi^{(i)\mathfrak{D}}(\Delta,\delta)$=true.

Esto se podría remediar con otra definición alternativa de aproximación sintáctica más modular, en la que no se aproximan todas las definiciones recursivas a la vez:

$(\mu X.G)^{[0]} = \Omega$
$(\mu X.G)^{[i+1]} = G[(\mu X.G)^{[i]}/X]$

en esta definición solo se itera i veces la sustitución en G de X por G para acabar sustituyendo X por Ω. De esta forma se podría demostrar directamente

$$(\mu X.G)^{[i]\mathfrak{D}}(\Delta,\delta) \leq (\mu X.G)^{[i+1]\mathfrak{D}}(\Delta,\delta)$$
$$(\mu X.G)^{\mathfrak{D}}(\Delta,\delta) = \bigsqcup_{i\in N} (\mu X.G)^{(i)\mathfrak{D}}(\Delta,\delta)$$

por inducción sobre i, sin tener que demostrarlo ni para las expresiones ni para las fórmulas, usando la existencialidad de G en X. El problema es que en $(\mu X.G)^{[i]}$ puede aparecer el operador μ y en este caso, el número de apariciones de este operador aumenta con i, por lo que no podemos encontrar una función de complejidad de forma que la complejidad de $(\mu X.G)$ sea mayor que la de sus aproximaciones.

4.- CAPACIDAD DE EXPRESION DE LRF

En esta sección queremos mostrar la capacidad de expresión de LRF. Para este propósito, vamos a demostrar como otras lógicas de programas pueden expresarse en LRF. En particular hemos escogido el μ-cálculo proposicional de Kozen PMC [Koz 83] y la lógica de primer orden dinamica para programas regulares de Harel QDL [Har 79], como representantes típicos. También vamos a demostrar que LRF puede verse como un fragmento de la lógica infinitaria clásica y que la terminación de programas no triviales no puede expresarse en el fragmento de primer orden de LRF, adaptando el conocido resultado de Kfoury y Park [KP 75].

La relación de capacidad de expresión definida en el capítulo 1, no es la que vamos a aplicar en esta sección, puesto que como se verá en cada caso, las signaturas de las dos lógicas que estamos comparando no son necesariamente la misma. El concepto queda claramente definido en los resultados que se obtienen.

PMC ES EXPRESABLE EN LRF

Dada una signatura del μ-cálculo proposicional $\Sigma=\Sigma_p \cup \Sigma_a$, (donde Σ_p es un conjunto de símbolos de proposición denotados por p, q y Σ_a es un conjunto de símbolos de acciones atómicas denotadas por a, b) y conjunto de variables proposicionales PVAR={P, Q, R, ...}, las Σ-fórmulas A, B de PMC se definen mediante la siguiente regla BNF:

$$A ::= p \mid P \mid \langle a \rangle B \mid \neg B \mid (B_1 \vee B_2) \mid \mu P.B$$

donde B debe ser positiva en P en $\mu P.B$; es decir, cada aparición de P en B debe estar dentro del alcance de un número par de símbolos de negación.

Una Σ-estructura para PMC es un sistema

$$\mathfrak{D} = \langle D,\ (p^{\mathfrak{D}})_{p\in\Sigma_p}\ ,\ (a^{\mathfrak{D}})_{a\in\Sigma_a}\rangle$$

donde D es un conjunto no vacío (el conjunto de estados), $p^{\mathfrak{D}}\subseteq D$ para todo $p\in\Sigma_p$, y $a^{\mathfrak{D}}\subseteq D\times D$ para todo $a\in\Sigma_a$.

Un contexto sobre una Σ-estructura $\mathfrak{D}$ para PMC es una aplicación Δ del conjunto de las variables proposicionales $P\in PVAR$ al conjunto de estados $\Delta(P)\subseteq D$.

Dada una Σ-fórmula A y una Σ-estructura $\mathfrak{D}$ para PMC, la denotación $A^{\mathfrak{D}}(\Delta)$ de A bajo un contexto Δ sobre $\mathfrak{D}$ es el subconjunto de D definido por:

$p^{\mathfrak{D}}(\Delta) = p^{\mathfrak{D}}$

$P^{\mathfrak{D}}(\Delta) = \Delta(P)$

$(\langle a\rangle B)^{\mathfrak{D}}(\Delta) = \{x\in D$ / existe $y\in B^{\mathfrak{D}}(\Delta)$ tal que $xa^{\mathfrak{D}}y\}$

$(\neg B)^{\mathfrak{D}}(\Delta) = D \setminus B^{\mathfrak{D}}(\Delta)$

$(B_1 \vee B_2)^{\mathfrak{D}}(\Delta) = B_1^{\mathfrak{D}}(\Delta) \cup B_2^{\mathfrak{D}}(\Delta)$

$(\mu P.B)^{\mathfrak{D}}(\Delta)$ = el menor punto fijo del operador monótono

$T=T(\mathfrak{D},\Delta,P,B)$: $\mathfrak{P}(D) \longrightarrow \mathfrak{P}(D)$, definido por $T(P)=B^{\mathfrak{D}}(\Delta[P/P])$

Una definición más detallada de esta definición puede encontrarse en [Koz 83].

Con la idea de expresar cada fórmula de PMC mediante una fórmula de LRF_M asociamos a cada signatura $\Sigma = \Sigma_p \cup \Sigma_a$ de PMC una signatura $\hat{\Sigma}=\hat{\Sigma}_{pd}$ de LRF de forma que $\hat{\Sigma}_{pd}$ contiene: para cada $p \in \Sigma_p$ un símbolo de predicado monádico q_p y para cada $a\in \Sigma_a$ un símbolo de predicado binario r_a.

Además asociamos a cada variable proposicional P de PMC una variable funcional 0-aria X_p de LRF.

Con esto, fijando arbitrariamente una variable de individuo x, podemos traducir cualquier Σ-fórmula A de PMC a una $\hat{\Sigma}$-fórmula $\varphi_A(x)$ de LRF siendo x su única variable libre, como sigue:

$A=p$: $\varphi_A(x) = q_p(x)$

$A=P$: $\varphi_A(x) = X_P \ni x$

$A=\langle a\rangle B$: $\varphi_A(x) = \exists y(P_a(x,y) \wedge \varphi_B(y))$

$A=\neg B$: $\varphi_A(x) = \neg\varphi_B(x)$

$A=B_1 \vee B_2$: $\varphi_A(x) = \varphi_{B1}(x) \vee \varphi_{B2}(x)$

$A=\mu P.B$: $\varphi_A(x) = \mu X_P.(\lambda.\varepsilon x\varphi_B(x)) \ni x$

1- LEMA

Para toda fórmula $A=\mu P.B$ (y por tanto B positiva en P) de PMC, la expresión funcional de LRFH $\lambda.\varepsilon x\varphi_B(x)$ asociada a $\varphi_A(x) = \mu X_P.(\lambda.\varepsilon x\varphi_B(x)) \ni x$ por la anterior traducción es positiva en X_P.

Demostración

Por inducción sobre B. ■

Para comparar semánticamente A y φ_A, asociamos ahora una $\hat{\Sigma}$-estructura $\hat{\mathfrak{D}}$ para LRF, junto con un contexto $\hat{\Delta}$ y una valoración $\hat{\delta}$, a cada $\mathfrak{D}$ y Δ para PMC, como sigue:

el dominio de $\hat{\mathfrak{D}}$ es el conjunto de estados D de $\mathfrak{D}$.

$q_p^{\hat{\mathfrak{D}}}(x)$ = true sii $x \in p^{\mathfrak{D}}$ para todo $p \in \Sigma_p$ y $x \in D$

$r_a^{\hat{\mathfrak{D}}}(x,y)$ = true sii $xa^{\mathfrak{D}}y$, para todo $a \in \Sigma_a$ y $x,y \in D$.

$\hat{\Delta}(X_P) = \Delta(P)$ para todo $P \in PVAR$.

en los demás casos $\hat{\Delta}$ y $\hat{\delta}$ pueden definirse arbitrariamente.

2- TEOREMA

Dada una Σ-fórmula A de PMC, una Σ-estructura $\mathfrak{D}$ para PMC, y un contexto Δ sobre $\mathfrak{D}$, se verifica:

$$A^{\mathfrak{D}}(\Delta) = \{x \in D \;/\; \hat{\mathfrak{D}} \vdash \varphi_A(x)(\hat{\Delta}, \hat{\delta}[x/x])\}$$

Demostración

Por inducción sobre la estructura de A. Veamos el caso $A=\mu P.B$:

$A^{\mathfrak{D}}(\Delta) = fix(T_B)$, donde $T_B(P) = B^{\mathfrak{D}}(\Delta[P/P]) \subseteq D$.

$\varphi_A^{\mathfrak{D}}(\hat{\Delta}, \hat{\delta}[x/x]) = fix(T)$ donde $T(P) = \{ x \in D \mid \varphi_B^{\mathfrak{D}}(\hat{\Delta}[P/X_P], \hat{\delta}[x/x]) \}$

(obsérvese que las funciones de $\mathfrak{N}\mathfrak{V}^{(0)}$ pueden identificarse con subconjuntos del dominio D). Por hipótesis de inducción para B, teniendo en cuenta que $\Delta[\hat{P}/P] = \hat{\Delta}[P/X_P]$, $T_B = T$ y por tanto, $fix(T) = fix(T_B)$ como se quería demostrar. ■

Obsérvese que es necesario usar en esta traducción de PMC a LRF la variante LRF_M, por que PMC no es continuo, solo es monótono.

QDL ES EXPRESABLE EN LRF

Sea $\Sigma = \Sigma_{fn} \cup \Sigma_{pd}$ una signatura de primer orden y VAR un conjunto de variables de individuo, como en la Subsección 2.1. Los conjuntos de Σ-fórmulas A,B y Σ-programas α,β de QDL se definen mediante las siguientes reglas BNF:

$$A ::= p(t_1,\ldots,t_n) \mid \neg B \mid B_1 \vee B_2 \mid \exists x B \mid \langle\alpha\rangle B$$
$$\alpha ::= x:=t \mid (\beta_1;\beta_2) \mid (\beta_1 \cup \beta_2) \mid \beta^* \mid A? \text{ (A fórmula de primer orden)}$$

donde $t, t_1, \ldots, t_n$ son Σ-términos.

Las Σ-estructuras $\mathfrak{D}$ para QDL son análogas a las Σ-estructuras para LRF. Las denotaciones de los Σ-términos, Σ-fórmulas y Σ-programas de QDL con respecto a una Σ-estructura $\mathfrak{D}$ dada y una valoración δ sobre $\mathfrak{D}$ se definen como sigue:

(a) $t^{\mathfrak{D}}(\delta) \in D$ para $t \in \Sigma$-términos: como en lógica de primer orden.

(b) $\alpha^{\mathfrak{D}} \subseteq St_D \times St_D$, donde St_D es el conjunto de todas las valoraciones $\delta: VAR \longrightarrow D$, y

$(x:=t)^{\mathfrak{D}}$ = $\{\langle\delta, \delta[t^{\mathfrak{D}}(\delta)/x]\rangle \;/\; \delta \in St_{\mathfrak{D}}\}$

$(\beta_1;\beta_2)^{\mathfrak{D}} = \beta_1^{\mathfrak{D}} \circ \beta_2^{\mathfrak{D}}$ = $\{\langle\delta_1,\delta_2\rangle$ / existe δ_3 tal que $\langle\delta_1,\delta_3\rangle \in \beta_1^{\mathfrak{D}}$ y $\langle\delta_3,\delta_2\rangle \in \beta_2^{\mathfrak{D}}\}$

$(\beta_1 \cup \beta_2)^{\mathfrak{D}} = \beta_1^{\mathfrak{D}} \cup \beta_2^{\mathfrak{D}}$

$(\beta^*)^{\mathfrak{D}}$ = la clausura reflexiva y transitiva de $\beta^{\mathfrak{D}}$.

$(A?)^{\mathfrak{D}}$ = $\{\langle\delta,\delta\rangle \;/\; A^{\mathfrak{D}}(\delta)=\text{true}\}$

(c) $A^{\mathfrak{D}}(\delta) \in \{\text{true}, \text{false}\}$ para Σ-fórmulas A de QDL:
Como en lógica de primer orden, junto con:

$$(\langle\alpha\rangle B)^{\mathfrak{D}}(\delta) = \begin{cases} \text{true} & \text{si existe } \delta' \text{ tal que } \langle\delta,\delta'\rangle \in \alpha^{\mathfrak{D}} \text{ y } B^{\mathfrak{D}}(\delta') = \text{true} \\ \text{false} & \text{en otro caso} \end{cases}$$

Como Goerdt en [Goe 87], consideramos un número k arbitrario pero fijo, para trabajar con el fragmento QDL_k de QDL cuyos programas y fórmulas usan a lo más k variables libres $x_1,\ldots,x_k$. Para traducir QDL_k a LRFs, extendemos Σ, a la signatura $\hat{\Sigma}$, con un nuevo símbolo de predicado monádico ind, k nuevos símbolos de función binarios asg_j $(1 \le j \le k)$, k nuevos símbolos de función monádicos pr_j $(1 \le j \le k)$, y un nuevo símbolo de función k-ario state.

Dada cualquier Σ-estructura $\mathfrak{D}$, consideramos la $\hat{\Sigma}$-estructura $\hat{\mathfrak{D}}$ cuyo dominio es $\hat{D} = D \cup D^k$ y donde los símbolos de $\hat{\Sigma}$ son interpretados como sigue:

(1) $asg_j^{\hat{\mathfrak{D}}} : \hat{D} \times \hat{D} \to \hat{D}$

$asg_j^{\hat{\mathfrak{D}}}(\langle x_1,\ldots,x_k\rangle, x) = \langle x_1,\ldots,x_{j-1},x,x_{j+1},\ldots,x_k\rangle \in D^k$ y arbitrariamente en los demás casos.

(2) $pr_j^{\hat{\mathfrak{D}}} : \hat{D} \to \hat{D}$

$pr_j^{\hat{\mathfrak{D}}}(\langle x_1,\ldots,x_k\rangle) = x_j \in D^k$ y arbitrario en otro caso.

(3) $ind^{\hat{\mathfrak{D}}} : \hat{D} \longrightarrow \{true, false\}$

$ind^{\hat{\mathfrak{D}}}(<x_1,\ldots,x_k>) = false$ para $x_1,\ldots,x_k \in D$

$ind^{\hat{\mathfrak{D}}}(x) = true$ para $x \in D$

(4) $state^{\hat{\mathfrak{D}}}: \hat{D} \longrightarrow \hat{D}$

$state^{\hat{\mathfrak{D}}}(x_1,\ldots,x_k) = <x_1,\ldots,x_k> \in D^k$ para $x_1,\ldots,x_k \in D$ y arbitrariamente en los demás casos.

(5) Los símbolos de Σ se interpretan de forma que para argumentos del dominio D, su valor sea el mismo que el de sus interpretaciones en $\mathfrak{D}$.

Dada una valoración δ: VAR $\longrightarrow$ D, la extendemos a una valoración $\hat{\delta}$ de V=VAR$\cup$FVAR sobre $\hat{D}$ arbitrariamente. Ahora, fijamos una variable de individuo x de LRF (arbitraria pero diferente de $x_1,\ldots,x_k$) y definimos una traducción sintáctica asignando:

(a)Una $\hat{\Sigma}$-expresión funcional monádica F_α de LRF a cada Σ-programa α de QDL_k

(b)Una $\hat{\Sigma}$-fórmula φ_A de LRF a cada Σ-fórmula A de QDL_k como sigue:

(i) $\alpha = x_j := t$: $F_\alpha = \lambda x.(\neg ind(x) \rightarrow asg_j(x, t[pr_1(x)/x_1 \ldots pr_k(x)/x_k]))$

$\alpha=\beta_1 \cup \beta_2$: $F_\alpha = \lambda x.(F_{\beta 1}(x) \cup F_{\beta 2}(x))$

$\alpha=\beta_1;\beta_2$: $F_\alpha = \lambda x.F_{\beta 2}(F_{\beta 1}(x))$

$\alpha=A?$: $F_\alpha = \lambda x.(\neg ind(x) \rightarrow (A[pr_1(x)/x_1 \ldots pr_k(x)/x_k] \rightarrow x))$

$\alpha=\beta^*$: $F_\alpha = \mu X.(\lambda x.x \cup X(F_\beta(x)))$

(ii) $A=p(t_1,\ldots,t_n)$: $\varphi_A = p(t_1,\ldots,t_n)$

$A=\neg B$: $\varphi_A = \neg\varphi_B$

$A=B_1 \vee B_2$: $\varphi_A = \varphi_{B1} \vee \varphi_{B2}$

$A=\exists y B$: $\varphi_A = \exists y(ind(y) \wedge \varphi_B)$

$A=<\alpha>B$: $\varphi_A = \exists z(\neg ind(z) \wedge F_\alpha(state(x_1,\ldots,x_k)) \ni z \wedge \varphi_B[pr_1(z)/x_1 \ldots pr_k(z)/x_k])$

(donde z es una variable que no aparece en A)

3- LEMA

Para todo programa $\alpha=\beta^*$ de QDL, la expresión funcional de LRFs $\lambda x.\, x \cup X(F_\beta(x))$ asociada a $F_\alpha = \mu X.(\lambda x.\, x \cup X(F_\beta(x)))$ por la anterior traducción es existencial en X.

Demostración

Por inducción sobre β. ∎

El siguiente resultado establece la corrección semántica de la traducción.

4- TEOREMA

Supuesto que $\hat{\mathfrak{D}}$ y $\hat{\delta}$ estan asociadas a $\mathfrak{D}$ y δ como se acaba de exponer. Entonces:

(a) $t^{\mathfrak{D}}(\delta) = t^{\hat{\mathfrak{D}}}(\delta)$ para todo $t \in TER_\Sigma$.

(b) $\langle y_1,\ldots,y_k\rangle \in F_\alpha^{\hat{\mathfrak{D}}}(\delta)(\langle x_1,\ldots,x_k\rangle)$ sii $\langle \delta[x_1/x_1,\ldots,x_k/x_k], \delta[y_1/x_1,\ldots,y_k/x_k]\rangle \in \alpha^{\mathfrak{D}}$ para todo $x_1,\ldots,x_k,y_1,\ldots,y_k \in D$ y todo Σ-programa α de QDL_k.

(c) $\varphi_A(\hat{\delta}) =$ true sii $A^{\mathfrak{D}}(\delta) =$ true para toda Σ-formula A de QDL_k.

Demostración

(a) es trivial. (b) y (c) por inducción simultánea sobre la estructura de α y A.

Veamos el caso $\alpha=\beta^*$ para (b): como el operador T asociado a F_α definido por $T(X)=(\lambda x.\, x \cup X(F_\beta(x)))^{\hat{\mathfrak{D}}}(\Delta[X/X], \delta)$ es continuo,

$F_\alpha^{\hat{\mathfrak{D}}}(\Delta,\delta) = fix(T) = \bigsqcup_{i\in\mathbb{N}} T^i$ (considerando T^0 la identidad), y por la semántica de β^*, es suficiente demostrar por inducción sobre i, identificando β^0 y F^0 con la identidad:

(1) $F_{\beta^i}(x) = \lambda x.\, F_\beta^i(x)$

(2) $T^i(\perp) = \bigsqcup_{j<i} F_\beta^{j\mathfrak{D}}(\delta)$ ∎

LRF ES EXPRESABLE EN $EL_{\omega_1\omega}$

Llamamos "$L\omega_1\omega$ extendida" (en símbolos $EL\omega_1\omega$) a la lógica infinitaria clásica $L\omega_1\omega$ (cfr. [Kei 71]) extendida de forma análoga a como hemos extendido la lógica de primer orden usual en la sección 1. Más exactamente: Las signaturas para $EL\omega_1\omega$ son las mismas que las signaturas para la lógica de primer orden extendida; los Σ-términos t de $EL\omega_1\omega$ los Σ-términos de primer orden; y las Σ-fórmulas A,B de $EL\omega_1\omega$ tienen la siguiente sintaxis:

$$A::= p(t_1,\ldots,t_n) \mid t_1=t_2 \mid X(t_1,\ldots,t_n)\ni t \mid \neg B \mid (B_1 \vee B_2) \mid \exists x B \mid \bigvee_{i\in N} B_i$$

es decir, a las fórmulas de EL_1 se le añade la formación de disyunciones numerables. Las conjunciones numerables $\bigwedge_{i\in N} B_i$ se consideran abreviaciones de $\neg \bigvee_{i\in N} \neg B_i$. La semántica de $EL\omega_1\omega$ es la extensión natural de la semántica de la lógica de primer orden extendida.

La mayoría de las lógicas de programas con semántica estándar son fragmentos de $L\omega_1\omega$. LRFs es también un fragmento de $EL\omega_1\omega$. Para comprobar esto, fijamos una signatura Σ y definimos una traducción sintáctica de las Σ-expresiones funcionales, Σ-expresiones y Σ-fórmulas de LRFs a Σ-fórmulas de $EL\omega_1\omega$.

(a) Σ-expresiones funcionales. Dada $F\in FEXP_\Sigma^{(n)}$ y n+1 variables nuevas $u_1,\ldots,u_n,v$, distintas dos a dos y sin apariciones en F, construimos una Σ-fórmula $A_F(u_1,\ldots,u_n,v)$ de $EL\omega_1\omega$ como sigue:

F=f: $\quad A_F(u_1,\ldots,u_n,v) = (f(u_1,\ldots,u_n)=v)$

F=X: $\quad A_F(u_1,\ldots,u_n,v) = X(u_1,\ldots,u_n)\ni v$

$F=\lambda x_1\ldots x_n.M$: $\quad A_F(u_1,\ldots,u_n,v) = A_M(v)[u_1/x_1,\ldots,u_n/x_n]$

$F=\mu X.G$: $\quad A_F(u_1,\ldots,u_n,v) = \bigvee_{i\in N} A_{(\mu X.G)^{(i)}}(u_1,\ldots,u_n,v)$

donde $(\mu X.G)^{(i)}$ son las aproximaciones sintacticas de la definición 18 de la sección 3. Obsérvese que en este caso, $B_{(\mu X.G)^{(i)}}$ se reduce a los casos anteriores puesto que el operador μ no aparece en $(\mu X.G)^{(i)}$.

(b) Σ-expresiones. Dada $M \in EXP_\Sigma$ una variable nueva v sin apariciones en M, construimos una Σ-formula $A_M(v)$ de $EL\omega_1\omega$ como sigue:

$M=t$: $A_M(v) = (t=v)$

$M=F(N_1,\ldots,N_n)$ $A_M(v) = \exists u_1 \ldots \exists u_n \ (A_{N1}(u) \wedge \ldots \wedge A_{Nn}(u) \wedge A_F(u_1,\ldots u_n,v))$
siendo $u_1,\ldots,u_n$ nuevas y distintas dos a dos.

$M=N_1 \cup N_2$: $A_M(v) = A_{N1}(v) \vee A_{N2}(v)$

$M=\varphi \rightarrow N$: $A_M(v) = \varphi \wedge A_N(v)$

$M=cx\varphi$: $A_M(v) = \varphi[v/x]$

(c) Σ-fórmulas. Dada $\varphi \in FFOR_\Sigma$ construimos una Σ-fórmula A_φ de $EL\omega_1\omega$ como sigue:

$\varphi=(t_1=t_2)$: $A_\varphi = (t_1=t_2)$

$\varphi=p(M_1,\ldots,M_n)$: $A_\varphi = \exists v_1 \ldots \exists v_n (A_{M1}(v_1) \wedge \ldots \wedge A_{Mn}(v_n) \wedge p(v_1,\ldots,v_n))$
siendo $v_1,\ldots,v_n$ nuevas y distintas dos a dos.

$\varphi=M \ni t$: $A_\varphi = A_M(v)[t/v]$, siendo v nueva

$\varphi=\neg\psi$: $A_\varphi = \neg A_\psi$

$\varphi=\psi_1 \vee \psi_2$: $A_\varphi = A_{\psi_1} \vee A_{\psi_2}$

$\varphi=\exists x\psi$: $A_\varphi = \exists x A_\psi$

Obsérvese que si en F (resp. M y φ) no aparece el operador μ, entonces A_F (resp. A_M y A_φ) son fórmulas de primer orden extendido (EL_1).

El siguiente resultado asegura la corrección de la traducción:

5- TEOREMA

Para cualquier Σ-structura $\mathfrak{D}$, contexto Δ sobre $\mathfrak{D}$ y valoración δ sobre $\mathfrak{D}$, se verifica:

(a) $grafo((F^{\mathfrak{D}}(\Delta,\delta)) = \{\langle x_1,\ldots,x_n,y\rangle \in D^{n+1} \ / \ A_F^{\mathfrak{D}}(\Delta,\delta[x_1/u_1,\ldots,x_n/u_n,y/v])=true\}$

(b) $M^{\mathfrak{D}}(\Delta,\delta) = \{x \in D \ / \ A_M^{\mathfrak{D}}(\Delta,\delta[x/v])= true\}$

(c) $\varphi^{\mathfrak{D}}(\Delta,\delta) = true$ sii $A_\varphi^{\mathfrak{D}}(\Delta,\delta) = true$

Demostración

Por inducción simultánea sobre la estructura de F, M y φ. Veamos el caso $F=(\mu X.G)$ para (a): Por el teorema 21 de la sección 3, $(\mu X.G)^{\mathfrak{D}}(\Delta,\delta) = \bigsqcup_{i\in N} (\mu X.G)^{(i)\mathfrak{D}}(\Delta,\delta)$ y por definición $A_F(u_1,\ldots,u_n,v) = \bigvee_{i\in N} A_{(\mu X.G)^{(i)}}(u_1,\ldots,u_n,v)$, por hipótesis de inducción, $A_{(\mu X.G)^{(i)}}(u_1,\ldots,u_n,v)$ define el grafo de $(\mu X.G)^{(i)}$, para todo $i\in N$, de lo que se concluye lo que se quería demostrar. ■

LA TERMINACION DE PROGRAMAS NO ES EXPRESABLE EN LRF

En el marco de LRF, la terminación requiere la evaluación de expresiones. Su indeterminismo permite considerar varias nociones de terminación ; ver Harel [Har 79] y [Har 84]. Aqui vamos a considerar solamente un tipo de terminación débil y vamos a demostrar que este concepto no es de primer orden en el marco semántico de LRF (excepto para casos triviales). Para otros tipos de terminación podrían obtenerse resultados análogos.

6- DEFINICION

Sea $M\in EXP_\Sigma$, $\Phi\subseteq FFOR_\Sigma$

(a) M termina debilmente en la Σ-estructura $\mathfrak{D}$ sii $M^{\mathfrak{D}}(\Delta,\delta)\neq\varnothing$, para todo Δ,δ sobre $\mathfrak{D}$.
(b) Φ asegura la terminacion débil de M sii M termina debilmente en todos los modelos de Φ.
(c) M es semánticamente libre de recursión c.r.a Φ sii existe alguna Σ-expresión N sin apariciones del operador μ, tal que $M^{\mathfrak{D}}(\Delta,\delta)=N^{\mathfrak{D}}(\Delta,\delta)$ para todo modelo $\mathfrak{D}\models\Phi$ y todo Δ, δ sobre $\mathfrak{D}$. ■

Dada $M\in EXP_\Sigma$, es fácil encontrar $\Phi\subseteq FFOR_\Sigma$ que asegure la terminación debil de M. Para esto es suficiente tomar $\Phi=\{\exists v\ M\exists v\}$, donde v es una

variable nueva. Si restringimos Φ a fórmulas de primer orden extendido, obtenemos el siguiente resultado (similar al conocido de Kfoury y Park [KP 75]).

7- TEOREMA

Sea $M \in EXP_\Sigma$ de LRFs y $\Phi \subseteq EFOR_\Sigma$. Si Φ asegura la terminación débil de M, entonce M es semánticamente libre de recursión c.r.a Φ.

Demostración

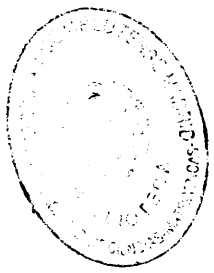
Supongamos la hipótesis. Por el Teorema 5, sabemos que M puede ser traducida a una Σ-fórmula $A_M(v)$ de $EL\omega_1\omega$, tomando como v una variable nueva que no aparecca en M. La traducción de las expresiones puede refinarse de forma que para M, $A_M(v)$ sea una fórmula de EL_1 o de la forma $\bigvee_{i\in N} B_i(v)$, donde $B_i(v) \in EFOR_\Sigma$ para todo $i \in N$. En efecto:

Para las expresiones funcionales y expresiones en las que no aparece el operador μ, definimos A_F y A_M como antes (por lo que estas fórmulas son de primer orden extendido), y para M en la que aparece el operador μ, aplicando que M es equivalente a la unión de sus aproximaciones, y que en sus aproximaciones no aparece el operador μ, definimos $A_M(v) = \bigvee_{i\in N} A_{M^{(i)}}(v)$.

Luego si $A_M(v)$ es de primer orden extendido, tomando $N = cv.A_M(v)$, tenemos M es semánticamente libre de recursión con respecto a Φ.

En otro caso $A_M(v) = \bigvee_{i\in N} B_i(v)$, consideremos el conjunto $\Phi \subseteq EFOR_\Sigma$ que asegura la terminación débil de M. Tomemos una constante nueva c. El conjunto $\Phi \cup \{\neg B_i(v)[c/v] \ / \ i \in N\}$ es obviamente insatisfactible. Por compacidad de la lógica de primer orden extendida, existe $k \in N$ tal que $\Phi \cup \{\neg B_i(v)[c/v] \ / \ i \leq k\}$ es insatisfactible. Esto significa que $\bigvee_{i\leq k} B_i(v)$ es una traducción válida de M en la lógica de primer orden extendida en todos los modelos de Φ. Por tanto, $N = cv \bigvee_{i\leq k} B_i(v)$ es una Σ-expresión sin apariciones de μ y tal que M es equivalente a N sobre todos los modelos $\mathfrak{D} \models \Phi$; es decir: M es semánticamente libre de recursión c.r.a Φ. ■

Como dijimos en el capítulo 1, la terminación de programas si es expresable dentro del marco no estándar de lógicas de programas. En el capítulo 4, veremos algunos ejemplos de terminación de programas que son posibles en la lógica no estándar que se presenta el capítulo, y no lo son en LRF.



CAPITULO 3:
CALCULOS DEDUCTIVOS PARA LRF

Del hecho de que QDL es expresable en LRFs y que el problema de validez para QDL es Π^1_1-completo, se deduce que el problema de validez para LRFs es Π^1_1-duro, y por tanto, que no existen cálculos finitarios correctos y completos para LRFs. El motivo es que el concepto de menor punto fijo de un operador continuo no es un concepto de primer orden, ya que entraña cuantificaciones existenciales sobre números naturales y el esquema de inducción.

Ante este problema procedemos de manera análoga a como se procedió en la lógica dinámica, expuesto en el capítulo 1. En este capítulo abordamos este problema dentro del marco de la semántica estándar. En primer lugar, el teorema 21 de la sección 2.3, según el cual

$$(\mu X.G) \text{ es equivalente a } \bigsqcup_{i\in N}(\mu X.G)^{(i)}$$

nos permite obtener un cálculo infinitario correcto y completo para LRFs, con una regla infinitaria que refleja el hecho de que cualquier elemento x que no pueda ser obtenido como valor de $T^i(\perp)$ para algún i, tampoco puede ser un valor de fix(T).

Por otro lado, usando la caracterización del menor punto fijo del teorema 12 de la sección 2.2, demostramos que el menor punto fijo de un operador continuo puede definirse en primer orden sobre estructuras aritméticas, ya que en este tipo de estructuras podemos cuantificar números naturales, codificar grafos finitos de funciones y aplicar el principio de inducción. Estas características de las estructuras aritméticas nos permiten definir

$(\mu X.G)(\vec{t})\ni t \iff \exists i\in N$ y una T-derivación de longitud i que produce $\langle\vec{t}, t\rangle$

Recuerdese lo dicho en la sección 2.2 acerca del problema adicional que suponen las funciones indeterministas, por el cual no podemos usar para este proposito la caracterización anterior de menor punto fijo, ya que

$$\exists i \in N \text{ tal que } (\mu X.G)^{(i)}(\vec{t}) \ni t$$

no puede codificarse si $(\mu X.G)^{(i)}(\vec{t})$ es un conjunto infinito, como se vió en el ejemplo 13 de la sección 2.2.

La definibilidad del menor punto fijo en estas estructuras, nos permite obtener un cálculo finitario aritméticamente correcto y completo para LRFc.

En ambos casos, los cálculos que presentamos, junto con los axiomas y reglas de primer orden, constan de reglas de reducción orientadas a la sintaxis, en el sentido de que reducen el significado de un programa al de sus constituyentes o partes, para cada una de las posibles estructuras de una expresión, de forma análoga a como se definió en la semántica. En el cálculo infinitario el operador μ, se reduce a su semántica operacional de iteración del cuerpo de su definición hasta encontrar el valor requerido, iterando simultáneamente todas las definiciones recursivas que aparecen en el cuerpo. Las reglas del cálculo aritmético para el operador μ son más composicionales, puesto que en ellas se hace referencia directa al cuerpo de la definición.

En el siguiente capítulo, abordaremos este problema a través de una semántica no estándar para LRF, en la que además será posible razonar sobre propiedades de los programas de LRF, que no pueden razonarse con la semántica estándar, como la terminación de programas.

1.- UN CALCULO INFINITARIO PARA LRF

En esta sección presentamos un cálculo infinitario correcto y completo para LRFs, en el mismo sentido que el cálculo infinitario de la lógica dinámica. En la lógica dinámica, la regla infinitaria está basada en el hecho de que todo estado alcanzado por el operador de iteración * puede ser alcanzado en algun número $i \in \mathbb{N}$ de pasos. Para LRF la regla infinitaria está basada en la equivalencia del operador μ y sus aproximaciones sintácticas, lo que exige que nos restrinjamos a LRFs.

La prueba de completitud que presentamos para este cálculo, se basa en el método de los Tableaux, aunque podía haberse obtenido de hecho de tantas formas como demostraciones de completitud para $L\omega_1\omega$ se han obtenido. Hemos escogido el método de los tableaux, puesto que es un procedimiento natural para el problema de satisfactibilidad, con el que además se pueden obtener condiciones suficientes sobre los cálculos para asegurar su completitud.

De la demostración de completitud, podremos concluir el teorema de Löwenheim-Skolem para LRFs. Este último resultado a su vez, nos permitirá demostrar que el problema de validez para LRFs es Π^1_1, que junto con el hecho ya demostrado en el capítulo anterior, de que QDL es expresable en LRFs, nos permitirá concluir, que el problema de validez para LRFs es Π^1_1-completo.

En primer lugar presentamos el método de los Tableaux para LRF.

EL METODO DE LOS TABLEAUX PARA LRF

El método de los tableaux [Bet 59] es un procedimiento elegante y natural para resover el problema de satisfactibilidad, de forma que se va reduciendo la cuestión de la satisfactibilidad de una fórmula a de sus componentes, y al mismo tiempo es una técnica para las demostraciones de completitud de cálculos.

En este apartado, presentamos el método de los Tableaux, siguiendo la formulación de Smullyan [Smu 86], para obtener condiciones suficientes de completitud para cálculos en el marco de LRF.

Para esto, denotamos por $\bar{\Sigma}$ a la signatura obtenida al extender una signatura Σ con un conjunto numerable C de símbolos de constantes nuevos. Y distinguimos siete categorías disjuntas de $\bar{\Sigma}$-fórmulas, a las cuales les asociamos un conjunto (finito o numerable) de $\bar{\Sigma}$-fórmulas más sencillas (con respecto a una función de complejidad que definiremos a continuación), llamadas constituyentes, en función de ciertas condiciones de satisfactibilidad entre las fórmulas y sus constituyentes. Esta clasificación es la siguiente, siendo $\Phi \subseteq FFOR_{\bar{\Sigma}}$:

(0) Fórmulas básicas: consiste de las fórmulas atómicas y sus negaciones. No tienen constituyentes.

$$t_1 = t_2,\ \neg t_1 = t_2,\ P(\vec{t}),\ \neg P(\vec{t}),\ X(\vec{t})\ni t \text{ y } \neg X(\vec{t})\ni t.$$

(1) Fórmulas conjuntivas o alfa: las denotamos por α y a sus constituyentes por α_1, tienen dos constituyentes y su relación es:

$$Sat(\Phi \cup \{\alpha\}) \iff Sat(\Phi \cup \{\alpha, \alpha_1, \alpha_2\})$$

α	α_1	α_2
$\neg\neg\psi$	ψ	ψ
$\neg(\psi_1\vee\psi_2)$	$\neg\psi_1$	$\neg\psi_2$
$t_1\ni t$	$t_1=t$	$t_1=t$
$\neg t_1\ni t$	$\neg t_1=t$	$\neg t_1=t$
$\epsilon x\varphi\ni t$	$\varphi[t/x]$	$\varphi[t/x]$
$\neg\epsilon x\varphi\ni t$	$\neg\varphi[t/x]$	$\neg\varphi[t/x]$
$(\lambda\vec{x}.M)(\vec{t})\ni t$	$M[\vec{t}/\vec{x}]\ni t$	$M[\vec{t}/\vec{x}]$
$\neg(\lambda\vec{x}.M)(\vec{t})\ni t$	$\neg M[\vec{t}/\vec{x}]\ni t$	$\neg M[\vec{t}/\vec{x}]$
$(\varphi\rightarrow N)\ni t$	φ	$N\ni t$
$\neg(N_1\cup N_2)\ni t$	$\neg N_1\ni t$	$\neg N_2\ni t$
$P(\vec{N})$	$\exists\vec{x}\bigwedge_{1\le i\le n}N_i\ni x_i\wedge P(\vec{x})$	$\exists\vec{x}\bigwedge_{1\le i\le n}N_i\ni x_i\wedge P(\vec{x})$
$\neg P(\vec{N})$	$\neg\exists\vec{x}\bigwedge_{1\le i\le n}N_i\ni x_i\wedge P(\vec{x})$	$\neg\exists\vec{x}\bigwedge_{1\le i\le n}N_i\ni x_i\wedge P(\vec{x})$
$F(\vec{N})\ni t$	$\exists\vec{x}\bigwedge_{1\le i\le n}N_i\ni x_i\wedge F(\vec{x})\ni t$	$\exists\vec{x}\bigwedge_{1\le i\le n}N_i\ni x_i\wedge F(\vec{x})\ni t$
$\neg F(\vec{N})\ni t$	$\neg\exists\vec{x}\bigwedge_{1\le i\le n}N_i\ni x_i\wedge F(\vec{x})\ni t$	$\neg\exists\vec{x}\bigwedge_{1\le i\le n}N_i\ni x_i\wedge F(\vec{x})\ni t$

en los cuatro últimos casos, debe verificarse que $N_j\notin TER_\Sigma$ para algún $1\le j\le n$, las variables $\vec{x}$ sean distintas entre si y para todo $1\le i\le n$, $x_i\notin lib(\alpha)$

(2) Fórmulas disyuntivas o beta: las denotamos por β y a sus constituyentes por β_1, tiene dos constituyentes y su relación es:

$$Sat(\Phi\cup\{\beta\}) \Leftrightarrow Sat(\Phi\cup\{\beta,\beta_1\}) \text{ o } Sat(\Phi\cup\{\beta,\beta_2\})$$

β	β_1	β_2
$\psi_1\vee\psi_2$	ψ_1	ψ_2
$\neg(\varphi\rightarrow N)\ni t$	$\neg\varphi$	$\neg N\ni t$
$(N_1\cup N_2)\ni t$	$N_1\ni t$	$N_2\ni t$

(3) Fórmulas universales o gamma: las denotamos por γ y a sus constituyentes por $\gamma(t)$. Tienen una cantidad numerable de constituyentes (para cada $t\in TER_\Sigma$) y su relación es:

$$Sat(\Phi\cup\{\gamma\}) \Leftrightarrow Sat(\Phi\cup\{\gamma,\gamma(t)\})$$

γ	$\gamma(t)$
$\neg\exists x\psi$	$\neg\psi[t/x]$

(4) Fórmulas existenciales o eta: las denotamos por η y a sus constituyentes por $\eta(t)$. Tienen una cantidad numerable de constituyentes (para cada $t\in TER_{\Sigma}$) y su relación es:

$Sat(\Phi\cup\{\eta\}) <=> Sat(\Phi\cup\{\eta,\eta(c)\})$ siendo $c\in C$ una constante nueva, es decir, que no aparece ni en Φ in en η.

η	$\eta(t)$
$\exists x\psi$	$\psi[t/x]$

(5) Fórmulas ω-conjuntivas u ω-alfa: las denotamos por $\alpha^{\bullet}$ y a sus constituyentes por $\alpha_i^{\bullet}$. Tienen una cantidad numerable de constituyentes (para cada $i\in N$) y su relación es:

$$Sat(\Phi\cup\{\alpha^{\bullet}\}) <=> Sat(\Phi\cup\{\alpha^{\bullet}\}\cup\{\alpha_i^{\bullet} \;/\; i\in N\})$$

$\alpha^{\bullet}$	$\alpha_i^{\bullet}$
$\neg(\mu X.G)(\vec{t})\ni t$	$\neg(\mu X.G)^{(1)}(\vec{t})\ni t$

(6) Fórmulas ω-disyuntivas u ω-beta: las denotamos por $\beta^{\bullet}$ y a sus constituyentes por $\beta_i^{\bullet}$. Tienen una cantidad numerable de constituyentes (para cada $i\in N$) y su relación es:

$$Sat(\Phi\cup\{\beta^{\bullet}\}) <=> Sat(\Phi\cup\{\beta,\beta_i^{\bullet}\}) \text{ para algun } i\in N$$

$\beta^{\bullet}$	$\beta_i^{\bullet}$
$(\mu X.G)(\vec{t})\ni t$	$(\mu X.G)^{(1)}(\vec{t})\ni t$

En las dos últimas categorías, $(\mu X.G)^{(i)}$ es la i-ésima aproximación sintáctica de $(\mu X.G)$, definidas en el capítulo 2 para LRFs. Además, estas dos categorías fuerzan a que los tableaux para LRFs sean árboles ω-ramificados, que es la diferencia esencial con los de primer orden.

En primer lugar vamos a demostrar que efectivamente esta clasificación es una partición que verifica las condiciones especificadas y que existe una función de complejidad de forma que las constituyentes de una fórmula sean más simples que ella.

1- LEMA

Toda $\bar{\Sigma}$-fórmula pertenece exactamente a una de las siete categorías anteriores. Además, esta clasificación verifica las condiciones de satisfactibilidad especificadas.

Demostración

Para demostrar que las siete categorías anteriores son una partición de las fórmulas de LRFs, basta tener en cuenta la condición exigida para los cuatro últimos casos de la categoría alfa.

Para demostrar que cada categoría verifica la relación dada para una fórmula y sus constituyentes, demostramos para una $\bar{\Sigma}$-estructura $\mathfrak{D}$ y un contexto Δ y una valoración δ sobre $\mathfrak{D}$:

(i) Para las fórmulas alfa: $\mathfrak{D}\models\alpha(\Delta,\delta)$ <=> $\mathfrak{D}\models\alpha_1\wedge\alpha_2(\Delta,\delta)$

(ii) Para las fórmulas beta: $\mathfrak{D}\models\beta(\Delta,\delta)$ <=> $\mathfrak{D}\models\beta_1\vee\beta_2(\Delta,\delta)$

(iii) Para las fórmulas gamma: $\mathfrak{D}\models\gamma(\Delta,\delta)$ => $\mathfrak{D}\models\gamma(t)(\Delta,\delta)$ para $t\in TER_{\bar{\Sigma}}$

(iv) Para las fórmulas eta: $\mathfrak{D}\models\eta(\Delta,\delta)$ <=> $\mathfrak{D}\models\eta(c)(\Delta,\delta[x/c])$ para $x\in D$ y $c\in C$ que no aparece en η

(v) Para las fórmulas ω-alfa: $\mathfrak{D}\models\alpha^{\bullet}(\Delta,\delta)$ <=> $\mathfrak{D}\models\alpha_i^{\bullet}(\Delta,\delta)$ para todo $i\in N$

(vi) Para las fórmulas ω-beta: $\mathfrak{D}\models\beta^{\bullet}(\Delta,\delta)$ <=> $\mathfrak{D}\models\beta_1^{\bullet}(\Delta,\delta)$ para algún $i\in N$

Lo cual es inmediato para (i)-(iv) por la semántica dada para LRFs y para (v) y (vi) por el teorema 21 de la sección 2.3, que afirma que $(\mu X.G)$ es equivalente a la unión de sus aproximaciones sintáctica $\bigsqcup_{i\in\mathbb{N}}(\mu X.G)^{(i)}$. ■

2- LEMA

Existe una función de complejidad $|.|$ para las fórmulas de LRF con la propiedad de que para toda fórmula φ, $|\varphi|<|\varphi_i|$ para todo constituyente φ_i de φ.

Demostración

En efecto, definimos la función $|.|$ para las fórmulas, las expresiones y las expresiones funcionales de forma mutuamente recursiva como sigue:

(i) Para $\varphi\in$FFOR:
$|\varphi| = 0$ para toda fórmula φ básica
$|\neg\varphi| = |\varphi| + 1$ para toda fórmula φ que no sea atómica
$|\psi_1\vee\psi_2| = \text{maximo}\{|\psi_1|, |\psi_2|\} + 1$
$|\exists x\psi| = |\psi| + 1$
$|M\ni t| = |M| + 2$ siempre que $M\ni t$ no sea una fórmula atómica.
$|P(\vec{N})| = \text{maximo}\{|N_i| \;/\; 1\le i\le n\} + (6 + n)$ siempre que para algún $1\le j\le n$ $N_j\notin$TER, es decir, siempre que $P(\vec{N})$ no sea una fórmula atómica.
(ii) Para $M\in$EXP:
$|t| = 0$ para todo $t\in$TER
$|N_1\cup N_2| = \text{maximo}\{|N_1|, |N_2|\} + 1$
$|\varphi\rightarrow N| = \text{maximo}\{|\varphi|, |N|\} + 1$
$|\varepsilon x\varphi| = |\varphi| + 1$
$|F(\vec{t})| = |F|$
$|F(\vec{N})| = \text{maximo}\{|F|, |N_i| \;/\; 1\le i\le n\} + (6 + n)$ siempre que para algún $1\le j\le n$ $N_j\notin$TER
(iii) Para $F\in$FEXP:
$|f| = 0$
$|X| = 0$
$|\lambda\vec{x}.M| = |M| + 1$
$|\mu X.G| = \omega + |G|$

Puede comprobarse que esta función verifica la condición pedida sin más que ir comprobando cada caso. Veamos algunos:

Para las fórmulas del tipo $P(\vec{N})$ (no atómica):
$|P(\vec{N})| = \text{maximo}\{|N_i| \ / \ 1 \le i \le n\} + (6 + n)$, y su constituyente
$|\exists \vec{x} \bigwedge_{1 \le i \le n} N_i \exists x_i \wedge P(\vec{x})| = |\bigwedge_{1 \le i \le n} N_i \exists x_i \wedge P(\vec{x})| + 1$, que formalmente es
$= \text{maximo}\{\neg(\bigvee_{1 \le i \le n} \neg N_i \exists x_i \vee \neg P(\vec{x}))\} = \text{maximo}\{|N_i| \ / \ 1 \le i \le n\} + (5 + n)$.

Para las fórmulas del tipo gamma y eta:
Basta con demostrar para toda φ, M y F: que para toda x y término t, $|\varphi[t/x]| = |\varphi|$, $|M[t/x]| = |M|$ y $|F[t/x]| = |F|$, lo que puede demostrarse de forma sencilla, por inducción simultánea sobre la estructura de φ, M y F.

Para las fórmulas del tipo α^* y β^*:
Se verifica puesto que en las aproximaciones sintácticas $(\mu X.G)^{(1)}$ no aparece el operador μ, y por tanto su complejidad es finita, segun la definición de $|.|$. Hubiese sido suficiente definir $|\mu X.G| = \omega$, pero la hemos definido de forma que esta función de complejidad pueda usarse con otros propositos.

En los demás casos se procede de forma similar al caso $P(\vec{N})$. ■

A diferencia con los tableaux para la lógica de primer orden clásica, un tableau para $\Phi \subseteq FFOR_{\Sigma}$ (a lo más numerable) es un árbol ω-ramificado con los nodos etiquetados por conjuntos (a lo más numerables) de $\bar{\Sigma}$-fórmulas obtenido segun la definición 6 de más abajo, para la que necesitamos las siguientes definiciones previas.

3- DEFINICION *(Rama Finita, Cerrada, Profundidad)*

Sea Σ un árbol con los nodos etiquetados por conjuntos de fórmulas:

(a) Una rama R de Σ es finita si tiene un número finito de nodos.

(b) Un conjunto de fórmulas L es cerrado si existe una fórmula φ tal que $\{\varphi, \neg\varphi\} \subseteq L$ o $ff \in L$. En otro caso se dice abierto.

(c) Una rama R de Σ es cerrada si $L = \cup\{L \ / \ L$ es una etiqueta de un nodo de R$\}$ es cerrado. En otro caso es abierta.

(d) Si Σ tiene todas sus ramas finitas, definimos la profundidad ρ de Σ por inducción sobre su estructura, como sigue:

Si Σ esta formado por un solo nodo, $\rho(\Sigma)=0$

Si Σ_i, $i \in I$ son los subárboles de Σ, $\rho(\Sigma)$ es el supremo de los ordinales $\rho(\Sigma_i)+1$ $(i \in I)$. ■

Obsérvese, que el hecho de que un árbol Σ tenga todas sus ramas finitas no quiere decir que Σ sea de profundidad finita (es decir, que $\rho(\Sigma)$ sea un ordinal finito) puesto que Σ puede ser ω-ramificado.

4- DEFINICION

Sea Φ un conjunto de fórmulas.

(a) Un término t es adecuado a Φ si todos los símbolos de función (y por tanto los de constante) que usa t aparecen en Φ y las variables de individuo libres de t aparecen libres en alguna fórmula de Φ.

(b) Los axiomas de la igualdad son:

$\forall x(x=x)$

$\forall x \forall y(x=y \rightarrow y=x)$

$\forall x \forall y \forall z(x=y \wedge y=z \rightarrow x=z)$

$\forall \vec{x} \forall \vec{y} \forall z(f(\vec{x})=z \wedge \vec{x}=\vec{y} \rightarrow f(\vec{y})=z)$

$\forall \vec{x} \forall \vec{y}(P(\vec{x}) \wedge \vec{x}=\vec{y} \rightarrow P(\vec{y}))$

$\forall \vec{x} \forall \vec{y} \forall z \forall v(X(\vec{x}) \ni z \wedge \vec{x}=\vec{y} \wedge z=v \rightarrow X(\vec{y}) \ni v)$

Siendo f un símbolo de función de aridad >0, P un símbolo de predicado de aridad >0 y $X \in FVAR$ de aridad ≥ 0.

Un axioma de la igualdad σ es adecuado a Φ si todos los símbolos de función y predicado que usa σ (y por tanto ambos de aridad > 0) aparecen en Φ y las variables funcionales libres de σ aparecen en alguna fórmula de Φ. ■

5-DEFINICION *(Extensión c.r.a Φ para Tableaux)*

Dados un conjunto de fórmulas Φ y dos árboles (ω-ramificados) con los nodos etiquetados por conjuntos de fórmulas Σ y Σ'con todas sus ramas finitas, decimos que Σ' es una extensión con respecto a Φ de Σ si para algún conjunto A de hojas abiertas de Σ, Σ' se ha obtenido al extender en Σ, de forma simultánea, cada una de las hojas de A de acuerdo con las siguientes reglas de extensión c.r.a Φ de hojas.

Sea H una hoja etiquetada por el conjunto de fórmulas L:

(1) HI-extensión: Si $\Phi_0 \subseteq \Phi$, añadir a H un hijo etiquetado por $L \cup \Phi_0$.

(2) EQ-extensión: Si $A\sigma$ es un conjunto de axiomas de la igualdad adecuados a Φ, añadir a H un hijo etiquetado por $L \cup \{A\sigma\}$.

(3) α-extensión: Si $\alpha \in L$, añadir a H un hijo etiquetado por $L \cup \{\alpha_1, \alpha_2\}$.

(4) β-extensión: Si $\beta \in L$, añadir a H dos hijos etiquetados por $L \cup \{\beta_1\}$ y $L \cup \{\beta_2\}$ respectivamente.

(5) γ-extensión: Si $\gamma \in L$, añadir a H un hijo etiquetado por $L \cup \{\gamma(t)\}$, donde t es un término adecuado a L o bien t es el primer símbolo de C si no hay términos adecuados a L.

(6) η-extensión: Si $\eta \in L$, añadir a H un hijo etiquetado por $L \cup \{\eta(c)\}$, siendo $c \in C$ una constante nueva; es decir, que no aparece en L.

(7) α^*-extensión: Si $\alpha^* \in L$, añadir a H un hijo etiquetado por $L \cup \{\alpha_i^* / i \in N\}$

(8) β^*-extensión: Si $\beta^* \in L$, añadir a H ω hijos etiquetados por $L \cup \{\beta_i^*\}$ $(i \in N)$ respectivamente.

6- DEFINICION *(Tableau para Φ)*

Sea Φ un conjunto de Σ-fórmulas. Σ es un tableau para Φ si es el límite $\bigsqcup_{i<\xi}\Sigma_i$ $(\xi\leq\omega)$ de una sucesión de tableaux $\Sigma_0,\Sigma_1,\ldots,\Sigma_i,\Sigma_{i+1},\ldots$ tal que Σ_0 tienen un único nodo etiquetado por un subconjunto de Φ y Σ_{i+1} es una extensión de Σ_i c.r.a Φ. ■

Obsérvese que para cada $i<\xi$, Σ_i tiene todas sus ramas finitas. Además, una rama finita es abierta (o cerrada) si lo es su hoja.

7- DEFINICION *(Tableau Cerrado)*

Un tableau Σ es cerrado si toda rama de Σ es finita y cerrada. En otro caso Σ es abierto. ■

8- LEMA

Sea Φ un conjunto de Σ-fórmulas y Σ un tableau para Φ con todas sus ramas finitas. Si Φ es satisfactible, entonces Σ tiene una hoja satisfactible.

Demostración

Sea $\Phi_0\subseteq\Phi$ el conjunto de fórmulas de Φ usado en Σ. Para toda Σ-estructura $\mathfrak{D}$ y Δ y δ contexto y valoración sobre $\mathfrak{D}$. Demostramos que $\mathfrak{D}\models\Phi_0(\Delta,\delta)$ => para alguna hoja H de Σ con etiqueta L y una $\bar{\Sigma}$-estructura $\hat{\mathfrak{D}}$ extensión de $\mathfrak{D}$, $\hat{\mathfrak{D}}\models L(\Delta,\delta)$.

En efecto, por definición de tableau $\Sigma = \bigsqcup_{i<\xi}\Sigma_i$ con $\xi\leq\omega$, supongamos la hipótesis y demostremos por inducción transfinita sobre i, que en Σ_i existe una rama R_i, extensión de R_{i-1} o igual a R_{i-1} (para $i>0$), cuya hoja H_i de etiqueta L_i se satisface en una estructura $\mathfrak{D}_i$ extensión de $\mathfrak{D}$ bajo Δ, δ.

Para i=0: H_0 es el único nodo de Σ_0, R_0 consiste solo de H_0 y $\mathfrak{D}_0$ es $\mathfrak{D}$, ya que $L_0\subseteq\Phi_0$.

Para $i>0$: Si la hoja H_i de la rama R_i de Σ_i no se extiende para obtener Σ_{i+1}, entonces H_i es una hoja de Σ_{i+1} que satisface las condiciones requeridas por hipótesis de inducción. Es decir, en este caso R_{i+1} es R_i y D_{i+1} es D_i. En otro caso, supongamos que a H_i se le ha aplicado una extensión, y distingamos casos:

η-extensión: entonces $L_{i+1}=L_i\cup\{\eta(c)\}$ (H_{i+1} es el hijo de la extensión de H_i) para $c\in C$ que no aparece en L_i. Sea D_{i+1} la extensión de D_i para el nuevo símbolo de función c, que interpreta c como el valor $x\in D$ que satisface η en D_i bajo Δ,δ (ya que $\eta\in L_i$).

β^*-extensión: entonces $L_{i+1}=L_i\cup\{\beta^*_k\}$ para un $k\in N$ tal que $D_i\models\beta^*_k(\Delta,\delta)$, que existe por el lema 1. Y por tanto H_{i+1} es el k-ésimo hijo de la extensión de H_i, y D_{i+1} es D_i.

Para los demás casos se procede de forma análoga, la estructura solo se aumenta por η-extensiones. Además, como todas las ramas de Σ son finitas, existe un $i\in N$ tal que para todo $j>i$, $R_j=R_i$. Luego tomando $H=H_i$ y $\hat{D}$ una extensión arbitraria de D_i a $\overline{\Sigma}$, obtenemos lo que buscabamos. ■

9- DEFINICION *(Conjunto de Hintikka)*

Un conjunto de $\overline{\Sigma}$-fórmulas H es un conjunto de Hintikka si satisface las siguientes condiciones:

(1) H es un conjunto abierto.
(2) Todo axioma de la igualdad σ adecuado a H pertenece a H.
(3) Para toda fórmula conjuntiva α: $\alpha\in H \Rightarrow \alpha_i\in H$ para todo $i=1,2$.
(4) Para toda fórmula disyuntiva β: $\beta\in H \Rightarrow \beta_j\in H$ para algún $j=1,2$.
(5) Para toda fórmula universal γ: $\gamma\in H \Rightarrow \gamma(t)\in H$ para todo $\overline{\Sigma}$-término t adecuado a H.
(6) Para toda fórmula existencial η: $\eta\in H \Rightarrow \eta(c)\in H$ para algún símbolo de constante nuevo $c\in C$.
(7) Para toda fórmula ω-conjuntiva α^*: $\alpha^*\in H \Rightarrow \alpha^*_i\in H$ para todo $i<\omega$.
(8) Para toda fórmula ω-disyuntiva β^*: $\beta^*\in H \Rightarrow \beta^*_j\in H$ para algún $j<\omega$. ■

10- TEOREMA *(Satisfactibilidad de los Conjuntos de Hintikka)*

Si H es un conjunto de Hintikka, entonces H es satisfactible.

Demostración

Definimos una relación de equivalencia sobre el conjunto de los $\overline{\Sigma}$-términos $TER_{\overline{\Sigma}}$ de la forma:

$$t \sim t' \text{ sii } t=t' \in H$$

Usando los axiomas de la igualdad se puede probar que esta es una relación de equivalencia. Denotamos por [t] a la clase de t, para todo $\overline{\Sigma}$-témino t. Definimos la $\overline{\Sigma}$-estructura $\mathfrak{D}$ de dominio $D=\{[t] \ / \ t \in TER_{\overline{\Sigma}}\}$ y las siguientes interpretaciones, contexto Δ y valoración δ sobre $\mathfrak{D}$

$$f^{\mathfrak{D}}([t_1],\ldots,[t_n]) = [f(t_1,\ldots,t_n)]$$
$$c^{\mathfrak{D}} = [c]$$

$$p^{\mathfrak{D}}([t_1],\ldots,[t_n]) = \begin{cases} \text{true} & \text{si } p(t_1,\ldots,t_n) \in H \\ \text{false} & \text{si } \neg p(t_1,\ldots,t_n) \in H \\ \text{arbitrario en otro caso} & \end{cases}$$

$$\Delta(X)([t_1],\ldots,[t_n]) = \{[t] \ / \ (X(t_1,\ldots,t_n) \ni t) \in H\}$$
$$\delta(x) = [x]$$

Otra vez usando la condición (2) se demuestra que estas interpretaciones están bien definidas.

Se comprueba por inducción sobre la estructura de t:

(0) $t^{\mathfrak{D}}(\delta) = [t]$

Y por inducción simultanea sobre $|.|$ de φ, M y F:

(1) $\varphi \in H \Rightarrow \varphi^{\mathfrak{D}}(\Delta,\delta)=\text{true}$

(2) $M^{\mathfrak{D}}(\Delta,\delta) = \{[t] \ / \ (M \ni t) \in H\}$

(3) $F^{\mathfrak{D}}(\Delta,\delta)([t_1],\ldots,[t_n]) = \{[t] \ / \ (F(t_1,\ldots,t_n) \ni t) \in H\}$ ■

11- DEFINICION *(Tableau completo)*

Un tableau $\mathcal{T}$ para un conjunto de fórmulas Φ es completo si toda rama abierta de $\mathcal{T}$ es completa; donde una rama abierta R es completa si el conjunto de fórmulas $H = \cup \{L / L$ es una etiqueta de un nodo de R$\}$ verifica $\Phi \subseteq H$ y H es un conjunto de Hintikka. ■

12- LEMA *(Existencia de tableau completo, tableau Canónico)*

Todo conjunto de Σ-fórmulas Φ tiene un tableau completo.

Demostración

En efecto, dado Φ vamos a conàtruir dicho tableau, al que llamamos canónico y denotamos por $\mathcal{T}_\Phi$. Para esto, consideremos una enumeración del siguiente conjunto de fórmulas

$\{\alpha / \alpha$ es una fórmula conjuntiva$\} \cup$
$\{\beta / \beta$ es una fórmula disyuntiva$\} \cup$
$\{\langle\gamma, \gamma(t)\rangle / \gamma$ es una fórmula universal, $t \in TER_\Sigma\} \cup$
$\{\eta / \eta$ es una fórmula existencial$\} \cup$
$\{\alpha^\bullet / \alpha^\bullet$ es una fórmula ω-conjuntiva$\} \cup$
$\{\beta^\bullet / \beta$ es una fórmula ω-disyuntiva$\}$

Decimos que una fórmula (o par de fórmulas, del conjunto anterior) φ es relevante para un conjunto de fórmulas L si verifica la condición correspondiente a su categoría, de las siguientes condiciones:

- α es relevante para L si $\alpha \in L$ y $\alpha_i \notin L$ para algún $i=1,2$.
- β es relevante para L si $\beta \in L$ y $\beta_i \notin L$ para todo $i=1,2$.
- $\langle\gamma, \gamma(t)\rangle$ es relevante para L si $\gamma \in L$ y $\gamma(t) \notin L$.
- η es relevante para L si $\eta \in L$ y $\eta(c) \notin L$ para todo símbolo de constante $c \in C$ nuevo.
- $\alpha^\bullet$ es relevante para L si $\alpha^\bullet \in L$ y $\alpha^\bullet_i \notin L$ para algún $i \in N$.
- $\beta^\bullet$ es relevante para L si $\beta^\bullet \in L$ y $\beta^\bullet_i \notin L$ para todo $i \in N$.

El tableau canónico para Φ se construye mediante el siguiente procedimiento:

1.- $i:=0$, $\Upsilon_0 := \bullet\Phi$

2.- $i:=1$, $\Upsilon_1 := \begin{array}{l}\bullet\Phi \\ \mid \\ \bullet\Phi\cup E_A\end{array}$

donde $E_A=\{\sigma \mid \sigma$ es un axioma de la igualdad adecuado a $\Phi\}$

3.- Si Υ_i es completo, acabar devolviendo Υ_i.

4.- En otro caso, sea $A = \{H \mid H$ es una hoja abierta de $\Upsilon_i\}$, para cada $H\in A$ tomar si existe la menor fórmula (o par de fórmulas) relevante para el conjunto de fórmulas L que etiqueta a H y aplicarle de forma simultánea a todas estas hojas la correspondiente regla de extensión de hojas c.r.a Φ.

5.- Sea Υ_{i+1} el árbol resultante de la extensión c.r.a Φ de Υ por 4 e $i:=i+1$, y volver a 3.

Definimos $\Upsilon_\Phi := \begin{cases} \bigsqcup_{i\leq j} \Upsilon_i & \text{si el algoritmo para en la vuelta } j \\ \bigsqcup_{i<\omega} \Upsilon_i & \text{en otro caso} \end{cases}$

Comprobemos que Υ_Φ es un tableau completo para Φ. Que Υ_Φ es un tableau para Φ es evidente. Si el algoritmo para en la vuelta i, entonces Υ_Φ es completo por la condición de parada del algoritmo. Supongamos por tanto que $\Upsilon_\Phi = \bigsqcup_{i<\omega}\Upsilon_i$ y demostremos para toda rama abierta R de Υ_Φ que el conjunto

$$L=\cup\{L \mid L \text{ es una etiqueta de un nodo de } R\}$$

es un conjunto de Hintikka, puesto que $\Phi\subseteq L$ se verifica por el paso 0 del algoritmo. Las condiciones (1) y (2) de conjunto de Hintikka se verifican por el paso 2 del algoritmo y por ser R abierta respectivamente. Para las condiciones (3)-(8) razonamos como sigue: sea $\varphi\in L$ una fórmula no básica, entonces existe $i\in N$ tal que φ aparece por primera vez en la rama R_i, siendo R_i la parte de la rama R correspondiente a Υ_i. Y sea $k\in N$ la posición que ocupa φ en la numeración considerada en la construcción de Υ_Φ. Entonces, en R_{i+k}, la parte de R correspondiente a Υ_{i+k}, estan las constituyentes correspondientes a φ. ■

13- TEOREMA *(Corrección y Completitud del Método de los Tableaux)*

Sea Φ un conjunto de fórmulas:

Φ no es satisfactible <=> Φ tiene un tableau cerrado.

Demostración

CORRECCION: Sea $\mathcal{T}$ un tableau cerrado para Φ. Por definición, todas las ramas de $\mathcal{T}$ son finitas y cerradas. Si Φ fuese satisfactible, por el lema 8 $\mathcal{T}$ tendría una rama satisfactible, lo que contradice que todas sus ramas sean cerradas.

COMPLETITUD: Si Φ no tiene tableau cerrado, $\mathcal{T}_\Phi$ no es cerrado, y existen dos posibilidades:

(a) $\mathcal{T}_\Phi$ tiene todas sus ramas finitas: Entonces, como no es cerrado, tiene una rama finita abierta.
(b) $\mathcal{T}_\Phi$ tiene una rama infinita: Entonces esta rama es abierta, puesto que en el algoritmo de construcción del tableau canónico, solo se extienden las ramas abiertas.

En ambos casos, podemos concluir que Φ es satisfactible, puesto que si $\mathcal{T}_\Phi$ tiene una rama R abierta, por ser $\mathcal{T}_\Phi$ completo el conjunto $\mathcal{L}$ de las etiquetas de R es de Hintikka. Es decir, si $R=\{nodo_0,\ldots,nodo_i,\ldots\}$ es una rama abierta de $\mathcal{T}_\Phi$ y L_i la etiqueta del $nodo_i$, entonces $\mathcal{L}=\bigcup_{i<\omega} L_i$ es un conjunto de Hintikka que incluye a Φ. ■

14- EJEMPLO

Sea Σ una signatura con un símbolo de constante 0 y dos funciones monádicas pred y succ. Definimos la función constante de un argumento Zero, por la siguiente expresión funcional:

$$\text{Zero} \equiv \mu X.\ (\lambda x.\ (x=0 \rightarrow 0 \cup \neg x=0 \rightarrow X(x-1))$$

Sus aproximaciones sintácticas son de la forma:

$Zero^{(0)} \equiv \lambda x.\ (\neg\ x{=}x \rightarrow x) = \Omega^{(1)}$

$Zero^{(i+1)} \equiv \lambda x.\ (x{=}0 \rightarrow 0 \cup$
$\neg x{=}0 \rightarrow Zero^{(i)}(x-1))$

Vamos a demostrar las consecuencias lógicas:

(a) $\neg(1{=}0) \models \neg Zero(0) \ni 1$
(b) $\neg(1{=}0) \models Zero(1) \ni 0$

Para esto, comprobamos que los conjuntos:

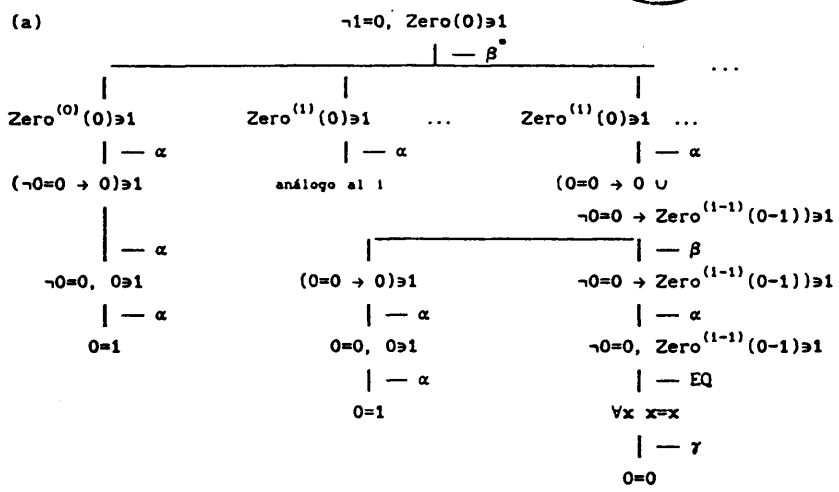
(a) $\{\neg(1{=}0),\ Zero(0) \ni 1\}$
(b) $\{\neg(1{=}0),\ \neg Zero(1) \ni 0\}$

son insatisfactibles, obteniendo un tableau cerrado para cada caso.

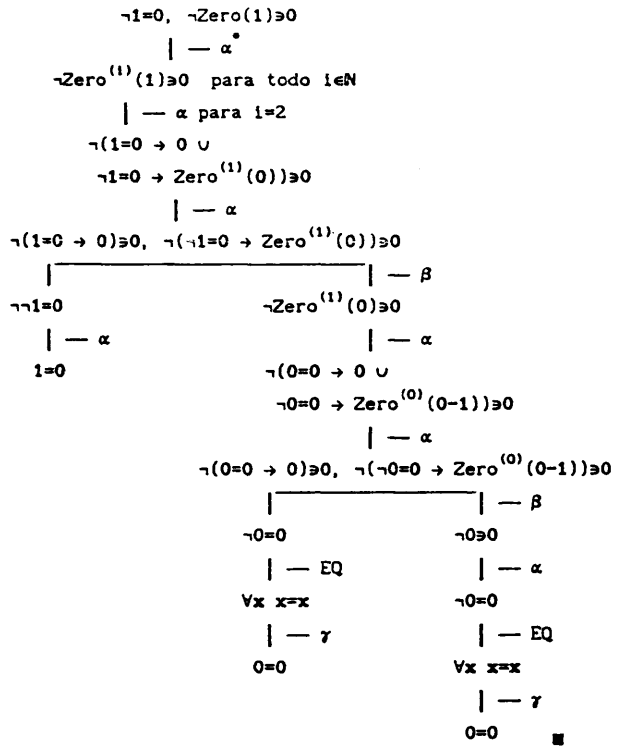
Para poder asegurar en (a) que Zero(0) no produce 1, es necesario comprobar que 1 no es producido por ninguna de las aproximaciones sintácticas de Zero(0).

Sin embargo, para asegurar que 0 es producido por Zero(1) es suficiente iterar dos veces la definición de Zero.

En la construcción de los tableaux, solo vamos a poner en cada nodo las fórmulas que se añaden al realizar la correspondiente extensión, de forma que la etiqueta real de un nodo es el conjunto de fórmulas que aparecen en los nodos desde la raíz hasta él. Además indicaremos la extensión que se ha realizado etiquetando los árcos con ella.



(b)


¬1=0, ¬Zero(1)∋0
| — α*
¬Zero(1)(1)∋0 para todo i∈N
| — α para i=2
¬(1=0 → 0 ∪
¬1=0 → Zero(1)(0))∋0
| — α
¬(1=0 → 0)∋0, ¬(¬1=0 → Zero(1)(0))∋0
| — β
¬¬1=0
| — α
1=0
¬Zero(1)(0)∋0
| — α
¬(0=0 → 0 ∪
¬0=0 → Zero(0)(0-1))∋0
| — α
¬(0=0 → 0)∋0, ¬(¬0=0 → Zero(0)(0-1))∋0
| — β
¬0=0
| — EQ
∀x x=x
| — γ
0=0
¬0∋0
| — α
¬0=0
| — EQ
∀x x=x
| — γ
0=0

■

CALCULO INFINITARIO

Como se ha dicho, el método de los tableaux, no solo es un método para obtener la satisfactibilidad de fórmulas, si no que es una herramienta para las demostraciones de completitud de cálculos. Antes de presentar el cálculo, establecemos condiciones suficientes que debe verificar una relación de deducibilidad formal para ser completa para LRFs.

El siguiente lema será usado en la demostración de las condiciones suficientes de completitud, ya que en ella se procede por inducción sobre la profundidad de tableaux, y la profundidad de un árbol está definida por inducción sobre la estructura del árbol.

15- LEMA

Sea Φ un conjunto de fórmulas y Σ un tableau para Φ con todas sus ramas finitas, y con etiqueta L en su raíz. Entonces Σ tiene alguna de las siguientes estructuras:

(i) Σ tiene un solo nodo etiquetado por $\Phi_0 \subseteq \Phi$.

(ii) Σ tiene exactamente un subárbol Σ_1 con etiqueta L_1 en su raíz, de forma que Σ_1 es un tableau para $\Phi \cup L_1$ y además en L se realizó una HI, EQ, α, γ, η o α^*-extensión c.r.a Φ.

(iii) Σ tiene exactamente dos subárboles Σ_1 y Σ_2 con etiqueta L_1 y L_2 en su raíz respectivamente, de forma que Σ_1 es un tableau para $\Phi \cup L_1$ y Σ_2 para $\Phi \cup L_2$ y además en L se realizó una β-extensión c.r.a Φ.

(iv) Σ tiene exactamente ω subárboles Σ_i ($i \in N$) con etiqueta L_i en su raíz, de forma que Σ_i es un tableau para $\Phi \cup L_i$ y además en L se realizó una β^*-extensión c.r.a Φ.

Demostración

El hecho de que si $\mathcal{I}$ no consiste de un solo nodo entonces a L se le aplicó una de las posibles extensiones c.r.a Φ se verifica por definición de tableau para Φ.

Para comprobar que un subárbol $\mathcal{I}_1$ de $\mathcal{I}$ es un tableau para $\Phi \cup L_1$, veamos que $\mathcal{I}_1$ verifica la definición de tableau para $\Phi \cup L_1$. Por hipótesis, $\mathcal{I}$ es $\bigcup_{k<\xi} \mathcal{I}_k$ $(\xi \leq \omega)$, tal que $\mathcal{I}_0$ verifica (1) del lema y $\mathcal{I}_{k+1}$ es una extensión c.r.a Φ de $\mathcal{I}_k$. Entonces $\mathcal{I}_1$ es $\bigcup_{k<\xi} \hat{\mathcal{I}}_k$ $(\xi \leq \omega)$, tal que $\hat{\mathcal{I}}_0 = \Phi \cup L_1$ y $\hat{\mathcal{I}}_{k+1}$ se obtiene a partir de $\hat{\mathcal{I}}_k$, aplicando a $\hat{\mathcal{I}}_k$ la misma extensión que se aplicó a $\mathcal{I}_k$ pero restringida a las hojas de $\hat{\mathcal{I}}_k$. Además, toda extensión con respecto a un conjunto de fórmulas lo es también con respecto a un conjunto mayor. Por lo que se puede concluir que $\mathcal{I}_1$ es un tableau para $\Phi \cup L_1$. ■

16- TEOREMA *(Condiciones de Completitud)*

Todo cálculo $\mathcal{C}$ cuya relación de deducibilidad $\vdash_{\mathcal{C}}$ (abreviadamente $\vdash$) satisface las siguientes condiciones es completo para LRFs.

(1) Si $\Phi \vdash \psi$ entonces $\Phi \cup \Phi' \vdash \psi$
(2) Si $\Phi \cup \{\sigma\} \vdash \psi$ entonces $\Phi \vdash \psi$ para cualquier axioma de la igualdad σ
(3) Para toda fórmula α: si $\Phi \cup \{\alpha_1, \alpha_2\} \vdash \psi$ entonces $\Phi \cup \{\alpha\} \vdash \psi$
(4) Para toda fórmula β: si $\Phi \cup \{\beta_1\} \vdash \psi$ y $\Phi \cup \{\beta_2\} \vdash \psi$ entonces $\Phi \cup \{\beta\} \vdash \psi$
(5) Para toda fórmula γ: si $\Phi \cup \{\gamma(t)\} \vdash \psi$ entonces $\Phi \cup \{\gamma\} \vdash \psi$ para todo término t
(6) Para toda fórmula η: si $\Phi \cup \{\eta(c)\} \vdash \psi$ entonces $\Phi \cup \{\eta\} \vdash \psi$ para c una constante nueva que no aparece ni en Φ ni en ψ ni en η
(7) Para toda fórmula α^*: si $\Phi \cup \{\alpha_i^* \ / \ i \in N\} \vdash \psi$ entonces $\Phi \cup \{\alpha^*\} \vdash \psi$
(8) Para toda fórmula β^*: si $\Phi \cup \{\beta_i^*\} \vdash \psi$ para todo $i \in N$ entonces $\Phi \cup \{\beta^*\} \vdash \psi$
(9) $\{\varphi, \neg\varphi\} \vdash ff$
(10) Si $\Phi \cup \{\neg\psi\} \vdash ff$ entonces $\Phi \vdash \psi$

Demostración

Se reduce a demostrar para cualquier fórmula φ, que si $\mathcal{I}$ es un tableau de ramas finitas para algún conjunto de fórmulas Φ tal que para toda rama de $\mathcal{I}$, $L \vdash \varphi$ siendo L la etiqueta de la hoja de la rama, entonces $\Phi \vdash \varphi$. Para

esto, procedemos por inducción transfinita sobre la profundidad de Σ, usando el lema anterior:

$\rho(\Sigma)=0$: Entonces Σ tiene un solo nodo y $L \subseteq \Phi$. Basta con aplicar (1).

$\rho(\Sigma)>0$: Entonces Σ tiene I subárboles Σ_i ($i \in I$), cada uno de los cuales es un tableau para $\Phi \cup L_i$ (L_i la etiqueta de su raíz). Además, para toda rama de Σ_i, $L_i \vdash \varphi$ siendo L_i la etiqueta de la hoja de la rama, por construcción; luego aplicando hipótesis de inducción obtenemos que $\Phi \cup L_i \vdash \varphi$ para todo $i \in I$. Distinguiendo casos en función de la extensión c.r.a Φ que se le aplicó a L (raíz de Σ), se concluye $\Phi \vdash \varphi$. Veamos el caso de α-extensión: I es unitario, $L_i = L \cup \{\alpha_1, \alpha_2\}$ y $\alpha \in L$, luego por (3) $\Phi \cup L \vdash \varphi$ y puesto que $L \subseteq \Phi$, $\Phi \vdash \varphi$. Los demás casos son análogos.

Demostremos ahora el teorema: Si $\Phi \vdash \cdot \psi$, entonces $\Phi \cup \{\neg\psi\}$ es insatisfactible y por el teorema anterior tiene un tableau cerrado Σ; es decir, todas sus ramas son finitas y cerradas. Luego por la condición (9), para toda rama de Σ, $L \vdash ff$ donde L es la etiqueta de la hoja de la rama. Y por tanto de $\Phi \cup \{\neg\psi\} \vdash ff$. Por último, por la condición (10), $\Phi \vdash \psi$. ■

El cálculo que presentamos a continuación es un cálculo de secuencias, también se pueden obtener cálculos al estilo Hilbert sin más que añadir a un cálculo de este tipo para la lógica de primer orden clásica los axiomas y reglas propios para la descomposición de un programa en sus partes. Hemos optado por un cálculo de secuencias en primer lugar porque nos parece un método mucho más natural para obtener deducciones que los cálculos de tipo Hilbert, y en segundo lugar porque se adecuan perfectamente al método de los tableaux.

Este cálculo, al que hemos denominado i, consiste de los siguientes axiomas (o secuencias obviamente correctas) y reglas de inferencia que reducen la corrección de una secuencia (conclusión de la regla) a la corrección de otras secuencias (premisas de la regla). Denotamos, en lo que sigue, $\Gamma \vdash \varphi$ a la secuencia de antecedente el conjunto de fórmulas Γ y consecuente la fórmula φ. Para el antecedente, casi nunca usaremos notación de conjunto sino fórmulas separadas por comas.

Axiomas y reglas de primer orden

(SP) $\Gamma \vdash \varphi$ si $\varphi \in \Gamma$

(FS) $\dfrac{\Gamma \vdash \varphi}{\Gamma, \Gamma' \vdash \varphi}$

(RC) $\dfrac{\Gamma \vdash \psi \quad \Gamma, \psi \vdash \varphi}{\Gamma \vdash \varphi}$

(v-C) $\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi}$ $\qquad$ $\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \psi \vee \varphi}$

(v-A) $\dfrac{\Gamma, \varphi \vdash \phi \quad \Gamma, \psi \vdash \phi}{\Gamma, \varphi \vee \psi \vdash \phi}$

(¬-C) $\dfrac{\Gamma, \varphi \vdash ff}{\Gamma \vdash \neg \varphi}$ $\qquad$ (¬-A) $\dfrac{\Gamma, \neg \varphi \vdash \varphi}{\Gamma, \neg \varphi \vdash ff}$

(ffA) $\Gamma, ff \vdash \varphi$ $\qquad$ (DN-A) $\dfrac{\Gamma, \varphi \vdash \psi}{\Gamma, \neg\neg \varphi \vdash \psi}$

(∃-C) $\dfrac{\Gamma \vdash \varphi[t/x]}{\Gamma \vdash \exists x \varphi}$ $\qquad$ (∃-A) $\dfrac{\Gamma, \varphi[c/x] \vdash \psi}{\Gamma, \exists x \varphi \vdash \psi}$ siendo $c \in C$ nueva

(ID) $\Gamma \vdash t=t$ $\qquad$ (SUS) $\dfrac{\Gamma \vdash \varphi[t/x]}{\Gamma, t=t' \vdash \varphi[t'/x]}$

Reglas para la descomposición de las expresiones

(∋-A) $\dfrac{\Gamma,\ t=t' \vdash \varphi}{\Gamma,\ t \ni t' \vdash \varphi}$ $\qquad$ (∋¬A) $\dfrac{\Gamma,\ \neg t=t' \vdash \varphi}{\Gamma,\ \neg t \ni t' \vdash \varphi}$

(PD-A) $$\dfrac{\Gamma,\ \exists y_1 .. \exists y_n (M_1 \ni y_1 \wedge \ldots \wedge M_n \ni y_n \wedge p(y_1, \ldots, y_n)) \vdash \varphi}{\Gamma,\ p(M_1, \ldots, M_n) \vdash \varphi}$$

$$(PD\neg A)\quad \frac{\Gamma,\ \neg\exists y_1..\exists y_n(M_1 \ni y_1 \wedge \ldots \wedge M_n \ni y_n \wedge p(y_1,\ldots,y_n)) \vdash \varphi}{\Gamma,\ \neg p(M_1,\ldots,M_n) \vdash \varphi}$$

donde las variables y_i no aparecen ni en $M_1,\ldots,M_{n-1}$ ni en M_n y son distintas entre si.

$$(\cup\text{-}A)\quad \frac{\Gamma,\ M \ni t \vee N \ni t \vdash \varphi}{\Gamma,\ (M \cup N) \ni t \vdash \varphi} \qquad (\cup\neg A)\quad \frac{\Gamma,\ \neg(M \ni t \vee N \ni t) \vdash \varphi}{\Gamma,\ \neg(M \cup N) \ni t \vdash \varphi}$$

$$(\to\text{-}A)\quad \frac{\Gamma,\ \psi \wedge M \ni t \vdash \varphi}{\Gamma,\ (\psi \to M) \ni t \vdash \varphi} \qquad (\to\neg A)\quad \frac{\Gamma,\ \neg(\psi \wedge M \ni t) \vdash \varphi}{\Gamma,\ \neg(\varphi \to M) \ni t \vdash \varphi}$$

$$(\varepsilon\text{-}A)\quad \frac{\Gamma,\ \psi[t/x] \vdash \varphi}{\Gamma,\ (\varepsilon x \psi) \ni t \vdash \varphi} \qquad (\varepsilon\neg A)\quad \frac{\Gamma,\ \neg\psi[t/x] \vdash \varphi}{\Gamma,\ \neg(\varepsilon x \psi) \ni t \vdash \varphi}$$

$$(AP\text{-}A)\quad \frac{\Gamma,\ \exists y_1..\exists y_n(M_1 \ni y_1 \wedge \ldots \wedge M_n \ni y_n \wedge F(y_1,\ldots,y_n)) \ni t \vdash \varphi}{\Gamma,\ F(M_1,\ldots,M_n) \ni t \vdash \varphi}$$

$$(AP\neg A)\quad \frac{\Gamma,\ \neg\exists y_1..\exists y_n(M_1 \ni y_1 \wedge \ldots \wedge M_n \ni y_n \wedge F(y_1,\ldots,y_n)) \ni t \vdash \varphi}{\Gamma,\ \neg F(M_1,\ldots,M_n) \ni t \vdash \varphi}$$

donde las variables y_i no aparecen ni en $M_1,\ldots,M_n$ ni en t y son distintas entres si.

$$(\lambda\text{-}A)\quad \frac{\Gamma,\ M[t_1/x_1,\ldots,t_n/x_n] \ni t \vdash \varphi}{\Gamma,\ (\lambda x_1,\ldots,x_n.M)(t_1,\ldots,t_n) \ni t \vdash \varphi}$$

$$(\lambda\neg A)\quad \frac{\Gamma,\ \neg M[t_1/x_1,\ldots,t_n/x_n] \ni t \vdash \varphi}{\Gamma,\ \neg(\lambda x_1,\ldots,x_n.M)(t_1,\ldots,t_n) \ni t \vdash \varphi}$$

$$(\mu\text{-A}) \quad \frac{\Gamma, (\mu X.G)^{(i)} \vdash \varphi \quad \text{para todo } i \in \mathbb{N}}{\Gamma, (\mu X.G) \vdash \varphi}$$

$$(\neg\mu\text{C}) \quad \frac{\Gamma, \{\neg(\mu X.G)^{(i)} / \ i \in \mathbb{N}\} \vdash \varphi}{\Gamma, \neg(\mu X.G) \vdash \varphi}$$

■

En las dos últimas reglas, $(\mu X.G)^{(i)}$ denota la i-ésima aproximación sintáctica de $\mu X.G$ definida en la definicón 18 de la sección 2.3.

Denotamos por $\vdash_i$ o simplemente por $\vdash$, a la relación de deducibilidad formal en este cálculo. Como en la lógica infinitaria clásica $L\omega_1\omega$, las demostraciones formales pueden ser infinitas (cfr. [Kei 71]), más formalmente:

9- DEFINICION

(a) Un árbol ω-ramificado con todas sus ramas finitas y con los nodos etiquetados por secuencias de la forma $\Gamma \vdash \psi$, siendo Γ a lo más numerable, es un árbol de deducción si verifica (i) o (ii):

(i) es un árbol con un solo nodo cuya etiqueta es un axioma del cálculo.

(ii) Su raíz, etiquetada por L, tiene ξ subárboles, $\xi \leq \omega$, con raices etiquetadas por L_i $(i \leq \xi)$ respectivamente, todos ellos árboles de deducción, de forma que existe una regla en el cálculo cuya conclusión es L y sus premisas son las secuencias L_i $(i \leq \xi)$.

(b) Dados un conjunto de fórmulas Φ (a lo más numerable) y una fórmula φ, φ es demostrable a partir de Φ en el cálculo i, denotado por $\Phi \vdash_i \varphi$ o simplemente por $\Phi \vdash \varphi$, si existe un árbol de deducción para $\Gamma \vdash \varphi$ con $\Gamma \subseteq \Phi$, es decir un árbol de deducción cuya raíz está etiquetada por $\Gamma \vdash \varphi$. ■

Las siguientes reglas (o axiomas) son derivadas:

$$(\rightarrow\text{+C})\ \frac{\Gamma\vdash\varphi\rightarrow\psi}{\Gamma,\varphi\vdash\psi} \qquad (\rightarrow\text{-C})\ \frac{\Gamma,\varphi\vdash\psi}{\Gamma\vdash\varphi\rightarrow\psi}$$

$$(\text{MP})\ \frac{\Gamma\vdash\varphi\rightarrow\psi \quad \Gamma\vdash\varphi}{\Gamma\vdash\psi} \qquad (\text{CD})\ \Gamma,\varphi,\neg\varphi\vdash\psi$$

$$(\text{RA})\ \frac{\Gamma,\neg\varphi\vdash ff}{\Gamma\vdash\varphi} \qquad (\text{DN-C})\ \frac{\Gamma\vdash\varphi}{\Gamma\vdash\neg\neg\varphi}$$

$$(\wedge\text{-A})\ \frac{\Gamma,\varphi,\psi\vdash\phi}{\Gamma,\varphi\wedge\psi\vdash\phi} \qquad (\wedge\text{-C})\ \frac{\Gamma\vdash\varphi \quad \Gamma\vdash\psi}{\Gamma\vdash\varphi\wedge\psi}$$

$$(\forall\text{-A})\ \frac{\Gamma,\varphi[t/x]\vdash\psi}{\Gamma,\forall x\varphi\vdash\psi} \qquad (\forall\text{-C})\ \frac{\Gamma\vdash\varphi[c/x]}{\Gamma\vdash\forall x\varphi}$$ siendo $c\in C$ nueva

(σ) $\Gamma\vdash\sigma$ para todo axioma de la igualdad σ.

$$(\mu\text{-C})\ \frac{\Gamma\vdash(\mu X.G)^{(1)}}{\Gamma\vdash(\mu X.G)}$$

$$(\neg\mu\text{A})\ \frac{\Gamma\vdash\neg(\mu X.G)^{(i)}\ \text{para todo } i\in N}{\Gamma\vdash\neg(\mu X.G)}$$

Evidentemente los axiomas y reglas de primer orden de i forman un cálculo correcto y completo para EL_1, al que hemos añadido un axioma composicional para cada una de las posibles estructuras de una expresión. Las reglas (μ-A) y ($\neg\mu$C) hacen referencia a la semántica operacional del operador μ de iterar todas las definiciones recursivas hasta encontar el valor requerido.

10- TEOREMA *(Corrección y Completitud)*

Para todo conjunto de fórmulas $\Phi \cup \{\varphi\}$ de LRFs:

$$\Phi \vdash \varphi \iff \Phi \models \varphi \ .$$

Demostración

CORRECION: Por inducción transfinita sobre la profundidad ξ del árbol de deducción, demostramos que la raíz de cualquier árbol de deducción es correcta. Decimos que una secuencia $\Gamma \vdash \varphi$ es correcta si $\Gamma \models \varphi$.

$\xi=0$: Hay que comprobar que todas las instancias de los axiomas $\Gamma \vdash \varphi$ del cálculo son correctas; es decir, $\Gamma \models \varphi$.

$\xi>0$: Hay que comprobar que todas las instancias de las reglas del cálculo verifican que la conclusión es correcta si lo son las premisas.

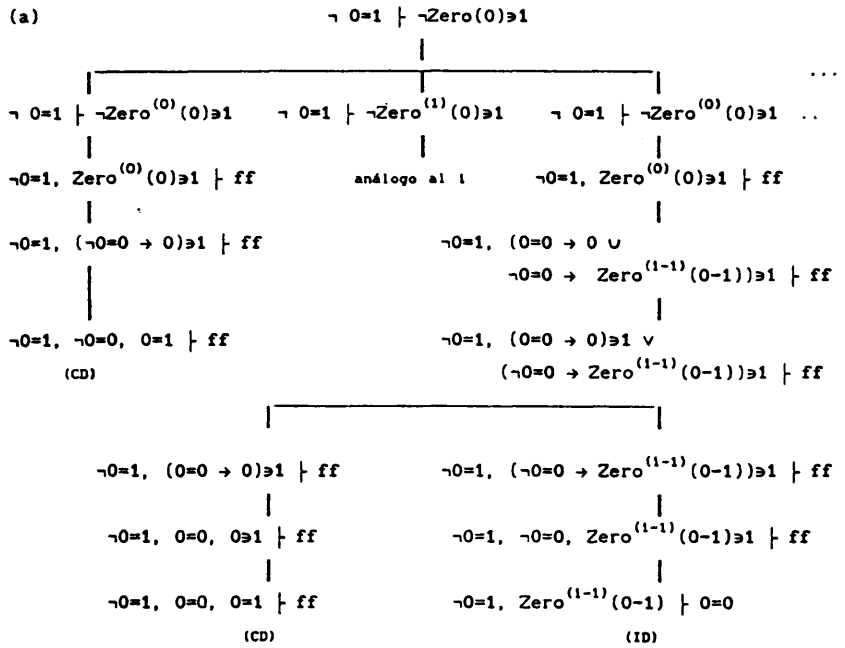
COMPLETITUD: Basta con demostrar que el cálculo verifica las condiciones del teorema 16. Todas son inmediatas:

(1) usando la regla (FS), (2) usando (σ) y (RC), (3) usando ($\wedge$-A) y (RC) basta demostrar $\Gamma \vdash \alpha \to \alpha_1 \wedge \alpha_2$ para cada tipo de fórmula α, (4) usando ($\vee$-A) y (RC) basta demostrar $\Gamma \vdash \beta \to \beta_1 \vee \beta_2$ para cada tipo de fórmula β, (5) usando ($\forall$-A), (6) usando ($\exists$-A), (7) usando ($\neg\mu$C), (8) usando (μ-A), (9) usando (CD) y (10) usando (RA). ■

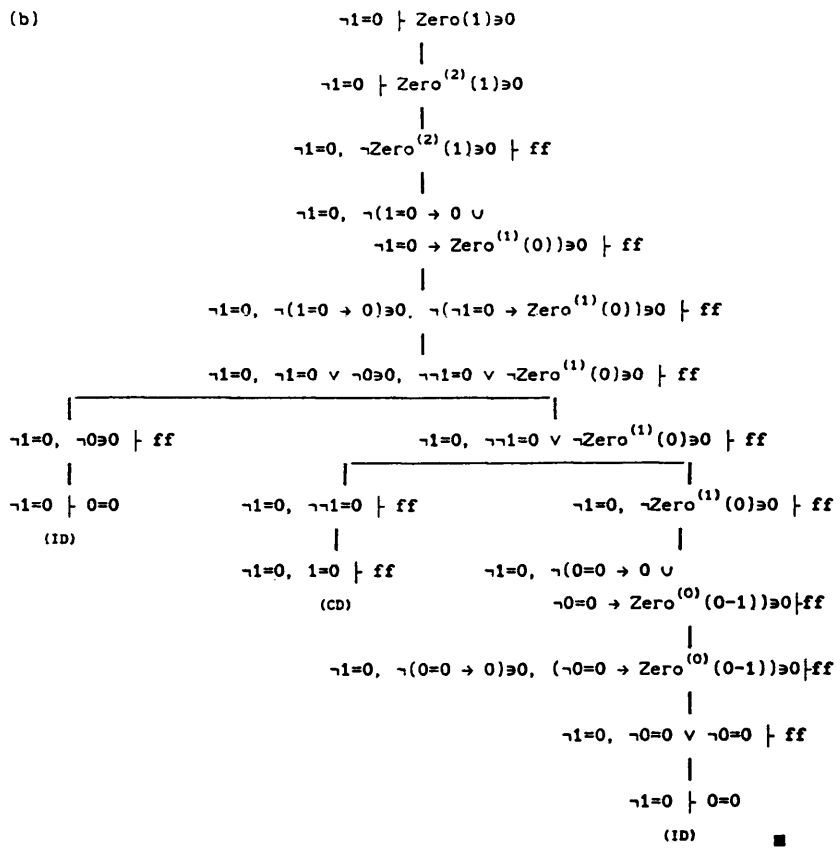
11- EJEMPLO

Veamos los mismos ejemplos que presentamos antes para los tableaux. La idea es la misma que entonces.

Tanto en la derivación (a) como en la (b), de las siguientes dos paginas, hemos realizado varios pasos en uno, en los casos en que estos corresponden a la lógica de primer orden, para no hacer los árboles demasiados grandes.


(a)
¬ 0=1 ⊢ ¬Zero(0)∋1
¬ 0=1 ⊢ ¬Zero(0)(0)∋1
¬ 0=1 ⊢ ¬Zero(1)(0)∋1
análogo al 1
¬ 0=1 ⊢ ¬Zero(0)(0)∋1
¬0=1, Zero(0)(0)∋1 ⊢ ff
¬0=1, Zero(0)(0)∋1 ⊢ ff
¬0=1, (¬0=0 → 0)∋1 ⊢ ff
¬0=1, (0=0 → 0 ∪
¬0=0 → Zero(1-1)(0-1))∋1 ⊢ ff
¬0=1, ¬0=0, 0=1 ⊢ ff
(CD)
¬0=1, (0=0 → 0)∋1 ∨
(¬0=0 → Zero(1-1)(0-1))∋1 ⊢ ff
¬0=1, (0=0 → 0)∋1 ⊢ ff
¬0=1, (¬0=0 → Zero(1-1)(0-1))∋1 ⊢ ff
¬0=1, 0=0, 0∋1 ⊢ ff
¬0=1, ¬0=0, Zero(1-1)(0-1)∋1 ⊢ ff
¬0=1, 0=0, 0=1 ⊢ ff
(CD)
¬0=1, Zero(1-1)(0-1) ⊢ 0=0
(ID)

(b)


¬1=0 ⊢ Zero(1)∋0
¬1=0 ⊢ Zero(2)(1)∋0
¬1=0, ¬Zero(2)(1)∋0 ⊢ ff
¬1=0, ¬(1=0 → 0 ∪
¬1=0 → Zero(1)(0))∋0 ⊢ ff
¬1=0, ¬(1=0 → 0)∋0, ¬(¬1=0 → Zero(1)(0))∋0 ⊢ ff
¬1=0, ¬1=0 ∨ ¬0∋0, ¬¬1=0 ∨ ¬Zero(1)(0)∋0 ⊢ ff
¬1=0, ¬0∋0 ⊢ ff
¬1=0 ⊢ 0=0
(ID)
¬1=0, ¬¬1=0 ∨ ¬Zero(1)(0)∋0 ⊢ ff
¬1=0, ¬¬1=0 ⊢ ff
¬1=0, 1=0 ⊢ ff
(CD)
¬1=0, ¬Zero(1)(0)∋0 ⊢ ff
¬1=0, ¬(0=0 → 0 ∪
¬0=0 → Zero(0)(0-1))∋0⊢ff
¬1=0, ¬(0=0 → 0)∋0, (¬0=0 → Zero(0)(0-1))∋0⊢ff
¬1=0, ¬0=0 ∨ ¬0=0 ⊢ ff
¬1=0 ⊢ 0=0
(ID)

■

Puesto que los modelos para los conjuntos de Hintikka son numerables, a partir del teorema de corrección y completitud del método de los tableaux, se deduce de forma inmediata

12- TEOREMA *(Löwenheim-Skolem)*

Toda LRFs-fórmula satisfactible tiene un modelo numerable. ■

Como ya hemos dicho, el problema de validez para LRFs es Π^1_1-duro, además, por el anterior teorema podemos demostrar que es Π^1_1 y por tanto concluir que es Π^1_1-completo.

13- TEOREMA

El problema de validez para LRFs es Π^1_1-completo. Más exactamente, para una signatura Σ suficientemente rica, el conjunto de todas las Σ-fórmulas de LRFs que son válidas en todas las Σ-estructuras es Π^1_1-completo.

Demostración

Para una signatura suficientemente rica, es sabido que el problema de validez para QDL es Π^1_1-completo (cfr. sección 1.1). Como QDL es expresable en LRFs (teorema 4 de la sección 2.4), se puede adaptar de forma sencilla, la demostración de que el problema de validez para QDL es Π^1_1-duro (cfr. [HMP 77]) a LRFs. Obtenemos así, que el problema de validez para LRFs Π^1_1-duro; es es decir, todo problema Π^1_1 se puede reducir a él.

Por lo tanto es suficiente probar que el problema de validez para LRFs pertenece a la clase Π^1_1. Esto se verifica para cualquier signatura numerable Σ, por el teorema de Löwenheim-Skolem para LRFs.

De hecho, para cualquier Σ-fórmula φ de LRFs, tenemos:

φ es válida en toda estructura sii
φ es válida en toda estructura numerable.

Como, las Σ-estructuras numerables $\mathfrak{D}$ pueden codificarse como funciones totales f sobre números naturales, y las Σ-fórmulas de LRFs φ pueden codificarse como números naturales n de tal forma que la relación $\mathfrak{D} \vdash \varphi$ puede verse como una relación aritmética R(f,n) (cfr. [Rog 67]). Mediante esta codificación se concluye que el conjunto de Σ-fórmulas de LRFs válidas es Π^1_1. ■

2.- UN CALCULO ARITMETICO PARA LRF

En esta sección presentamos un cálculo finitario aritméticamente correcto y completo para LRFc, en el mismo sentido que el cálculo aritmético de Harel para QDL. La completitud de este tipo de cálculos, se basa en la expresividad de la lógica de primer orden sobre estructuras aritméticas. De manera que a cada fórmula de LRFc le hacemos corresponder una fórmula de EL_1 lógicamente equivalente bajo estructuras aritméticas, de acuerdo con la noción de completitud relativa de Cook para corrección de programas.

Este proceso se complica para el operador μ, bastante más que para el operador * en la lógica dinámica. En este caso, haciendo uso de que en estructuras aritméticas podemos cuantificar números naturales, codificar grafos finitos de funciones y aplicar el principio de inducción, expresamos mediante una fórmula de primer orden extendido la caracterización del menor punto fijo de la definición 8 de la sección 2.2.

En este sentido, la noción de estructura aritmética, esta basada en exigir a una estructura, las condiciones naturales necesarias para que esta definición sea posible. Con el término de "condiciones naturales", nos referimos al hecho de que en la programación real "siempre" se dispone de estas condiciones.

En el siguiente capítulo ampliaremos esta clase de estructuras, de forma que conteniendo las condiciones necesarias para la expresividad, sean axiomatizables. Lo cual no ocurre para las estructuras aritméticas, y es muy conveniente, puesto que en el cálculo aritmético la teoría de una estructura hace el papel de oráculo.

En primer lugar, definimos la noción de estructura aritmética en el sentido de Harel (cfr. [Har 79]).

1- DEFINICION *(Estructura Aritmética)*

Una Σ-estructura $\mathfrak{D}$ es aritmética si Σ incluye un símbolo de predicado monádico nat, dos constantes 0, 1, y dos símbolos de función binarias +, comp, tal que se verifican las siguientes condiciones:

(1) $\mathbb{N} \subseteq D$ y para todo $x \in D$: $nat^{\mathfrak{D}}(x)$ = true sii $x \in \mathbb{N}$.
(2) $0^{\mathfrak{D}} = 0 \in \mathbb{N}$, $1^{\mathfrak{D}} = 1 \in \mathbb{N}$, y $x +^{\mathfrak{D}} y = x+y$ para todo $x, y \in \mathbb{N}$.
(3) para toda secuencia finita $\langle x_1, \ldots, x_n \rangle$ con $x_1 \in D$, existe algun $u \in D$ tal que $x_i = comp^{\mathfrak{D}}(i,u)$ para $0 \leq i \leq n$. ■

2- DEFINICION *(Símbolos definidos)*

Introducimos ahora dos símbolos definidos de predicados binarios : < y ≤, de forma que para todo par de términos t_1 y t_2:

$t_1 < t_2$ suple a $\exists x\ ((t_1 + x) + 1 = t_2)$
$t_1 \leq t_2$ suple a $t_1 < t_2 \vee t_1 = t_2$. ■

Además, para simplificar la notación, usaremos k, l, n y m como variables de individuo que solo pueden interpretarse con valores naturales; es decir que verifican el predicado nat, y nos referiremos a comp(i,u) por u[i].

EXPRESIVIDAD DE LAS ESTRUCTURAS ARITMETICAS

Como hemos dicho al inicio de esta sección, la completitud aritmética de un cálculo finitario para LRFc se basa en la expresividad de la lógica de primer orden extendida en estructuras aritméticas, de acuerdo con la noción de completitud relativa de Cook (cfr. [Coo 78]). De forma que a cada fórmula de LRFc le asociamos una fórmula de EL_1 equivalente sobre estructuras aritméticas.

En primer lugar definimos esta traducción asignando:

(a) A cada fórmula φ, una fórmula E_φ, con las mismas variable libres que tiene φ equivalente a φ.

(b) A cada expresión M, una fórmula $E_M(y)$ con una variable libre más que M, que intuitivamente admite los mismos valores que produce M.

(c) A cada expresión funcional F, una fórmula $E_F(\vec{x},y)$ con n+1 variables libre más que F (siendo n la aridad de F), que intuitivamente define el grafo de F.

por inducción simultánea sobre la función de complejidad $|.|$ definida en la sección anterior.

(a) Traslación de las fórmulas

$\varphi = t_1 = t_2$: $E_\varphi = t_1 = t_2$

$\varphi = p(M_1,\ldots,M_n)$: $E_\varphi = \exists \vec{y}\ (\bigwedge_{1\le j\le n} E_{Mj}(y_j) \wedge p(\vec{y})\ )$

siendo $\vec{y}$ variables nuevas y distintas entre sí.

$\varphi = M \ni t$: $E_\varphi = E_M(y)[t/y]$

$\varphi = \neg\psi$: $E_\varphi = \neg E_\psi$

$\varphi = \psi_1 \vee \psi_2$: $E_\varphi = E_{\psi_1} \vee E_{\psi_2}$

$\varphi = \exists x\psi$: $E_\varphi = \exists x E_\psi$

(b) Traslación de las expresiones

$M = t \in TER_\Sigma$: $E_M(y) = t = y$

$M = F(M_1,\ldots,M_n)$: $E_M(y) = \exists \vec{y}(\bigwedge_{1\le j\le n} E_{Mj}(y_j) \wedge E_F(\vec{y},y))$

siendo $\vec{y}$ variables nuevas y distintas entre sí.

$M = M_1 \cup M_2$: $E_M(y) = E_{M1}(y) \vee E_{M2}(y)$

$M = \varphi \to N$: $E_M(y) = E_\varphi \wedge E_N(y)$

$M = c x \varphi$: $E_M(y) = E_\varphi[y/x]$

(c) Traslación de las expresiones funcionales

F=f: $E_F(\vec{x},y) = f(\vec{x}) = y$

F=X: $E_F(\vec{x},y) = X(\vec{x}) \ni y$

$F=\lambda\vec{z}.M$: $E_F(\vec{x},y) = E_M(y)[\vec{x}/\vec{z}]$

$F=\mu X.G$: $E_F(\vec{x},y) = \exists k\ BE_F(k,\vec{x},y)$

Como ya hemos dicho, esta última fórmula se basa en la caracterización del menor punto fijo del teorema 12 de la sección 2.2, de forma que el significado de la fórmula $BE_F(k,\vec{x},y)$ tiene que ser: "puede construirse una derivación de longitud k+1 que prueba que $F(\vec{x})$ admite el valor y". Este es el punto clave de la traducción. A continuación vamos a desarrollar la definición formal de BE_F en varias etapas, explicando su significado.

$$(1)\ BE_F(k,\vec{x},y) = \exists\vec{u}\ \exists v\ (CD_{G,X}(k,\vec{u},v) \wedge CP(k,\vec{u},v,\vec{x},y)$$

Que quiere decir: "Las secuencias codificadas por $\vec{u},v$ son una derivación de longitud k+1 c.r. al operador asociado a G,X y los k-ésimos miembros de estas secuencias muestran que esta derivación produce la tupla $\langle\vec{x},y\rangle$".

$$(2)\ CP(k,\vec{u},v,\vec{x},y) = \left(\bigwedge_{1\le j\le n} u_j[k] = x_j\right) \wedge v[k] = y$$

Que quiere decir: "Las k-ésimas componentes de las secuencias $\vec{u},v$ forman la tupla $\langle\vec{x},y\rangle$".

$$(3)\ CD_{G,X}(k,\vec{u},v) = \forall l\ (0\le l \wedge l\le k \longrightarrow PV_{G,X}(l,\vec{u},v)).$$

donde

$$PV_{G,X}(l,\vec{u},v) = E_{G[\lambda\vec{x}.\varepsilon yGR(l,\vec{u},v,\vec{x},y)/X,u_1[l]/x_1,\ldots,u_n[l]/x_n,v[l]/y]}(\vec{x},y)$$

y

$$GR(l,\vec{u},v,\vec{x},y) = \exists m\ (m<l \wedge \left(\bigwedge_{1\le j\le n} u_j[m] = x_j\right) \wedge v[m] = y)$$

Que quiere decir: "Para $0\le l\le k$, la l-ésima tupla $\langle\vec{u}[l],v[l]\rangle$ dada por los códigos de las secuencias, satisface que v[l] es un posible resultado de la evaluación de G cuando las variables de individuo $\vec{x}$ son interpretadas como $\vec{u}[l]$, y la variable funcional X es interpretada como la función finita cuyo grafo son las tuplas $\langle\vec{u}[m],\vec{v}[m]\rangle$, $m<l$ dadas por las secuencias".

Obsérvese que solamente es necesaria el uso de la función de complejidad en la definición de $PV_{G,X}$, puesto que en los demás casos la traducción se reduce a la de sus componentes. La función de complejidad verifica que la complejidad de Θ (fórmula, expresión o expresión funcional) es mayor que la de sus componentes, y además permite definir $PV_{G,X}$ ya que $|\mu X.G| = \omega + |G|$ y $|PV_{G,X}| \leq |G| + \xi$ con $\xi < \omega$ puesto que $|GR|$ es finita. Evidentemente que la fórmula

$$PV'_{G,X}(1, \vec{u}, v) = E_G(\vec{x}, y)[\lambda \vec{x}.cyGR(1, \vec{u}, v, \vec{x}, y)/X, u_1[1]/x_1, \ldots, u_n[1]/x_n, v[1]/y]$$

es equivalente a $PV_{G,X}$ puesto que ambas, aplicando sustitución una vez demostrada la corrección de la traducción, definen a la función

$$G[\lambda \vec{x}.cyGR(1, \vec{u}, v, \vec{x}, y)/X, u_1[1]/x_1, \ldots, u_n[1]/x_n, v[1]/y]$$

además, como G es existencial en X, para mayor valor de 1 en $PV_{G,X}(1, \vec{u}, v)$ (análogamente en PV') mayor es la función que define. Por último, la definición de $PV'_{G,X}$ se reduce a la de su componente, pero no es de primer orden extendido. Esto podía haberse evitado definiendo una sustitución especial de variables funcionales por fórmulas de EL_1 sobre fórmulas de EL_1, pero pensamos que esta función de complejidad es bastante natural.

El siguiente resultado técnico será usado en el próximo teorema.

3- LEMA

Sea Θ cualquier Σ-expresión funcional, expresión o fórmula de LRFc. Si Θ es positiva (resp. negativa, existencial, universal) en una variable funcional Y dada, entonces la Σ-fórmula de primer orden extendido que la expresa E_Θ tiene la misma propiedad.

Demostración

Por inducción simultánea sobre la función $|.|$ para las Σ-fórmulas, Σ-expresiones y Σ-expresiones funcionales. Consideremos por ejemplo, el caso $F = \mu X.G$, donde X es distinta de Y. Si asumimos que G es positiva en Y,

entonces $E_G(\vec{x},y)$ es también positiva en Y, por hipótesis de inducción. Inspeccionando $E_F(\vec{x},y)$, se comprueba que todas las apariciones libres de Y en $E_F(\vec{x},y)$ están en la subfórmula $PV_{G,X}(1,\vec{u},v)$ y corresponden a las apariciones de Y en $E_G(\vec{x},y)$, afectadas por dos símbolos de negación adicionales. (Esto es fácil de ver reduciendo $\forall$ e $\rightarrow$ a $\exists,\neg$ y $\vee$ en $CD_{G,X}(k,\vec{u},v)$). Entonces es claro que $E_F(\vec{x},y)$ es positiva en Y. Los demás casos se comprueban más fácilmente. ■

La relación exacta entre Θ y E_Θ queda establecida por el siguiente teorema

4- TEOREMA *(Expresividad)*

Dada una Σ-estructura aritmética $\mathfrak{D}$ y Δ,δ un contexto y una valoración sobre $\mathfrak{D}$, respectivamente:

(a) para toda Σ-fórmula φ de LRFc existe una Σ-fórmula E_φ de primer orden extendido (con $lib(E_\varphi)=lib(\varphi)$) tal que $\varphi^{\mathfrak{D}}(\Delta,\delta)=E_\varphi^{\mathfrak{D}}(\Delta,\delta)$.

(b) para toda Σ-expresión M de LRFc existe una Σ-fórmula $E_M(y)$ de primer orden extendido tal que $M^{\mathfrak{D}}(\Delta,\delta)=\{\ y\in D\ /\ E_M^{\mathfrak{D}}(\Delta,\delta[y/y])=true\ \}$, siendo y una variable nueva y $lib(E_M)\cup\{y\}=lib(M)$.

(c) para toda Σ-expresión funcional F de LRFc existe una Σ-fórmula $E_F(\vec{x},y)$ de primer orde extendido, siendo $\vec{x}$ e y variables nuevas y distintas entre sí, con $lib(E_F)\cup\{\vec{x},y\}=lib(F)$, tal que
$F^{\mathfrak{D}}(\Delta,\delta)(\vec{x})=\{y\in D/E_F^{\mathfrak{D}}(\Delta,\delta[\vec{x}/\vec{x},y/y])=true\ \}$.

Demostración

Hay que demostrar que la traducción anterior es correcta, por inducción simultánea sobre la función de complejidad $|.|$ para las fórmulas, expresiones y expresiones funcionales. Veamos para (c) el caso de $F=\mu X.G$: Sea T el operador continuo asociado a F. Por el teorema 12 de la sección 2.2, $\langle\vec{x},y\rangle\in fix(T)$ sii ex. k, $\vec{u}$, v tal que $\langle\vec{u},v\rangle$ es una T-derivación de longitud k+1 que produce $\langle\vec{x},y\rangle$. Por hipótesis de inducción,

$E_{G[\lambda\vec{x}.\,\varepsilon y GR(1,\vec{u},v,\vec{x},y)/X]}(\vec{x},y)$ define $T(fun_1(\langle\vec{u},v\rangle)$

(cfr. definición 8 de la sección 2.2), de forma que

$PV_{G,X}(1,\vec{u},v) = E_{G[\lambda\vec{x}.\,\varepsilon y GR(1,\vec{u},v,\vec{x},y)/X,\,u_1[1]/x_1,\ldots,u_n[1]/x_n,\,v[1]/y]}(\vec{x},y)$

afirma que $v[1]\in T(fun_1(\vec{u},v))(\vec{u}[1])$.

Además $CP(k,\vec{u},v,\vec{x},y) = (\bigwedge_{1\le j\le n} u_j[k] = x_j) \wedge v[k] = y$

afirma que $\langle\vec{x},y\rangle$ es producido por la T-derivación $\langle\vec{u},v\rangle$ de longitud $k^{\circ}1$, puesto que $\langle\vec{x},y\rangle$ son los k-ésimos elementos de $\langle\vec{u},v\rangle$. ■

CALCULO ARITMETICO

Presentamos, al igual que en la sección anterior, un cálculo de secuencias, que solo se diferencia del anterior en el tratamiento del operador μ. En particular, el cálculo, denominado α, consiste de los siguientes axiomas y reglas:

Axiomas y reglas para primer orden

Como en el cálculo infinitario.

Axiomas y reglas para la descomposición de las expresiones

Como en el cálculo infinitario, excepto las reglas (μ-A) y ($\neg\mu$C) para el operador μ, y añadiendo las siguientes dos nuevas reglas:

Regla de Invarianza

$$(I^{\circ})\ \frac{\Gamma,\ G[\lambda\vec{x}.\,\varepsilon y.E(\vec{x},y)/X](\vec{x}) \ni y \vdash E(\vec{x},y)\qquad \Gamma \vdash \neg E(\vec{t},t)}{\Gamma \vdash \neg(\mu X.G)(\vec{t}) \ni t}$$

Regla de Convergencia

$$(C^{\bullet})\ \frac{\Gamma,\ E'(k,\vec{x},y)\ \vdash\ G[\lambda\vec{x}.\varepsilon y\ \exists l\ (l < k \wedge E'(l,\vec{x},y))/X](\vec{x})\ni y \qquad \Gamma\ \vdash\ \exists k\ E'(k,\vec{t},t)}{\Gamma\ \vdash\ (\mu X.G)(\vec{t})\ni t}$$

Las últimas dos reglas son los ingredientes esenciales del cálculo y necesitan alguna explicación adicional. $E(\vec{x},y)$ se entiende como una fórmula de primer orden extendida que contiene a $\vec{x}$ e y entre sus variables libres. $\vec{t},t$ denotan términos, de forma que $E(\vec{t},t)$ es una abreviatura de $E(\vec{x},y)[\vec{t}/\vec{x},t/y]$. G se entiende como una expresión funcional que debe ser existencial en la variable funcional X. La notación en $(C^{\bullet})$ debe entenderse de forma similar.

El siguiente resultado muestra que ambas reglas reflejan la semántica de la recursión adecuadamente. La idea es que E y E' son fórmulas que expresan una función en el sentido del teorema de expresividad. De esta forma, intuitivamente la primera premisa de $(I^{\bullet})$ afirma que $T(E)\lhd E$ y la segunda afirma que la tupla $\langle\vec{t},t\rangle$ no pertenece al grafo de E, de lo que se concluye que tampoco pertenece al grafo de fix(T). Análogamente, la primera premisa de $(C^{\bullet})$ afirma que $E'_k \lhd T(E'_{k+1})$ y la segunda que $\langle\vec{t},t\rangle$ es una tupla del grafo de $\exists k E'_k$, de lo que se concluye que también pertenece al grafo de fix(T). La corrección de este razonamiento se demuestra a continuación. Pero primero, definimos el concepto de deducibilidad formal en este cálculo.

5- DEFINICION

(a) Dada una estructura $\mathfrak{D}$: $\alpha(\mathfrak{D})$ es el cálculo que se obtiene de añadir al cálculo α los axiomas $\Gamma\vdash E$, para toda $E\in ETh(\mathfrak{D})$, donde $ETh(\mathfrak{D})$ es la teoría de primer orden extendido de $\mathfrak{D}$, es decir

$$ETh(\mathfrak{D}) = \{\ E\in EFOR_{\Sigma}\ /\ \mathfrak{D}\vdash E\ \}$$

(b) Dada una fórmula φ, φ es demostrable en el cálculo $\alpha(\mathfrak{D})$, denotado por $ETh(\mathfrak{D})\vdash_{\alpha}\varphi$ o simplemente por $ETh(\mathfrak{D})\vdash\varphi$, si existe un árbol de deducción finito para $\varnothing\vdash\varphi$, es decir un árbol de deducción cuya raíz está etiquetada por $\varnothing\vdash\varphi$. Donde ahora, árbol de deducción, se define de forma análoga a la definición 9, pero finitamente ramificado. ■

6- TEOREMA *(Correción y Completitud Aritméticas)*

El cálculo anterior α es aritméticamente correcto y completo para LRFc. Es decir: para una Σ-estructura aritmética $\mathfrak{D}$ y una fórmula de LRFc $\varphi \in FFOR_{\Sigma}$, se verifica

$$\mathfrak{D} \models \varphi \iff ETh(\mathfrak{D}) \vdash_{\alpha} \varphi$$

Demostración

CORRECION: $ETh(\mathfrak{D}) \vdash_{\alpha} \varphi \Rightarrow \mathfrak{D} \models \varphi$.

Por inducción sobre la profundidad N del árbol de deducción, demostramos que la raiz de cualquier árbol de deducción es correcta en $\mathfrak{D}$. Decimos que una secuencia $\Gamma \vdash \varphi$ es correcta en $\mathfrak{D}$, si $\mathfrak{D} \models \Gamma$ implica $\mathfrak{D} \models \varphi$, y lo denotamos por $\Gamma \models_{\mathfrak{D}} \varphi$.

N=0: Hay que comprobar que todas las instancias de los axiomas $\Gamma \vdash \varphi$ del cálculo α son correctas en $\mathfrak{D}$; es decir $\Gamma \models_{\mathfrak{D}} \varphi$.

N>0: Hay que comprobar que todas las instancias de las reglas del cálculo verifican que la conclusión es correcta en $\mathfrak{D}$ si lo son las premisas.

Todas las demostraciones son una restricción a estructuras aritméticas de las demostraciones del cálculo infinitario, ya que había que demostrar lo mismo pero sobre cualquier estructura, salvo para las nuevas reglas.

Consideremos una instancia de $(I^{\bullet})$ y otra de $(C^{\bullet})$. Supongamos que sus premisas son correctas en $\mathfrak{D}$ y probamos que sus conclusiones también lo son. Para esto, tomemos un contexto Δ y una valoración δ sobre $\mathfrak{D}$ y consideramos:

(a) El operador monótono y continuo $T: \mathcal{P}\mathfrak{I}_{\mathfrak{D}}^{(n)} \longrightarrow \mathcal{P}\mathfrak{I}_{\mathfrak{D}}^{(n)}$ dado por

$$T(X) = G^{\mathfrak{D}}(\Delta[X/X], \delta)$$

(b) El menor punto fijo de T:

$$Y = (\mu X.G)^{\mathfrak{D}}(\Delta, \delta)$$

(c) La función no determinista $Z \in \mathcal{P}\mathfrak{I}_{\mathfrak{D}}^{(n)}$ definida por la condición

$$y \in Z(\vec{x}) \text{ sii } E(\vec{x}, y)^{\mathfrak{D}}(\Delta, \delta[\vec{x}/\vec{x}, y/y])$$

(d) Para cada número natural k, $Z_k \in \Pi\mathfrak{I}_D^{(n)}$ definida por la condición

$$y \in Z_k(x) \text{ sii } E'(k, \vec{x}, y)^{\mathfrak{D}}(\Delta, \delta[k/k, \vec{x}/\vec{x}, y/y])$$

(e) Los datos d_1, d definidos por $d_i = t_i^{\mathfrak{D}}(\Delta, \delta)$, $1 \leq i \leq n$; $d = t^{\mathfrak{D}}(\Delta, \delta)$

Supongamos que $\mathfrak{D} \vdash \Gamma$:

Para (I^*) Tenemos que probar que $\mathfrak{D} \vdash \neg(\mu X.G)(\vec{t}) \ni t$. El hecho de que la primera premisa sea correcta en $\mathfrak{D}$ significa que $T(Z) \triangleleft Z$. Por el lema 10 de la sección 2.2, se sigue que $Y \triangleleft Z$. El hecho de que la segunda premisa sea correcta en $\mathfrak{D}$ implica que $d \in Z(\vec{d})$. Por tanto, $d \in Y(\vec{d})$, y la conclusión es correcta en $\mathfrak{D}$.

Para (C^*) Tenemos que probar que $\mathfrak{D} \vdash (\mu X.G)(\vec{t}) \ni t$. Puesto que la primera premisa es correcta en $\mathfrak{D}$, sabemos que $Z_k \triangleleft T(\bigsqcup_{l<k} Z_l)$ para todo natural k. Como todas las Z_k están uniformemente definidas por $E'(k, \vec{x}, y)$ y $\mathfrak{D}$ es aritmética, por inducción sobre k podemos concluir que $Z_k \triangleleft Y$ para todo natural k. En efecto:

k=0: $Z_0 = T(\perp) \triangleleft Y$, puesto que $T(\bigsqcup_{l<0} Z_l) = T(\perp)$ y $Y = fix(T)$.

k>0: $Z_k \triangleleft T(\bigsqcup_{l<k} Z_l) = \bigsqcup_{l<k} T(Z_l) \triangleleft \bigsqcup_{l<k} T(Y) = T(Y) = Y$ por hipotesis de inducción y monotonía de T.

Por otro lado, el hecho de que la segunda premisa sea correcta en $\mathfrak{D}$ implica que $d \in Z_k(\vec{d})$ para algún natural k. Por lo tanto, $d \in Y(\vec{d})$ y la conclusión es correcta en $\mathfrak{D}$.

COMPLETITUD: $\mathfrak{D} \vdash \varphi \Rightarrow ETh(\mathfrak{D}) \vdash_G \varphi$.

Evidentemente, es suficiente probar los siguientes tres puntos (donde (2) y (3) juegan un papel auxiliar):

(1) Para toda $\varphi \in FFOR_\Sigma$: $ETh(\mathfrak{D}) \vdash_G \varphi \leftrightarrow E_\varphi$

(2) Para toda $M \in EXP_\Sigma$: $ETh(\mathfrak{D}) \vdash_G M \ni y \leftrightarrow E_M(y)$

(3) Para toda $F \in FEXP_\Sigma^{(n)}$: $ETh(\mathfrak{D}) \vdash_G F(\vec{x}) \ni y \leftrightarrow E_F(\vec{x}, y)$

En efecto, si $\mathfrak{D}\vdash\varphi$ entonces $E_\varphi\in ETh(\mathfrak{D})$, es decir $ETh(\mathfrak{D})\vdash E_\varphi$, y aplicando (1) y (MP) se obtiene $ETh(\mathfrak{D})\vdash\varphi$.

Para demostrar (1)-(3), se procede por inducción simultánea sobre la función de complejidad $|.|$ de φ, M y F.

Veamos un esquema de la prueba de (3) cuando F es $\mu X.G$, los demás casos son sencillos. Vamos a probar:

(a) $ETh(\mathfrak{D}),\ \neg E_F(\vec{x},y)\ \vdash\ \neg F(\vec{x})\ni y$
(b) $ETh(\mathfrak{D}),\ E_F(\vec{x},y)\ \vdash\ F(\vec{x})\ni y$

Para esto usaremos las reglas (I^*) y (C^*).

(a) Usamos la regla (I^*) de forma que como $E(\vec{x},y)$ tomamos la fórmula $\exists k\ BE_F(k,\vec{x},y)$, es decir, $E_F(\vec{x},y)$, cfr. teorema de expresividad. Con esto, la demostración formal para (a) puede construirse como sigue:

$$\dfrac{ETh(\mathfrak{D}),\ \neg E_F(\vec{x},y)\ \vdash\ \neg F(\vec{x})\ni y}{|\ -\!\!-\ (I^*)}$$

$$\dfrac{\dfrac{}{ETh(\mathfrak{D}),\ \neg E_F(\vec{x},y)\ \vdash\ \neg E_F(\vec{x},y)}\ (SP) \qquad \dfrac{\dfrac{\dfrac{}{ETh(\mathfrak{D}),\ \neg E_F(\vec{x},y),\ G[\lambda\vec{x}.\varepsilon y\ E(\vec{x},y)/X](\vec{x})\ni y\ \vdash\ E(\vec{x},y)}}{ETh(\mathfrak{D}),\ \neg E_F(\vec{x},y)\ \vdash\ \neg G[\lambda\vec{x}.\varepsilon y\ E(\vec{x},y)/X](\vec{x})\ni y}}{ETh(\mathfrak{D}),\ \neg E_F(\vec{x},y)\ \vdash\ \neg E_{G[\lambda\vec{x}.\varepsilon y\ E(\vec{x},y)/X]}(\vec{x},y)}\ (a.1)$$

Luego basta con demostrar:

$$\mathfrak{D}\ \vdash\ \neg E_F(\vec{x},y)\ \longrightarrow\ \neg E_{G[\lambda\vec{x}.\varepsilon y\ E(\vec{x},y)/X]}(\vec{x},y)$$

es decir

(a.1) $\mathfrak{D} \vdash E_{G[\lambda\vec{x}.\varepsilon y\ E(\vec{x},y)/X]}(\vec{x},y) \rightarrow E_F(\vec{x},y)$

Sean un contexto Δ y una valoración δ sobre $\mathfrak{D}$ fijadas. Consideremos T, Y, Z, definidas como en la demostración de la corrección. Para que (a.1) sea cierta en $\mathfrak{D}$ bajo Δ,δ es suficiente que

(a.2) $T(Z) \leq Z$

Además en este caso se da la igualdad.

Como E es E_F, sabemos que $Z=(\mu X.G)^{\mathfrak{D}}(\Delta,\delta)$, es decir, Z=Y. Por el teorema 4, Y es el menor punto fijo de T, y por tanto se verifica T(Y)=Y, es decir (a.2).

(b) Usamos la regla (C^*) de forma que como $E'(k,\vec{x},y)$ tomamos la fórmula $BE_F(k,\vec{x},y)$, cfr. teorema de expresividad. Con esto, la demostracione formal para (b) puede construirse como sigue:

$$\dfrac{\dfrac{}{ETh(\mathfrak{D}),\ E_F(\vec{x},y) \vdash F(\vec{x})\ni y}\ (C^*)}{\dfrac{ETh(\mathfrak{D}),\ E_F(\vec{x},y) \vdash \exists k\ E'(k,\vec{x},y)}{(SP)} \qquad \dfrac{ETh(\mathfrak{D}),\ E_F(\vec{x},y),\ E'(k,\vec{x},y) \vdash G[\lambda\vec{x}.\varepsilon y\ \exists 1(1<k \wedge E'(1,\vec{x},y))/X](\vec{x})\ni y}{ETh(\mathfrak{D}),\ E_F(\vec{x},y),\ E'(k,\vec{x},y) \vdash E_{G[\lambda\vec{x}.\varepsilon y\ \exists 1(1<k \wedge E'(1,\vec{x},y))/X]}(\vec{x},y)}\ (b.1)}$$

Luego basta con demostrar:

(b.1) $\mathfrak{D} \vdash E'(k,\vec{x},y) \rightarrow E_{G[\lambda\vec{x}.\varepsilon y\ \exists 1(1<k \wedge E'(1,\vec{x},y))/X]}(\vec{x},y)$

Sean un contexto Δ y una valoración δ sobre $\mathfrak{D}$ fijadas. Consideremos T, Y, y Z_k (para todo natural k), definidas como en la demostración de la corrección. Para que (b.1) sea cierta en $\mathfrak{D}$ bajo Δ,δ es suficiente que

(b.2) $Z_k \in T(\bigcup_{l<k} Z_l)$ para todo natural k.

Además en este caso se da la igualdad.

Como E' es BE_F, el grafo de cada Z_k consiste exactamente de aquellas tuplas $\langle \vec{x}, y \rangle$ las cuales tienen una T-derivación de longitud k+1 (recordar la construcción de BE_F). Luego por definición de T-derivación se verifica que $Z_0 = T(\perp)$, $\bigcup_{l<k} Z_l = Z_{k-1}$ (puesto que una derivación que produce $\langle \vec{x}, y \rangle$ se convierte en otra de mayor longitud que también produce $\langle \vec{x}, y \rangle$, sin más que repetir alguno de sus elementos en el mismo lugar donde aparecen) y que $T(Z_{k-1}) = Z_k$. Luego se verifica (b.2). ∎

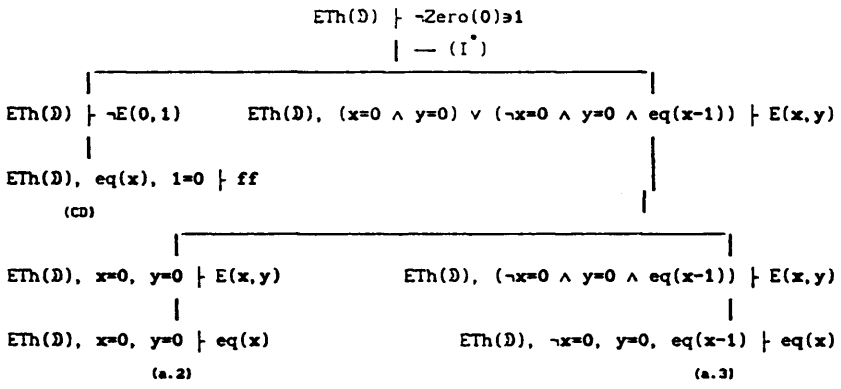
7- EJEMPLO

Veamos los mismos ejemplos que presentamos para el cálculo infinitario. Ahora, tanto cuando usemos ($I^\bullet$) como ($C^\bullet$), la idea es especificar mediante una fórmula de primer orden extendido, el grafo de Zero. Evidentemente, un par $\langle x,y \rangle \in$ grafo(Zero) si y=0 y x puede generarse a partir de las constantes 0 y 1 y la función + de la estructura aritmética $\mathfrak{D}$.

(a) Para obtener un árbol de derivación para $ETh(\mathfrak{D}) \vdash \neg Zero(0) \ni 1$, usaremos la regla ($I^\bullet$) de forma que

$E(x,y) = eq(x) \wedge y=0$ siendo
$eq(x) = \exists k\ \exists s\ (s(0)=0 \wedge s(k)=x \wedge \forall l\ (l<k \rightarrow s(l+1)=s(l)+1)$,
$G = \lambda x.\ (x=0 \rightarrow 0 \cup \neg x=0 \rightarrow X(x-1))$,
$G[\lambda x.\varepsilon y.E(x,y)/X] = \lambda x.\ (x=0 \rightarrow 0 \cup$
$\neg x=0 \rightarrow (\lambda x.\ \varepsilon y\ E(x,y))(x-1))$
y
$\vdash G[\lambda x.\varepsilon y.E(x,y)/X](x) \ni y \leftrightarrow (x=0 \wedge y=0) \vee (\neg x=0 \wedge y=0 \wedge eq(x-1))$.

Por tanto:



Razonemos que:

(a.2) $(x=0 \wedge y=0 \longrightarrow eq(x)) \in ETh(\mathfrak{D})$

(a.3) $(\neg x=0 \wedge y=0 \wedge eq(x-1) \longrightarrow eq(x)) \in ETh(\mathfrak{D})$

(a.2) Basta con tomar k=0 y s=<0>.

(a.3) Basta con tomar s=concat(s^C,<x>) y $k=k^C+1$, siendo k^C y s^C la longitud y la secuencia que existen para eq(x-1).

(b) Para obtener un árbol de derivación para $ETh(\mathfrak{D}) \vdash Zero(1) \ni 0$, usaremos la regla ($C^\bullet$) de forma que

$E'(k,x,y) = eq(k,x) \wedge y=0$ siendo

$eq(k,x) = \exists s\ (s(0)=0 \wedge s(k)=x \wedge \forall 1\ (1<k \rightarrow s(1+1)=s(1)+1)$,

$G = \lambda x.\ (x=0 \rightarrow 0 \cup \neg x=0 \rightarrow X(x-1))$,

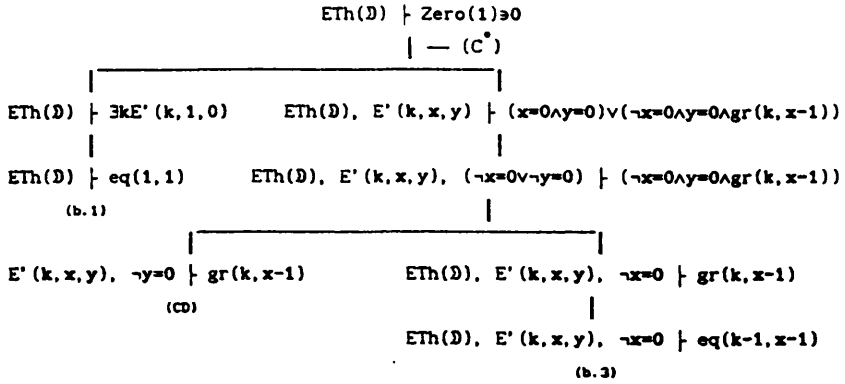
$G[\lambda x.\varepsilon y\ \exists 1\ (1<k \wedge E'(1,x,y))/X] = \lambda x.\ (x=0 \rightarrow 0 \cup$
$\neg x=0 \rightarrow (\lambda x.\ \varepsilon y\ \exists 1\ 1<k \wedge E'(1,x,y))(x-1))$

y

$\vdash G[\lambda x.\varepsilon y\ \exists 1\ (1<k \wedge E'(1,x,y))/X](x) \ni y \leftrightarrow$
$(x=0 \wedge y=0) \vee (\neg x=0 \wedge y=0 \wedge gr(k,x-1))$ siendo

$gr(k,x) = \exists 1\ 1<k \wedge eq(k,x) \wedge y=0$.

Por tanto:



Razonemos que:

(b.1) $eq(1,1) \in ETh(\mathfrak{D})$

(b.3) $(E'(k,x,y) \wedge \neg x=0 \rightarrow eq(k-1,x-1)) \in ETh(\mathfrak{D})$

(b.1) Basta con tomar $s=\langle 0,1\rangle$.

(b.3) Basta con tomar s tal que $s^C = concat(s,\langle x\rangle)$, siendo s^C la secuencia que existe para $eq(k,x)$, segun $E'(k,x,y)$. ■

CAPITULO 4:

ENFOQUES SEMANTICOS NO ESTANDAR: LAS LOGICAS NLRF Y ALRF

Como ya hemos dicho, el concepto de menor punto fijo de un operador no es un concepto de primer orden. En el capítulo anterior hemos abordado este problema dentro del marco de la semántica no estándar. En particular hemos demostrado que en estructuras aritméticas y para una expresión funcional ($\mu X.G$) de LRF_o este concepto puede expresarse mediante una fórmula de primer orden.

En este capítulo, presentamos una semántica no estándar para las funciones definidas recursivamente. Esta semántica está basada precisamente en la caracterización del menor punto fijo del lema 10 de la sección 2.2, sobre la que también se fundamenta el teorema de expresividad para estructuras aritméticas. Para poder expresar esta caracterización en "primer orden", como se ha visto en las estructuras aritméticas, es necesario poder hablar de derivaciones (en el sentido de la definición 8 de la sección 2.2) en el "lenguaje de primer orden".

En la primera sección, siguiendo la linea de Andréka, Németi y Sain [ANS 82] introducimos un tipo de estructuras de tres géneros: uno para los datos, otro para los números internos y otro para las secuencias internas de datos, indexadas por números internos. A estas estructuras las llamamos pseudoaritméticas, por su evidente relación con las aritméticas. De hecho, las estructuras pseudoaritméticas estándar; es decir, aquellas en las que los números y las secuencias internas son los naturales y las secuencias finitas de datos respectivamente, son una variante de tres géneros de las estructuras aritméticas.

En la segunda sección, tomamos estas estructuras como modelos de una Lógica No estándar para Funciones Recursivas NLLRF, que aunque está inspirada en LRF, admite modelos no estándar y tiene un cálculo finitario correcto y completo. De hecho, NLRF es una lógica de primer orden de tres géneros, puesto que su semántica está definida mediante una traducción sintáctica a primer orden. Además, la semántica de NNLRF está bien definida sin necesidad de exigir que los operadores sean ccontinuos o monótonos.

Aunque sobre estructuras pseudoaritméticas estándar la semántica de NLRFc coincide con la estándar, sin embargo, NLRF admite modelos demasiados libres y se usa principalmente como marco para definir lógicas más refinadas con menos modelos pero más razonables.

En la tercera sección, formulamos un conjunto de axiomas de "primer orden", llamados axiomas de admisibilidad, y definimos la lógica ALRF como una restricción de NLRF a modelos admisibles. Demostramos que la semántica de ALRFc puede ser caracterizada en términos de menor punto fijo definible y la de ALRF$_M$ en términos de "casi" menor punto fijo definible. Y presentamos un cálculo finitario correcto y completo para ALRF. En contraste con el cálculo para NLRF de la sección 2 y con otros cálculos conocidos para lógicas de programas no estándar, nuestro cálculo consta de axiomas y reglas orientadas a la sintaxis. Todos estos resultados están basados en sus equivalentes para LRF sobre estructuras aritméticas. El conjunto de axiomas de admisibilidad es decidible, y se basa en las condiciones necesarias para poder demostrar que la fórmula de primer orden, que define la semántica de una expresión funcional ($\mu X.G$), expresa el menor punto fijo definible de su operador asociado.

Finalmente, en la sección 4 mostramos una técnica para transferir derivaciones del cálculo para ALRF al cálculo aritmético para la lógica estándar LRF.

1.- ESTRUCTURAS PSEUDOARITMETICAS: UNA LOGICA DE PRIMER ORDEN EXTENDIDA CON TRES GENEROS

En esta sección, introducimos las estructuras pseudoaritméticas, siguiendo la idea del trabajo de Andréka, Németi y Sain [ANS 82]. Estas son estructuras de tres géneros basadas en un género d para los datos, un género n para los números internos y un género s para secuencias internas de datos, indexadas por números internos. Además tienen un símbolo de función distinguido para acceder a las componentes de las secuencias. Una subclase de estas estructuras es la que hemos denominado estructuras aritméticas heterogéneas, formadas intuitivamente por las estructuras pseudoaritméticas estándar. También presentamos la lógica de primer orden extendida heterogénea sobre estructuras pseudoaritméticas NEL1, que es una adaptación sintáctica de la lógica EL1 a estas estructuras.

1- DEFINICION *(Signaturas y Estructuras Pseudoaritméticas)*

Sea Σ una signatura de primer orden, en el sentido de la definición 1 de la sección 2.1:

(a) La signatura pseudoaritmética Σpa asociada a Σ es una signatura de tres géneros cuyos nombres son n, d y s, con los siguientes símbolos:

(a.1) Un símbolo de función $f:d^n \rightarrow d$ para todo símbolo de función n-ario $f \in \Sigma_{fn}$.
(a.2) Un símbolo de predicado $p:d^n$ para todo símbolo de predicado n-ario $p \in \Sigma_{pd}$.
(a.3) Constantes (i.e. símbolos de funciones 0-arias) $0,1:n$.
(a.4) Un símbolo de función $+:nxn \rightarrow n$.
(a.5) Un símbolo de función $comp:nxs \rightarrow d$.

Asumimos que los símbolos 0,1,+ y comp no pueden aparecer en la signatura Σ.

(b) Una Σ-estructura pseudoaritmética $\mathfrak{M}$ es una estructura de tres géneros de signatura Σpa. Así, podemos imaginar que $\mathfrak{M}$ está compuesta por:

$$\mathfrak{M}=<\mathfrak{N}; \mathfrak{D}; S; comp^{\mathfrak{M}}>$$

donde $\mathfrak{N}$ es una Λ-estructura de un género que interpreta a 0, 1 y + (donde Λ={0,1,+}), $\mathfrak{D}$ es una Σ-estructura de un género, S es un conjunto no vacío, y $comp^{\mathfrak{M}}: N \times S \rightarrow D$, donde N y D son los domínios de $\mathfrak{N}$ y $\mathfrak{D}$ respectivamente. ■

Ahora vamos a destacar una subclase muy importante de estructuras pseudoaritméticas. Estas son una variante de tres géneros de las estructuras aritméticas del capítulo anterior. Por tanto, esta clase de estructuras no es axiomatizable en lógica de primer orden. Llamamos a estas estructuras, aritméticas heterogéneas para distinguirlas de las aritméticas de un género.

2- DEFINICION

Una Σ-estructura pseudoaritmética $\mathfrak{M}$ se dice aritmética heterogenea si verifica las siguientes tres condiciones:

(i) $\mathfrak{N}$ es, salvo isomorfismo, el modelo estándar $<N,0,1,+>$ de la Aritmética de Presburger

(ii) S es el conjunto D^* de todas las secuencias finitas no vacias de elementos de D

(iii) Para $i \in N$ y $s=<x_1,\ldots,x_k> \in D^*$, $comp^{\mathfrak{M}}$ se comporta como sigue:

$$comp^{\mathfrak{M}}(i,<x_0,\ldots,x_k>) = \begin{cases} x_i & \text{if } i \leq k \\ x_k & \text{if } i>k \end{cases}$$

Obsérvese que una estructura aritmética heterogenea $\mathfrak{M}$ está univocamente determinada, salvo isomorfismo, por su parte $\mathfrak{D}$ de género d. Por esta razón hablaremos de la estructura aritmética heterogénea asociada a $\mathfrak{D}$, y la denotamos por $\mathfrak{M}_{\mathfrak{D}}$. ■

Ahora, adaptamos apropiadamente la lógica de primer orden extendida del cápitulo 1 a estructuras pseudoaritméticas, que denotaremos por NEL1. NEL1 no es más que una variante de tres géneros de EL1.

3- DEFINICION *(Sintaxis)*

Sea FVAR el conjunto de variables de función de la definición 1 de la sección 2.1. Fijamos tres conjuntos numerables VAR_σ, a cuyos elementos los nombramos por $x{:}\sigma, y{:}\sigma, z{:}\sigma$, de variables de individuo de género σ, para σ=n,d,s. Denotamos por $VAR = VAR_n \cup VAR_d \cup VAR_s$. Σpa-términos de primer orden de género σ ($t \in NTER_{\Sigma,\sigma}$) Σpa-fórmulas de primer orden extendido ($\varphi \in NEFOR_\Sigma$) son términos y fórmulas de tres géneros definidos como sigue:

(a) $NTER_{\Sigma,n}$

$t ::= x{:}n \mid 0 \mid 1 \mid (t_1 + t_2)$

(b) $NTER_{\Sigma,d}$

$t ::= x{:}d \mid f(t_1,\ldots,t_n)$ donde $f{:}d^n \to d \in \Sigma_{fn}$ y $t_i \in NTER_{\Sigma,d}$
$\mid comp(t', x{:}s)$ donde $t' \in NTER_{\Sigma,n}$

(c) $NTER_{\Sigma,s}$

$t ::= x{:}s$ (i.e., solo variables)

Denotamos $NTER_\Sigma = NTER_{\Sigma,n} \cup NTER_{\Sigma,d} \cup NTER_{\Sigma,s}$

(d) $NEFOR_\Sigma$

$E ::= (t_1 = t_2)$ donde $t_1, t_2 \in NTER_{\Sigma,\sigma}$
$\mid p(t_1,\ldots,t_n)$ donde $p{:}d^n \in \Sigma_{pd}$ y $t_i \in NTER_{\Sigma,d}$
$\mid X(t_1,\ldots,t_n) \ni t$ donde $t_i, t \in NTER_{\Sigma,d}$
$\mid \neg E_1$
$\mid (E_1 \vee E_2)$
$\mid \exists x{:}\sigma E$ ■

La semántica es análoga a la definida para EL1 en las definicones 2 y 3 de la sección 2.1. Las diferencias técnicas más importantes se refieren a las valoraciones, estas deben definirse como funciones que preservan los géneros del conjunto VAR al conjunto $N \cup D \cup S$; es decir, $\delta(x{:}n) \in N$, $\delta(x{:}d) \in D$ y $\delta(x{:}s) \in S$. Como solo hay variables funcionales de un género, los contextos, asocian a cada variable funcional de aridad n una función del conjunto $\mathfrak{N}\mathfrak{F}_D^{(n)}$. Omitimos los detalles.

Los resultados ya formulados en la sección 1 del cápitulo 2 para la lógica de primer orden extendida, se verifican también para la lógica de primer orden extendida sobre estructuras pseudoaritméticas. Puede probarse de forma análoga, puesto que estos son solo una variante de tres géneros de los correspondientes resultados para EL_1:

4- TEOREMA

La lógica de primer orden extendida sobre estructuras pseudoaritméticas (NEL_1) tiene un cálculo finitario correcto y completo.

Demostración

Basta con adaptar el dado para EL_1 a tres géneros, pero como ya ocurría para EL_1, en el teorema 5 de la siguiente sección se presenta un cálculo finitario correcto y completo para una lógica que incluye como fragmento a NEL_1. ■

5- TEOREMA

La lógica de primer orden extendida sobre estructuras pseudoaritméticas satisface el teorema de compacidad y el teorema de Löwenheim-Skolem. ■

En el resto del trabajo, nos referiremos a la lógica de primer orden extendida sobre estructuras pseudoaritméticas, simplemente por lógica de primer orden extendida.

2.- LA LOGICA NO ESTANDAR NLRF

Como hemos dicho en la introducción de este capítulo, NLRF es una lógica no estándar para funciones recursivas, cuyas fórmulas se interpretan sobre estructuras pseudoaritméticas. El proposito de NLRF es proveer de un marco adecuado donde definir otras lógicas no estándar para funciones recursivas, restringiendo los modelos mediante axiomas lógicos que imponen propiedades deseadas sobre los modelos. Puesto que los modelos de NLRF son totalmente libres, y sobre muchos de ellos la semántica de los programas no es razonable. Aunque sobre la clase de las estructuras aritméticas heterogéneas NLRFc es equivalente a LRFc.

Sintácticamente, NLRF es una lógica de tres géneros que incluye a la lógica de primer orden extendida (NEL1) como fragmento propio. Y la semántica la definimos mediante una traducción sintáctica a NEL1, de manera que NLRF no es más expresiva que NEL1.

1- DEFINICION *(Sintaxis)*

Sea Σ una signatura de primer orden. Las Σ_{pa}-expresiones funcionales de aridad $n \geq 0$ $(F, G \in NFEXP_{\Sigma}^{(n)})$, Σ_{pa}-expresiones de género d $(M, N \in NEXP_{\Sigma})$ y Σ_{pa}-fórmulas $(\varphi, \psi \in NFFOR_{\Sigma})$ de NLRF se definen como sigue:

(a) $NFEXP_{\Sigma}^{(n)}$:

$F ::= f$ donde $f: d^n \rightarrow d \in \Sigma_{fn}$

| X donde $X \in FVAR$

| $\lambda x_1 : d \ldots x_n : d.M$ donde $M \in NEXP_{\Sigma}$

| $\mu X.G$

(b) $NEXP_{\Sigma}$:

$M ::= t$ donde $t \in NTER_{\Sigma, d}$

| $F(M_1, \ldots, M_n)$ donde $F \in NFEXP_{\Sigma}^{(n)}$, $M_1, \ldots M_n \in NEXP_{\Sigma}$

| $(M \mathcal{N} N)$ donde $M, N \in NEXP_{\Sigma}$

| $(\varphi \rightarrow N)$ donde $\varphi \in NFFOR_{\Sigma}$, $N \in NEXP_{\Sigma}$

| $\varepsilon x : d \; \varphi$ donde $\varphi \in NFFOR_{\Sigma}$

(c) $NFFOR_{\Sigma}$:

$\varphi ::= (t_1 = t_2)$ donde $t_1, t_2 \in NTER_{\Sigma,\sigma}$

$\mid p(M_1,\ldots,M_n)$ donde $p:d \in \Sigma_{pd}$

$\mid M \ni t$ donde $M \in NEXP_{\Sigma}$ y $t \in NTER_{\Sigma,d}$

$\mid \neg\psi \mid (\varphi_1 \vee \varphi_2) \mid \exists x:\sigma\ \psi$ ■

Las restricciones sintácticas NLRFs, NLRFM y NLRFc se adaptan directamente desde LRF de las definiciones 3 y 4 de la sección 2.3, donde las restricciones sobre subfórmulas del tipo $\exists x:\sigma\varphi$ deben entenderse sobre cualquier género σ.

Como hicimos para LRF sobre estructuras aritméticas (sección 3.2), podemos introducir dos símbolos definidos de predicado $<, \leq : dxd$ tal que, para todo par de términos t_1, t_2 de género n:

$t_1 < t_2$ abrevia $\exists x:n\ (\ (t_1 + x:n) + 1 = t_2)$

(donde $x:n$ no aparece ni en t_1 ni en t_2)

y

$t_1 \leq t_2$ abrevia $t_1 < t_2 \vee t_1 = t_2$

Notese que estas abreviaturas tienen también sentido en la lógica de primer orden extendido (NEL_1).

También, usaremos a veces $u:s[t]$ como abreviatura de $comp(t,u:s)$ para todo término t de género n.

Además, los símbolos definidos $< >$, $[\ ]$ y $\downarrow$ previamente definidos para LRF en la definición 9 de la sección 2.3, tienen el mismo significado para NLRF.

Semánticamente, NLRF colapsará con la lógica de primer orden extendida. Más exactamente: la semántica de NLRF será definida por medio de una traducción sintáctica a fórmulas de primer orden extendido, con la misma idea que el teorema de expresividad del capítulo anterior para estructuras aritméticas. Para una signatura Σ, la traducción asocia:

(a) A cada $\varphi \in NFFOR_{\Sigma}$, E_{φ}

(b) A cada $M \in NEXP_{\Sigma}$, $E_M(y:d)$, donde $y:d$ es una variable arbitraria que no aparece en M

(c) A cada $F \in NFEXP_{\Sigma}^{(n)}$, $E_F(x_1:d,\ldots,x_n:d,y:d)$, donde $x_i:d, y:d$ son variables arbitrarias distintas entre sí que no aparecen en F.

Como ya hemos dicho, la intuición que hay detras de la traducción es la del teorema de expresividad del capítulo anterior; es decir:

(a) E_{φ} es equivalente a φ.

(b) $E_M(y:d)$ dice: "$y:d$ es un posible valor de M"

(c) $E_F(x_1:d,\ldots,x_n:d,y:d)$ dice: "$y:d$ es un posible valor de $F(x_1:d,\ldots,x_n:d)$"

Ahora precisamos la traducción, mediante una definición mutuamente recursiva sobre la función de complejidad $|.|$ (análoga a la definida en la sección 3.1 para LRF) para fórmulas, expresiones y expresiones funcionales.

(a) Traslación de fórmulas

$\varphi = t_1 = t_2$:	$E_{\varphi} = t_1 = t_2$
$\varphi = p(M_1,\ldots,M_n)$:	$E_{\varphi} = \exists \vec{y}:d\,(\bigwedge_{1\le j\le n} E_{M_j}(y_j:d) \wedge p\,(\vec{y}:d)\,)$
$\varphi = M \ni t$:	$E_{\varphi} = E_M(y:d)[t/y:d]$
$\varphi = \neg\psi$:	$E_{\varphi} = \neg E_{\psi}$
$\varphi = \psi_1 \vee \psi_2$:	$E_{\varphi} = E_{\psi_1} \vee E_{\psi_2}$
$\varphi = \exists z:\sigma\,\psi$:	$E_{\varphi} = \exists z:\sigma\, E_{\psi}$

(b) Traslación de expresiones

$M = t \in NTER_{\Sigma,d}$:	$E_M(y:d) = t = y:d$
$M = F(M_1,..M_n)$:	$E_M(y:d) = \exists y:d\,(\bigwedge_{1\le j\le n} E_{M_j}(y_j:d) \wedge E_F(\vec{y}:d, y:d))$
$M = M_1 \cup M_2$:	$E_M(y:d) = E_{M_1}(y:d) \vee E_{M_2}(y:d)$
$M = \varphi \rightarrow N$:	$E_M(y:d) = E_{\varphi} \wedge E_N(y:d)$
$M = c\ x:d\ \varphi$:	$E_M(y:d) = E_{\varphi}[y:d/x:d]$

(c) Traslación de expresiones funcionales

$F=f$: $\quad E_F(\vec{x}:d, y:d) = f(\vec{x}:d) = y:d$

$F=X$: $\quad E_F(\vec{x}:d, y:d) = X(\vec{x}:d) \ni y:d$

$F=\lambda z:d.M$: $\quad E_F(\vec{x}:d, y:d) = E_M(y:d)[\vec{x}:d/\vec{z}:d])$

$F=\mu X.G$: $\quad E_F(\vec{x}:d, y:d) = \exists k:n\ BE_F(k:n, \vec{x}:d, y:d)$

$BE_F(k:n, \vec{x}:d, y:d)$ se construye análogamente a como se hizo en el teorema de expresividad para estructuras aritméticas, salvo que ahora todas las referencias que se hacían a números naturales y códigos de secuencias se transfieren a elementos de género n y s respectivamente. Es decir, su significado es: "$F(\vec{x}:d)$ acepta el valor y producido por una derivación interna de longitud interna k". A continuación, volvemos a desarrollar la definicón formal de BE_F en varias etapas, intentando explicar las diferencias de concepto con su significado estándar sobre estructuras aritméticas.

(1) $BE_F(k:n, \vec{x}:d, y:d) =$

$$\exists \vec{u}:s\ \exists v:s(CD_{G,X}(k:n, \vec{u}:s, v:s) \wedge CP(k:n, \vec{u}:s, v:s, \vec{x}:d, y:d)$$

Dice: "Las secuencias internas $\vec{u}, v$ codifican una derivación de longitud k c.r.al operador asociado a G,X y los k-ésimos mlembros de estas secuencias muestran que esta derivación produce la tupla $\langle \vec{x}, y \rangle$". Obsérvese que el caracter interno de las secuancias y de la longitud cambian el significado de finitud de estas secuencias, ya que k denota un número internamente finito pero no necesariamente finito.

(2) $CP(k:n, \vec{u}:s, v:s, \vec{x}:d, y:d) =$

$$(\bigwedge_{1 \le j \le n} u_j:s[k:n] = x_j:d) \wedge v:s[k:n] = y:d$$

Dice: "Las k-ésimas componentes de las secuencias $\vec{u}, v$ codifican la tupla $\langle \vec{x}, y \rangle$".

(3) $CD_{G,X}(k:n, \vec{u}:s, v:s) =$

$$\forall l:n(0 \le l:n \wedge l:n \le k:n \rightarrow PV_{G,X}(l:n, \vec{u}:s, v:s)).$$

donde

$PV_{G,X}(l:n, \vec{u}:s, v:s) =$

$$E_{G[\lambda \vec{x}:d.\, \varepsilon y:d GR(l:n, \vec{u}:s, v:s, \vec{x}:d, y:d)/X, \vec{u}:s[l]/\vec{x}, v:s[l]/y]}(\vec{x}:d, y:d)$$

y

$GR(l:n, \vec{u}:s, v:s, \vec{x}:d, y:d) =$

$$\exists m:n(m:n<l:n \wedge (\bigwedge_{1 \leq j \leq n} u_j:s[m:n] = x_j:d) \wedge v:s[m:n] = y:d)$$

Dice: "Para $0 \leq l \leq k$, la l-ésima tupla $\langle \vec{u}[l], v[l] \rangle$ codificada por las secuencias, satisface que v[l] es un posible valor de la evaluación de G cuando las variables de individuo $\vec{x}$ son interpretadas como $\vec{u}[l]$ y la variable de función X es interpretada como la función internamente finita cuyo grafo está dado por las tuplas $\langle \vec{u}[m], \vec{v}[m] \rangle$, $m<l$ codificadas por las secuencias". Obsérvese que sobre las funciones codificadas por secuencias internas solo se puede asegurar que son internamente finitas pero no necesariamente finitas.

Como en las estructuras pseudoaritméticas los números y las secuencias son parte de la estructura, el significado de esta fórmula no tiene nada que ver con el menor punto fijo del operador asociado, aunque evidentemente la intención es que su significado esté muy relacionado con este concepto. Pero para poder dar un significado adecuado a esta fórmula fuera de las estructuras aritméticas, tendremos que exigir que las estructuras pseudoaritméticas cumplan ciertas restricciones.

Entre estas restricciones se encuentran condiciones mínimas sobre el comportamiento de los números y las secuencias internas y el principio de inducción sobre NEL_1, que junto con el concepto de función definible (necesario para poder aplicar el principio de inducción), nos permitirá demostrar en la siguiente sección que el significado de esta fórmula sobre estructuras "admisibles" es el menor punto fijo definible del operador asociado (que evidentemente es definible).

El siguiente resultado técnico será usado en la siguiente sección.

2- LEMA

Sea Θ una Σ-expresión funcional, expresión o fórmula de NLRF. Si Θ es positiva (resp. negativa, existencial, universal) en una variable de función Y dada, entonces la Σ-fórmula de primer orden extendido de su translación E_Θ tiene la misma propiedad.

Demostración

Por inducción simultanea sobre la función de complejidad $|.|$ para las expresiones funcionales, expresiones y fórmulas. Consideremos, por ejemplo, el caso $F=\mu X.G$, donde X es diferente de Y. Si asumimos que G es positiva en Y, entonces $E_G(\vec{x}:d,y:d)$ es también positiva en Y, por hipótesis de indución. Inspeccionando $E_F(\vec{x}:d,y:d)$, se comprueba que todas las apariciones libres de Y en $E_F(x:d,y:d)$ estan en la subfórmula $PV_{G,X}(l:n,\vec{u}:s,v:s)$ y corresponden a las apariciones de Y en $E_G(\vec{x}:d,y:d)$, afectadas por dos símbolos de negación adicionales. (Esto es fácil de ver reduciendo $\forall$ y $\rightarrow$ a $\exists,\neg$ y $\vee$ en $CD_{G,X}(k:n,\vec{u}:s,v:s)$). Entonces es claro que también $E_F(\vec{x}:d,y:d)$ es positiva en Y. Los demás casos se comprueban similarmente. ■

Ahora, estamos en posición de definir la semántica de NLRF.

3- DEFINICION *(Semántica Denotacional)*

Sea $\mathfrak{M} = \langle \mathfrak{N}; \mathfrak{D}; S; comp^{\mathfrak{M}} \rangle$ una Σ-estructura pseudoaritmética. Sean Δ,δ un contexto y una valoración sobre $\mathfrak{M}$ respectivamente.

Las denotaciones de las fórmulas, expresiones y expresiones funcionales para NLRF son definidas via la traducción sintáctica anterior como sigue:

(a) Fórmulas

$$\varphi^{\mathfrak{M}}(\Delta,\delta) = E_\varphi^{\mathfrak{M}}(\Delta,\delta)$$

(b) Expresiones

$$M^{\mathfrak{M}}(\Delta,\delta) = \{y/\ E_M(y:d)^{\mathfrak{M}}(\Delta,\delta[y/y:d])=true\}$$

(c) Expresiones funcionales

$$grafo(F^{\mathfrak{M}}(\Delta,\delta)) = \{<\vec{x},y>\in D^{n+1}/E_F(\vec{x}:d,y:d)^{\mathfrak{M}}(\Delta,\delta[\vec{x}/\vec{x}:d,y/y:d]) = true\}$$ ■

Es claro que sobre las estructuras aritméticas heterogéneas, la semántica de NLRFc es equivalente a la semántica estándar de LRFc (salvo las diferencias obvias de sintaxis). En otras palabras: la traducción sintáctica expresa el significado estándar cuando las fórmulas traducidas se interpretan en estructuras aritméticas heterogéneas, lo que desde luego no ocurre con NLRFH. Sin embargo, como ya hemos dicho, sobre algunas estructuras pseudoaritméticas, la semántica de NLRF se comportará arbitrariamente, puesto que los individuos de género n y s no tendrán nada que ver con los verdaderos números y secuencias.

De hecho, no se pretende que NLRF sea la lógica no estándar para programas funcionales, sino que sirva de marco para definir estas lógicas, mediante axiomas lógicos designados a imponer propiedades buenas sobre los modelos, como se hizó para NDL (una lógica no estándar para programas imperativos, expuesto en el capítulo 1). Para finalizar esta sección, presentamos dos resultados que muestran que NLRF es una base solida para este propósito. El primero es análogo al resultado similar de Andréka, Németi y Sain [ANS 82] para NDL; es decir, que NLRF tiene un cálculo finitario correcto y completo.

Para obtener este cálculo, segun la definición de la semántica de NLRF, es suficiente fijar un cálculo correcto y completo para la lógica de primer orden extendida y aumentarlo con nuevos axiomas lógicos para poder demostrar formalmente todas las equivalencias $\varphi\leftrightarrow E_\varphi$ para $\varphi\in NFFOR_\Sigma$. De hecho, podríamos tomar el conjunto decidible de estas equivalencias como nuevos axiomas. Acercandonos más a un cálculo orientado a la sintaxis escogemos el siguiente cálculo al que denominamos π, donde C_d, C_n, y C_s son tres nuevos conjuntos numerables de constantes auxiliares disjuntos y en la sustitución $\varphi[t/x:\sigma]$ se entiende que t es un término de género σ.

Axiomas y reglas de primer orden

(SP) $\Gamma \vdash \varphi$ si $\varphi \in \Gamma$

(FS) $\dfrac{\Gamma \vdash \varphi}{\Gamma, \Gamma' \vdash \varphi}$

(RC) $\dfrac{\Gamma \vdash \psi \quad \Gamma, \psi \vdash \varphi}{\Gamma \vdash \varphi}$

(v-C) $\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi}$ $\qquad$ $\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \psi \vee \varphi}$

(v-A) $\dfrac{\Gamma, \varphi \vdash \phi \quad \Gamma, \psi \vdash \phi}{\Gamma, \varphi \vee \psi \vdash \phi}$

(¬-C) $\dfrac{\Gamma, \varphi \vdash ff}{\Gamma \vdash \neg \varphi}$ $\qquad$ (¬-A) $\dfrac{\Gamma, \neg \varphi \vdash \varphi}{\Gamma, \neg \varphi \vdash ff}$

(ffA) $\Gamma, ff \vdash \varphi$ $\qquad$ (DN-A) $\dfrac{\Gamma, \varphi \vdash \psi}{\Gamma, \neg\neg \varphi \vdash \psi}$

(∃-C) $\dfrac{\Gamma \vdash \varphi[t/x:\sigma]}{\Gamma \vdash \exists x:\sigma \varphi}$ $\qquad$ (∃-A) $\dfrac{\Gamma, \varphi[c/x:\sigma] \vdash \psi}{\Gamma, \exists x:\sigma \varphi \vdash \psi}$ siendo $c \in C_\sigma$ nueva

(ID) $\Gamma \vdash t=t$ $\qquad$ (SUS) $\dfrac{\Gamma \vdash \varphi[t/x:\sigma]}{\Gamma, t=t' \vdash \varphi[t'/x:\sigma]}$

Reglas para la descomposición de las expresiones

(∋-A) $\dfrac{\Gamma,\ t=t' \vdash \varphi}{\Gamma,\ t \ni t' \vdash \varphi}$ $\qquad$ (∋¬A) $\dfrac{\Gamma,\ \neg t=t' \vdash \varphi}{\Gamma,\ \neg t \ni t' \vdash \varphi}$

(PD-A) $$\dfrac{\Gamma,\ \exists y_1:d \ldots \exists y_n:d (M_1 \ni y_1:d \wedge \ldots \wedge M_n \ni y_n:d \wedge p(y_1:d,\ldots,y_n:d)) \vdash \varphi}{\Gamma,\ p(M_1,\ldots,M_n) \vdash \varphi}$$

$$(PD\neg A)\quad \frac{\Gamma,\ \neg\exists y_1{:}d \ldots \exists y_n{:}d(M_1 \ni y_1{:}d \wedge \ldots \wedge M_n \ni y_n{:}d \wedge p(y_1{:}d,\ldots,y_n{:}d)) \vdash \varphi}{\Gamma,\ \neg p(M_1,\ldots,M_n) \vdash \varphi}$$

donde las variables $y_i{:}d$ no aparecen ni en $M_1,\ldots,M_{n-1}$ ni en M_n y son distintas entre sí.

$$(\cup\text{-}A)\quad \frac{\Gamma,\ M \ni t \vee N \ni t \vdash \varphi}{\Gamma,\ (M \cup N) \ni t \vdash \varphi} \qquad (\cup\neg A)\quad \frac{\Gamma,\ \neg(M \ni t \vee N \ni t) \vdash \varphi}{\Gamma,\ \neg(M \cup N) \ni t \vdash \varphi}$$

$$(\rightarrow\text{-}A)\quad \frac{\Gamma,\ \psi \wedge M \ni t \vdash \varphi}{\Gamma,\ (\psi \rightarrow M) \ni t \vdash \varphi} \qquad (\rightarrow\neg A)\quad \frac{\Gamma,\ \neg(\psi \wedge M \ni t) \vdash \varphi}{\Gamma,\ \neg(\varphi \rightarrow M) \ni t \vdash \varphi}$$

$$(\varepsilon\text{-}A)\quad \frac{\Gamma,\ \psi[t/x{:}d] \vdash \varphi}{\Gamma,\ (\varepsilon x{:}d\psi) \ni t \vdash \varphi} \qquad (\varepsilon\neg A)\quad \frac{\Gamma,\ \neg\psi[t/x{:}d] \vdash \varphi}{\Gamma,\ \neg(\varepsilon x{:}d\psi) \ni t \vdash \varphi}$$

$$(AP\text{-}A)\quad \frac{\Gamma,\ \exists y_1{:}d \ldots \exists y_n{:}d(M_1 \ni y_1{:}d \wedge \ldots \wedge M_n \ni y_n{:}d \wedge F(y_1{:}d,\ldots,y_n{:}d)) \ni t \vdash \varphi}{\Gamma,\ F(M_1,\ldots,M_n) \ni t \vdash \varphi}$$

$$(AP\neg A)\quad \frac{\Gamma,\ \neg\exists y_1{:}d \ldots \exists y_n{:}d(M_1 \ni y_1{:}d \wedge \ldots \wedge M_n \ni y_n{:}d \wedge F(y_1{:}d,\ldots,y_n{:}d)) \ni t \vdash \varphi}{\Gamma,\ \neg F(M_1,\ldots,M_n) \ni t \vdash \varphi}$$

donde las variables $y_i{:}d$ no aparecen ni en $M_1,\ldots,M_n$ ni en t y son distintas entre sí.

$$(\lambda\text{-}A)\quad \frac{\Gamma,\ M[t_1/x_1{:}d,\ldots,t_n/x_n{:}d] \ni t \vdash \varphi}{\Gamma,\ (\lambda x_1{:}d,\ldots,x_n{:}d.M)(t_1,\ldots,t_n) \ni t \vdash \varphi}$$

$$(\lambda\neg A)\quad \frac{\Gamma,\ \neg M[t_1/x_1{:}d,\ldots,t_n/x_n{:}d]\ni t \vdash \varphi}{\Gamma,\ \neg(\lambda x_1{:}d,\ldots,x_n{:}d.M)(t_1,\ldots,t_n)\ni t \vdash \varphi}$$

$$(\mu\text{-}A)\quad \frac{\Gamma,\ \exists k{:}n\ BE_{\mu X.G}(k{:}n,\vec{x}{:}d,y{:}d)[t_1/x_1{:}d,\ldots,t_n/x_n{:}d,t/y{:}d] \vdash \varphi}{\Gamma,\ (\mu X.G)(t_1,\ldots,t_n)\ni t \vdash \varphi}$$

$$(\mu\neg A)\quad \frac{\Gamma,\ \neg\ \exists k{:}n\ BE_{\mu X.G}(k{:}n,\vec{x}{:}d,y{:}d)[t_1/x_1{:}d,\ldots,t_n/x_n{:}d,t/y{:}d] \vdash \varphi}{\Gamma,\ \neg\ (\mu X.G)(t_1,\ldots,t_n)\ni t \vdash \varphi}$$

Obsérvese la similaridad entre este cálculo y el infinitario dado en el cápitulo 2 para LRFs. Las diferncias fundamentales son que: (μ-A) antes era una regla infinitaria y ($\mu\neg$A) añadía un conjunto numerable de fórmulas al antecedente. Además este cálculo es completo y correcto para todo NLRF, sin necesidad de restricciones sintácticas.

Evidentemente los axiomas y reglas de primer orden de π forman un cálculo correcto y completo para NEL$_1$, al que hemos añadido una regla composicional para cada una de las posibles estructuras de una expresión. Ahora las reglas (μ-A) y ($\neg\mu$C) no hacen referencia a la semántica operacional del operador μ. Incidentalmente, estas dos reglas son también el ingrediente menos orientado a la sintaxis del cálculo, puesto que $BE_{\mu X.G}$ no hace referencia a subexpresiones de $\mu X.G$ de forma suficientemente clara y sencilla. Esto será mejorado en la siguiente sección, sustituyendolas por reglas análogas a las del cálculo aritmético.

La relación de deducción formal en este cálculo es la clásica de primer orden para cálculos de secuencias:

4- DEFINICION

Dados un conjunto de fórmulas Φ (a lo más numerable) y una fórmula φ, φ es demostrable a partir de Φ en el cálculo n, denotado por $\Phi \vdash_n \varphi$ o simplemente por $\Phi \vdash \varphi$, si existe un árbol de deducción finito para $\Gamma \vdash \varphi$ con $\Gamma \subseteq \Phi$ finito, es decir un árbol de deducción cuya raiz está etiquetada por $\Gamma \vdash \varphi$, donde ahora árbol de deducción se define de forma análoga a la definición 9 de la sección 3.2, pero finitamente ramificado. ■

5- TEOREMA *(corrección y completitud)*

Para todo conjunto de fórmulas $\Phi \cup \{\varphi\}$ de NLRF:

$$\Phi \vdash \varphi \iff \Phi \models \varphi \ .$$

Demostración

CORRECION: Por inducción sobre la profundidad N del árbol de deducción, demostramos que la raíz de cualquier árbol de deducción es correcta. Decimos que una secuencia $\Gamma \vdash \varphi$ es correcta si $\Gamma \models \varphi$; es decir si $\mathfrak{M} \models \Gamma$ implica $\mathfrak{M} \models \varphi$ para cualquier estructura pseudoaritmética $\mathfrak{M}$.

N=0: Hay que comprobar que todas las instancias de los axiomas $\Gamma \vdash \varphi$ del cálculo son correctas; es decir, $\Gamma \models \varphi$. Todos los axiomas son de primer orden y su corrección puede encontrarse por ejemplo en [EFT 84].

N>0: Hay que comprobar que todas las instancias de las reglas del cálculo verifican que la conclusión es correcta si lo son las premisas. Lo cual salvo para las reglas del operador μ es análogo al cálculo infinitario, y para las dos reglas del operador μ es evidente a partir de la semántica dada ahora para este operador.

COMPLETITUD: Podemos demostrarlo con la misma técnica seguida para el cálculo infinitario, por medio de conjuntos de Hintikka y Tableaux. El proceso sería análogo cambiando todas las condiciones dadas para el operador μ. En este caso desaparecerían los tipos de fórmulas (α^*) y (β^*), quedando solo cinco categorías de fórmulas, como en primer orden. Las

fórmulas $(\mu X.G)(\vec{t})\ni t$ y $\neg(\mu X.G)(\vec{t})\ni t$ ahora son de tipo (α) con un solo constituyente: $\exists k{:}n \quad BE_{\mu X.G}(k{:}n, \vec{x}{:}d, y{:}d)[\vec{t}/\vec{x}{:}d, t/y{:}d]$ y su negación respectivamente. Con lo que los tableaux pasan a ser árboles binarios con nodos etiquetados por conjuntos de $\bar{\Sigma}$-fórmulas finitos, como en primer orden.

Sin embargo, supuesta la completitud de este cálculo para NEL1 es suficiente demostrar para cualquier $\varphi\in NEXP_{\Sigma}$:

$$(a) \quad \vdash \varphi \ \leftrightarrow \ E_{\varphi}$$

Puesto que por la semántica dada para NLRF, $\Phi\vdash\varphi$ <=> $E_{\Phi}\vdash E_{\varphi}$ siendo Φ un conjunto de fórmulas de NLRF y $E_{\Phi} = \{ E_{\phi} \ / \ \phi\in\Phi \}$. Por lo tanto: $\Phi\vdash\varphi$ <=> $E_{\Phi}\vdash E_{\varphi}$ <=> $E_{\Phi}\vdash E_{\varphi}$ <=> $\Phi\vdash\varphi$, supuesto (a).

La demostración de (a) es trivial por inducción simultánea sobre la complejidad $|.|$ para las fórmulas, expresiones y expresiones funcionales de NLRF:

Si φ es de primer orden extendido, entonces $E_{\varphi}=\varphi$.

En otro caso, hay dos reglas para cada posible tipo de fórmula φ que directamente o previo uso de hipótesis de inducción, tienen la forma

$$\frac{\Gamma,\ E_{\varphi}\vdash\psi}{\Gamma,\ \varphi\ \vdash\psi} \qquad\qquad \frac{\Gamma,\ \neg E_{\varphi}\vdash\psi}{\Gamma\ \neg\varphi\vdash\psi}$$

que permiten junto con algunas reglas de primer orden, obtener la derivación deseada. ■

A partir de la demostración de este teorema, podemos deducir como siempre los siguientes resultados:

6- TEOREMA

NLRF satisface el teorema de Compacidad y el teorema de Löwenheim-Skolem. ■

Terminamos esta sección recalcando, que de hecho el poder de expresión de NLRF es el de la lógica de primer orden clásica, por la traducción sintáctica a NEL_1 que define la semántica.

3.- LA LOGICA ADMISIBLE ALRF

"Números" y "secuencias" internas en estructuras pseudoaritméticas pueden no tener ninguna relación con los números naturales y las secuencias finitas. Consecuentemente, la semántica de NLRF hace que los programas no tengan un sentido razonable en muchos de los posibles modelos. En esta sección, axiomatizamos una clase de estas estructuras, llamadas estructuras pseudoaritméticas admisibles, en lógica de primer orden extendida, y definimos la Lógica Admisible para Funciones Recursivas (ALRF) como la restricción de NLRF a modelos admisibles. Demostramos que la semántica de ALRF$_K$ se caracteriza en términos de casi menor punto fijo definible y la de ALRFc tiene una caracterización natural en términos de menor punto fijo definible. El conjunto de axiomas admisibles está basado en exigir las propiedades necesarias para poder demostrar estas caracterizaciones, pero no nos hemos preocupado de que este conjunto sea el más débil que permita esta caracterización. También probamos la existencia de un cálculo finitario correcto y completo para ALRF, análogo al dado para LRF sobre estructuras aritméticas. Acabamos esta sección comentando brevemente la capacidad de deducción de ALRF.

En primer lugar, presentamos el conjunto de axiomas de admisibilidad, que está motivado en la demostración del lema 10 de la sección 1.2. En particular es necesario poder aplicar el principio de inducción sobre los números internos y concatenar secuencias internas. A continuación tenemos que demostrar resultados análogos a estos, pero referidos a derivaciones internas a la estructura, segun se vió en la sección 2 de este capítulo.

1- DEFINICION *(Axiomas de Admisibilidad, Estructuras Admisibles)*

Para una signatura Σ, el conjunto A_Σ de los axiomas de admisibilidad consiste de las Σ-fórmulas de primer orden extendido siguientes:

(a) Axiomas básicos para los números

$\forall x{:}n \forall y{:}n\ (x{:}n+1 = y{:}n+1) \rightarrow x{:}n = y{:}n$

$\forall x{:}n\ \neg\ x{:}n+1 = 0$

$\forall x{:}n \forall y{:}n\ x{:}n = 0 \vee \exists y{:}n\ x{:}n = y{:}n+1$

$\forall x{:}n\ x{:}n+0 = x{:}n$

$\forall x{:}n \forall y{:}n\ x{:}n+(y{:}n+1) = (x{:}n+y{:}n)+1$

Es decir, la Aritmética de Presburger PA: todas las Σpa-sentencías en las cuales solo aparecen los símbolos 0, 1, + y variables cuantificadas de género n, y son ciertas en la estructura $\mathfrak{N} = \langle N, 0, 1, + \rangle$.

(b) Axioma de inducción para los números

$$ind_{x:n}(E) : \forall x{:}n(\forall y{:}n(y{:}n<x{:}n \rightarrow E[y{:}n/x{:}n]) \rightarrow E)$$
$$\text{para toda } E \in EFOR_{\Sigma pa, x:n \in VAR_n}$$

(c) Conjunto de axiomas para las secuencias

$$col_{i:n, m:n, y:d}(E) : \forall i{:}n(i{:}n \leq m{:}n \rightarrow \exists y{:}d\ E)$$
$$\rightarrow \exists w{:}s \forall i{:}n(i{:}n \leq m{:}n \rightarrow E[w{:}s[i{:}n]/y{:}d])$$

Una Σ-estructura pseudoaritmetica $\mathfrak{M}$ es llamada admisible si $\mathfrak{M} \vdash A_\Sigma$. ■

2- LEMA

Todas las estructuras admisibles son modelo de los siguientes dos axiomas:

$unit{:}\forall x{:}d \exists u{:}s\ u{:}s[0] = x{:}d$

$conc{:}\forall u{:}s \forall k{:}n \forall v{:}s \forall l{:}n \exists w{:}s(\forall i{:}n(i{:}n \leq k{:}n \rightarrow u{:}s[i{:}n] = w{:}s[i{:}n]) \wedge$
$\forall j{:}n(j{:}n \leq l{:}n \rightarrow v{:}s[j{:}n] = w{:}s[(k{:}n + j{:}n + 1]))$

Es decir: todas las secuencias de longitud uno existen, y cualquier par de secuencias finitas (internamente) se pueden concatenar.

Demostración

unit y conc se siguen en NLRF como consecuencias lógicas de instancias particulares de $col_{i:n,m:n,y:d}(E)$, usando los otros axiomas de admisibilidad.

Para unit, E puede cogerse como

$$y:d = x:d$$

Para conc, es suficiente tomar como E la fórmula

$$(i:n \leq k:n \wedge y:d = u:s[i:n]) \vee$$
$$\exists j:n\ (\ j:n \leq l:n \wedge i:n = k:n + j:n + 1 \wedge y:d = v:s[j:n]\ ) \quad \blacksquare$$

Sin más que comprobar la validez en $\mathfrak{M}_{\mathfrak{D}}$ de los axiomas de admisibilidad, obtenemos también:

3- TEOREMA

Para una Σ-estructura $\mathfrak{D}$, la estructura aritmética heterogenea $\mathfrak{M}_{\mathfrak{D}}$ asociada a $\mathfrak{D}$ (cfr. definición 2 de la sección 1) es admisible.

Desde luego que hay estructuras admisibles no estándar, puesto que la clase de las estructuras aritméticas heterogéneas no es axiomatizable en primer orden.

Entendemos por una lógica un sistema dado por la especificación de las posibles signaturas, la sintaxis, las estructuras permitidas y las reglas semánticas para obtener denotaciones. Para nuestro propósito, no es necesaria la definición formal de este concepto que ha sido estudiado por investigadores de teoría de modelos (cfr. [BF 85]) y, más recientemente, de intuicionismo (cfr. [GB 83]). Simplemente decimos que ALRF es la lógica que tiene la misma sintaxis y semántica que NLRF, excepto que los módelos están restringidos a estructuras admisibles.

Las denotaciones semánticas, satisfactibilidad y consecuencia lógica se definen por tanto para ALRF de forma natural. Cuando trabajamos con ALRF, reemplazaremos el símbolo $\vdash$ por $\vdash_A$.

SEMANTICA DEL MENOR PUNTO FIJO DEFINIBLE

El propósito de restringir las estructuras pseudoaritméticas con los axiomas de admisibilidad, es poder caracterizar la semántica dada para las expresiones funcionales $F=\mu X.G$, justamente de forma inversa a la dada para estructuras aritméticas, en las que E_F define el menor punto fijo del operador T asociado a G,X. En estructuras admisibles la caracterización no puede ser exactamente en los mismos términos, puesto que como se ha resaltado en la sección anterior, el caracter interno de los números y las secuencias, nos obliga a hablar de T-derivaciones internas. La diferencia fundamental entre las T-derivaciones internas y estándar, es que las funciones codificadas por secuencias internas no tienen por que ser finitas, con lo que en lugar de trabajar con el operador $\bar{T}$, lo haremos con el operador análogo pero sobre funciones internamente finitas $\hat{T}$. Además, en la demostración de la caracterización semántica de F en estructuras aritméticas, es necesario aplicar el principio de inducción sobre los números. En ALRF solo podemos aplicar este principio para fórmulas de NEL_1, lo que a su vez nos obliga a restringir la caracterización del menor punto fijo a la de menor punto fijo definible. Por último, esta caracterización será posible para $ALRF_C$, y para $ALRF_H$ será en términos de casi menor punto fijo definible.

A continuación, introducimos los conceptos y resultados necesarios para este propósito.

4- DEFINICION *(Función Definible e Internamente Finita)*

(a) Sea $\mathfrak{M} = \langle \Pi; \mathfrak{D}; S; comp^{\mathfrak{M}} \rangle$ una Σ-estructura. Fijados un contexto Δ y una valoración δ sobre $\mathfrak{M}$. Una función no determinista $X \in \Pi\tilde{\mathfrak{I}}_D^{(n)}$ es llamada definible en $\mathfrak{M}$ c.r.a Δ,δ si existe una fórmula de primer orden extendido $E \in NEFOR_\Sigma$ con a lo más n+1 variables libres $x_1:d,\ldots,x_n:d,y:d$ tal que

$$grafo(X) = \{\langle x_1,\ldots,x_n,y\rangle \in D^{n+1} / E^{\mathfrak{M}}(\Delta, \delta[x_1/x_1:d,\ldots,x_n/x_n:d,y/y:d]) = true\}$$

(b) Sea $\mathfrak{M} = \langle \mathfrak{N}; \mathfrak{D}; S; comp^{\mathfrak{M}} \rangle$ una Σ-estructura. Una función $X \in \mathfrak{N}\mathfrak{F}_D^{(n)}$ es internamente finita en $\mathfrak{M}$ si existen un número interno k y n+1 secuencias internas $s_1, \ldots, s_n, s$ tales que

$$grafo(X) = \{\langle s_1[i], \ldots s_n[i], s[i] \rangle \ / \ 0 \leq i < k\}$$

donde $\leq$ y $<$ también tienen significado interno. ■

5- DEFINICION *(T-Derivación Interna)*

Sea $\mathfrak{M} = \langle \mathfrak{N}; \mathfrak{D}; S; comp^{\mathfrak{M}} \rangle$ una Σ-estructura y $T: \mathfrak{N}\mathfrak{F}_D^{(n)} \longrightarrow \mathfrak{N}\mathfrak{F}_D^{(n)}$ un operador.

(a) Decimos que una (n+1)-tupla $\langle \vec{s}, s \rangle$ de elementos de S, es una T-derivación interna de longitud k+1: si para todo $0 \leq j \leq k$,

$$s[j] \in T(fun_j(\langle \vec{s}, s \rangle))(\vec{s}[j])$$

donde $fun_j(\langle \vec{s}, s \rangle) \in \mathfrak{N}\mathfrak{F}_D^{(n)}$ es la función internamente finita cuyo grafo es el conjunto $\{\langle \vec{s}[1], s[1] \rangle \ / \ 0 \leq l < j\}$ y k, j y $\leq$ tienen también significados internos a $\mathfrak{M}$.

(b) Decimos que una (n+1)-tupla $\langle \vec{x}, y \rangle \in D^{n+1}$ es producida por la T-derivación interna $\langle \vec{s}, s \rangle$ de longitud k+1, si $\langle \vec{x}, y \rangle = \langle \vec{s}[k], s[k] \rangle$. ■

6- DEFINICION

Sea $\mathfrak{M} = \langle \mathfrak{N}; \mathfrak{D}; S; comp^{\mathfrak{M}} \rangle$ una Σ-estructura. A un operador $T: \mathfrak{N}\mathfrak{F}_D^{(n)} \longrightarrow \mathfrak{N}\mathfrak{F}_D^{(n)}$ le asociamos el operador que denotamos $\hat{T}: \mathfrak{N}\mathfrak{F}_D^{(n)} \longrightarrow \mathfrak{N}\mathfrak{F}_D^{(n)}$ definido por: $\hat{T}(X) = \cup\{T(X_0) \ / \ X_0 \in X$ internamente finita en $\mathfrak{M}\}$ para toda $X \in \mathfrak{N}\mathfrak{F}_D^{(n)}$. ■

7- DEFINICION *(Casi Menor Punto Fijo y Menor Punto Fijo Definible)*

Sea $\mathfrak{M} = \langle \mathfrak{N}; \mathfrak{D}; S; comp^{\mathfrak{M}} \rangle$ una Σ-estructura, Δ y δ un contexto y una valoración sobre $\mathfrak{M}$ y un operador $T: \mathfrak{N}\mathfrak{F}_D^{(n)} \longrightarrow \mathfrak{N}\mathfrak{F}_D^{(n)}$, una función $X \in \mathfrak{N}\mathfrak{F}_D^{(n)}$ es:

(a) El casi menor punto fijo definible de T en $\mathfrak{M}$ c.r.a Δ,δ sii verifica:
X es definible en $\mathfrak{M}$ c.r.a Δ,δ,
$\hat{T}(X)\sqsubset X\sqsubset T(X)$ y
$X\sqsubset Z$ para toda $Z\in\mathfrak{N}\mathfrak{F}_D^{(n)}$ definible en $\mathfrak{M}$ c.r.a Δ,δ y tal que $T(Z)\sqsubset Z$.

(b) El menor punto fijo definible de T en $\mathfrak{M}$ c.r. a Δ,δ sii verifica:
X es definible en $\mathfrak{M}$ c.r.a Δ,δ,
$T(X)=X$ y
$X\sqsubset Z$ para toda $Z\in\mathfrak{N}\mathfrak{F}_D^{(n)}$ definible en $\mathfrak{M}$ c.r.a Δ,δ y tal que $T(Z)\sqsubset Z$ ■

8- PROPIEDADES

Sea una Σ-estructura admisible $\mathfrak{M}=\langle\mathfrak{N};\mathfrak{D};S;comp^{\mathfrak{M}}\rangle$ y un operador monótono $T:\mathfrak{N}\mathfrak{F}_D^{(n)}\longrightarrow\mathfrak{N}\mathfrak{F}_D^{(n)}$, entonces:

(i) $\hat{T}=\hat{\hat{T}}$
(ii) $\hat{T}$ es monótono
(iii) T es continuo $\Leftrightarrow$ $\hat{T}=T$
(iv) Toda función de $\mathfrak{N}\mathfrak{F}_D^{(n)}$ finita es internamente finita en $\mathfrak{M}$
(v) $\hat{T}(Z)=T(Z)$ para toda $Z\in\mathfrak{N}\mathfrak{F}_D^{(n)}$ internamente finita
(vi) $\bar{T}\sqsubset\hat{T}$ ■

Ahora estamos en posición de demostrar el principal resultado de esta sección. Resultados similares han sido obtenidos por Cartwright [Car 83], [Car 84], Cartwright y McCarthy [CM 79], Hajek [Haj 86] y Pasztor [Paz 87] para otras lógicas de programas no estándar.

9- LEMA

Sea $G\in NFEXP_\Sigma^{(n)}$ positiva en la variable funcional n-aria X. Entonces, para cualquier Σ-estructura admisible $\mathfrak{M}=\langle\mathfrak{N};\mathfrak{D};S;comp^{\mathfrak{M}}\rangle$, contexto Δ y valoración δ sobre $\mathfrak{M}$, la denotación $Y=(\mu X.G)^{\mathfrak{M}}(\Delta,\delta)$ verifica las siguientes condiciones, donde $\hat{T}$ es el operador asociado al operador monótono $T=T(\mathfrak{M},\Delta,\delta,X,G)$ dado por $T(X)=G^{\mathfrak{M}}(\Delta[X/X],\delta)$:

(i) Y es definible en $\mathfrak{M}$ c.r.a Δ,δ

(ii) $Y \ll \hat{T}(Y)$

(iii) $\hat{T}(Y) \ll Y$

(iv) $Y \ll Z$ para toda $Z \in \mathfrak{N}\mathfrak{I}^{(n)}$ definible en $\mathfrak{M}$ c.r.a Δ,δ y tal que $\hat{T}(Z) \ll Z$

(v) $Y \ll T(Y)$

(vi) $Y \ll Z$ para toda $Z \in \mathfrak{N}\mathfrak{I}^{(n)}$ definible en $\mathfrak{M}$ c.r.a Δ,δ y tal que $T(Z) \ll Z$

Demostración

Asumamos la hipótesis. Por como está definida la semántica de ALRF, sabemos que , para cualquier $X \in \mathfrak{N}\mathfrak{I}_D^{(n)}$:

$$grafo(T(X)) = \{<\vec{x},y> \in D^{n+1} / E_G^{\mathfrak{M}}(\vec{x}:d, y:d)(\Delta[X/X], \delta[\vec{x},y/\vec{x}:d, y:d]) = true\}$$

Por el lema 2, sabemos que $E_G(\vec{x}:d, y:d)$ es positiva en X. T es entonces un operador monótono, por argumentos similares a los usados en las demostraciones de los lemas 15 y 16 de la sección 2.3.

Además $\hat{T}(X)$ es definible si lo es $X \in \mathfrak{N}\mathfrak{I}_D^{(n)}$, mediante la fórmula de primer orden extendido

$$E_{\hat{X}}(\vec{x}:d, y:d) = \exists \vec{u}:s \forall v:s \exists k:n E_X(\vec{u}:s, v:s, k:n, x:d, y:d),$$

donde, abusando de la notación

$$E_{\hat{X}}(\vec{u}:s, v:s, k:n, x:d, y:d) =$$
$$(GR(\vec{u}:s, v:s, k:n, \vec{x}:d, y:d) \subseteq grafo(X) \wedge$$
$$<\vec{x}:d, y:d> \in grafo(T(GR(\vec{u}:s, v:s, k:n, \vec{x}:d, y:d)))$$

siendo ambas partes de la conjunción definibles, ya que T y X lo son por hipótesis.

A lo largo de esta demostración, será necesario formalizar varias inducciones internas, mediante fórmulas de primer orden extendido. Y como lo más interesante es comprobar que el proceso de inducción es definible, para luego proceder semánticamente, no escribiremos estas fórmulas con rigor sintactico, sino como hemos hecho ahora para $E_{\hat{X}}$. Denotaremos por E_Z a la fórmula que define a la función Z, en el caso de que esta sea definible.

Trabajamos con grafos y con $F=\mu X.G$:

Demostración de (i)

Esto es trivial, puesto que la translación a primer orden extendido $E_{(\mu X.\phi)}$ define Y en $\mathfrak{M}$ c.r.a Δ, δ.

Demostración de (ii)

Sea $\langle\vec{x},y\rangle \in$ grafo(Y). Entonces $\langle\vec{x},y\rangle$ tiene una T-derivación interna. Por definición de T-derivación interna, se sigue que $\langle\vec{x},y\rangle \in$grafo($T(Y_j)$) para algun $Y_j \prec Y$ internamente finita en $\mathfrak{M}$. Entonces $\langle\vec{x},y\rangle \in$grafo($\hat{T}(Y)$) por definición de $\hat{T}$.

Demostración de (iii)

Sea $\langle\vec{x},y\rangle \in$grafo($\hat{T}(Y)$). Como hemos dicho antes, $\hat{T}(Y)$ es definible porque Y lo es. Por tanto podemos aplicar el axioma de inducción sobre la fórmula de primer orden extendido:

$$\forall l{:}n \forall \vec{u}_0{:}s \forall v_0{:}s\ (E_{Y^{\wedge}}(\vec{u}_0{:}s, v_0{:}s, l{:}n, \vec{x}{:}d, y{:}d) \rightarrow \exists \vec{u}{:}s \exists v{:}s \exists k{:}n$$
$$(BE_F(\vec{u}{:}s, v{:}s, k{:}n, \vec{x}{:}d, y{:}d) \wedge l{:}n \leq k{:}n \wedge \vec{u}_0{:}s \subseteq \vec{u}{:}s \wedge v_0{:}s \leq v{:}s))$$

cuyo significado intuitivo es que cualquier tupla producida por $\hat{T}(Y)$ mediante $T(Y_0)$, también es producida por una T-derivación interna que incluye al grafo de Y_0. Más fórmalmente, si $\langle\vec{x},y\rangle \in$grafo($\hat{T}(Y)$), por definición de $\hat{T}$ existe una función internamente finita $Y_0 \prec Y$ tal que $\langle\vec{x},y\rangle \in$grafo($T(Y_0)$). Supongamos que grafo($Y_0$) = $\{\langle\vec{x}_j, y_j\rangle / 0 \leq j < l\}$ para cierto número interno l. Para cada $0 \leq j < l$, existe (por definición de Y) en $\mathfrak{M}$ una T-derivación interna, codificada por algunas secuencias internas de $\mathfrak{M}$ $\vec{u}_j, v_j$ y con longitud dada por algún número interno de $\mathfrak{M}$ k_j, que produce $\langle\vec{x}_j, y_j\rangle$ como miembro de Y en el sentido expresado por la fórmula BE_F; ver la sección 2. Por el lema 2, $\mathfrak{M} \vdash$conc y $\mathfrak{M} \vdash$unit. Por inducción interna en $\mathfrak{M}$ sobre l se demuestra que existen n+1 secuencias internas en $\mathfrak{M}$ $\vec{u}$, v que son la concatenación de las l T-derivaciones internas anteriores, que codifican una T-derivación interna para $\langle\vec{x},y\rangle$. Y podemos concluir que $\langle\vec{x},y\rangle \in$grafo(Y). Veamos la inducción interna sobre l:

l=0: entonces $Y_0 = \perp$ y podemos tomar k=1, y convertir $x_1, \ldots, x_n$ e y en n+1 secuencias internas de longitud uno, usando unit.

l+1>0: l+1, $\vec{u}_0$, v_0 codifican el grafo de Y_0. Sea $\vec{u}_1, v_1$ y k_1 la T-derivación que existe para $\langle \vec{x}_1, y_1 \rangle \in \text{grafo}(Y_0)$. Por definición de T-derivación, $\langle \vec{u}_1(0), v_1(0) \rangle \in \text{grafo}(\hat{T}(\perp))$. Luego aplicando hipótesis de inducción a l sobre $\vec{u}_0$, v_0 y $\langle \vec{u}_1(0), v_1(0) \rangle$, obtenemos $\vec{u}'$, v' y k', que concatenamos con $\vec{u}_1, v_1$ y k_1 y entonces utilizando unit y conc otra vez, obtenemos n+1 secuencias internas finalizando en $x_1, \ldots, x_n$ e y, respectivamente. Puesto que T es monótono, estas secuencias codifican ahora una T-derivación interna de $\langle \vec{x}, y \rangle$, como se quería demostrar.
La formalización total de este argumento necesita usar propiedades de los números internos que están garantizadas por la Aritmética de Presburger.

Demostración de (iv)
Sea $Z \in \mathfrak{N}\mathfrak{J}_0^{(n)}$ definible en $\mathfrak{M}$ c.r.a Δ, δ y tal que $\hat{T}(Z) \subseteq Z$. Sea $\langle \vec{x}, y \rangle \in \text{grafo}(Y)$, entonces $\langle \vec{x}, y \rangle$ tiene una T-derivación interna de longitud k (número interno). Por inducción interna en $\mathfrak{M}$ sobre k se demuestra que $\langle \vec{x}, y \rangle \in \text{grafo}(Z)$. Obsérvese que esta inducción es posible, ya que al ser Z definible, podemos usar el axioma de inducción para la fórmula

$$(BE_F(k:n, \vec{x}:n, y:n) \wedge (E_{\hat{Z}}(\vec{x}:n, y:n) \rightarrow E_Z(\vec{x}:n, y:n))) \longrightarrow E_Z(\vec{x}:n, y:n).$$

k=0: $\langle \vec{x}, y \rangle \in T(\perp) = \hat{T}(\perp) \subseteq \hat{T}(Z) \subseteq Z$ por definición y monotonía de $\hat{T}$.

k+1>0: Sea $\vec{u}$, v la T-derivación interna de longitud k+1 que produce $\langle \vec{x}, y \rangle$. Entonces, por hipótesis de inducción, todos las tuplas $\langle \vec{u}[j], v[j] \rangle$ para $0 \leq j < k$, pertenecen al grafo de Z, puesto que cualquier prefijo de una T-derivación interna es también una T-derivación interna. Y por definición de T-derivación interna, $\langle \vec{x}, y \rangle \in \text{grafo}(T(\text{fun}_k \langle \vec{u}, v \rangle)) = \text{grafo}(\hat{T}(\text{fun}_k \langle \vec{u}, v \rangle)) \subseteq \text{grafo}(\hat{T}(Z))$ siendo $\text{grafo}(\text{fun}_k \langle \vec{u}, v \rangle) = \{ \langle \vec{u}[j], v[j] \rangle \ / \ 0 \leq j < k \}$ por monotonía de $\hat{T}$, y por tanto $\langle \vec{x}, y \rangle \in \text{grafo}(Z)$.

Demostración de (v)
Sea $\langle \vec{x}, y \rangle \in \text{grafo}(Y)$. Entonces $\langle \vec{x}, y \rangle$ tiene una T-derivación interna. Por la definición de derivación interna, se sigue que $\langle \vec{x}, y \rangle \in \text{grafo}(T(Y_j))$ para alguna $Y_j \subseteq Y$ internamente finita en $\mathfrak{M}$. Entonces $\langle \vec{x}, y \rangle \in \text{grafo}(T(Y))$ por monotonía de T.

Demostración de (vi)

Asumamos que $Z \in \mathfrak{N}\mathfrak{I}_D^{(n)}$ es definible en $\mathfrak{M}$ c.r.a Δ, δ y tal que $T(Z) \lhd Z$. Sea $\langle \vec{x}, y \rangle \in grafo(Y)$. Entonces, existe en $\mathfrak{M}$ una T-derivación interna que produce $\langle \vec{x}, y \rangle$, codificada por algunas secuencias internas de $\mathfrak{M}$ $\vec{u}, v$ y con longitud dada por algun número interno de $\mathfrak{M}$ k. Podemos aplicar el axioma de inducción sobre k dentro de $\mathfrak{M}$ para probarlo, sobre la fórmula de primer orden extendido:

$$(BE_F(k:n, \vec{x}:n, y:n) \wedge (E_{G[Ez/X]}(\vec{x}:n, y:n) \rightarrow E_z(\vec{x}:n, y:n))) \rightarrow E_z(\vec{x}:n, y:n).$$

Sea $\langle \vec{x}_j, y_j \rangle$ la tupla $\langle \vec{u}[j], v[j] \rangle$, computada en $\mathfrak{M}$, para todo número j interno en $\mathfrak{M}$ menor o igual que k. Entonces $\langle \vec{x}_k, y_k \rangle = \langle \vec{x}, y \rangle$, y para j menor que k en $\mathfrak{M}$, $\langle \vec{x}_j, y_j \rangle \in grafo(Z)$ por hipótesis de inducción, puesto que cualquier prefijo de una T-derivación interna es también una T-derivación interna. Sea $Z_k \lhd Z$ tal que el $grafo(Z_k)$ consiste de todas las tuplas $\langle \vec{x}_j, y_j \rangle$ para j menor que k en $\mathfrak{M}$. Entonces $\langle \vec{x}, y \rangle \in grafo(T(Z_k))$, porque $\langle \vec{u}, v \rangle$ codifica una T-derivación interna. Finalmente $\langle \vec{x}, y \rangle \in grafo(Z)$ por la hipótesis $T(Z) \lhd Z$ y monotonía.

Como en los demás puntos, la Aritmetica de Presburger soporta la formalización completa de esta argumentación. ■

Como corolario de este lema y de las propiedades 8, obtenemos los resultados de la caracterización semántica.

10- TEOREMA *(Semántica del Casi Menor Punto Fijo Definible)*

Sea $G \in NFEXP_\Sigma^{(n)}$ positiva en la variable funcional n-aría X. Entonces, para cualquier Σ-estructura admisible $\mathfrak{M} = \langle \mathfrak{N}; \mathfrak{D}; S; comp^{\mathfrak{M}} \rangle$, contexto Δ y valoración δ sobre $\mathfrak{M}$, la denotación $Y = (\mu X.G)^{\mathfrak{M}}(\Delta, \delta)$ es:

(a) El menor punto fijo definible en $\mathfrak{M}$ c.r.a Δ, δ del operador $\hat{T}$ asociado al operador monótono $T = T(\mathfrak{M}, \Delta, \delta, X, G)$ dado por $T(X) = G^{\mathfrak{M}}(\Delta[X/X], \delta)$.

(b) El casi menor punto fijo definible en $\mathfrak{M}$ c.r.a Δ, δ del operador monótono $T = T(\mathfrak{M}, \Delta, \delta, X, G)$ dado por $T(X) = G^{\mathfrak{M}}(\Delta[X/X], \delta)$. ■

11- TEOREMA *(Semántica del Menor Punto Fijo Definible)*

Sea $G \in NFEXP_{\Sigma}^{(n)}$ existencial en la variable funcional n-aría X. Entonces, para cualquier Σ-estructura admisible $\mathfrak{M} = \langle \mathfrak{N}; \mathfrak{D}; S; comp^{\mathfrak{M}} \rangle$, contexto Δ y valoración δ sobre $\mathfrak{M}$, la denotación $(\mu X.G)^{\mathfrak{M}}(\Delta, \delta)$ es el menor punto fijo definible del operador continuo $T=T(\mathfrak{M}, \Delta, \delta, X, G)$ dado por $T(X)=G^{\mathfrak{M}}(\Delta[X/X], \delta)$. ■

En estructuras aritméticas heterogéneas los conceptos de función finita y función internamente finita coinciden y por tanto los operadores $\bar{T}$ Y $\hat{T}$ son equivalentes. Luego en ALRFH sobre estas estructuras, la denotación de $\mu X.G$ es el menor punto fijo del operador $\bar{T}$ o $\hat{T}$ puesto que ahora podemos aplicar el principio de inducción sin tener en cuenta la definibilidad de las funciones. Y por tanto en ALRFc sobre estructuras aritméticas heterogéneas la denotación de $\mu X.G$ es el menor punto fijo del operador T. En resumen, ALRFc sobre estructuras aritméticas heterogéneas es equivalente a LRFc, como ocurría con NLRFc, mientras que ALRFH no lo es a LRFH puesto que los operadores T y $\bar{T}$ no son equivalentes.

12- EJEMPLO

ALRF permite algunos tipos de razonamientos sobre programas que son imposibles tanto en la demasiado estricta LRF como en la demasiado liberal NLRF. Como un ejemplo sencillo, consideremos una signatura Σ con una constante zero y un símbolo de función monádica succ. Sea PRED, DOBLE las Σ-expresiones funcionales siguientes:

$$PRED = \lambda x{:}d(x{:}d = zero \longrightarrow zero \cup \exists y{:}d(x{:}d = succ(y{:}d)) \longrightarrow y{:}d)$$

$$DOBLE = \mu X.\lambda x{:}d(x{:}d{=}zero \longrightarrow zero \cup \neg x{:}d{=}zero \longrightarrow succ(succ(X(PRED(x{:}d)))))$$

La fórmula $\varphi = \forall x{:}d\ \exists y{:}d\ DOBLE(x{:}d) \exists y{:}d$

que expresa la terminación débil de DOBLE, es intuitivamente cierta en el modelo deseado. Supongamos que escogemos un conjunto $\Phi \subseteq NEFOR_{\Sigma}$ como

axiomatización de los datos, que no incluye axiomas sobre los números ni las secuencías internas. Por el teorema 7 de la sección 2.4 (adaptado para nuestro formalismo de tres géneros), $\Phi \vdash \varphi$ no es cierto en LRF. Y tampoco lo es en NLRF, puesto que Φ admitiría también modelos arbitrarios. Sin embargo, $\Phi \vdash_A \varphi$ será cierto, siempre que Φ sea lo suficientemente fuerte como para permitir inducción sobre los datos. Bajo esta suposición, y para cualquier $\mathfrak{M} \vdash_A \Phi$ y cualquier dato interno x de $\mathfrak{M}$, $\mathfrak{M}$ tendrá dos secuencías internas u,v que codifiquen una derivación interna

$$<0^d, 0^d>, <1^d, 2^d>, \ldots, <x^d, (2x)^d>$$

la cual produce el valor $(2x)^d$ para DOBLE(x^d) en $\mathfrak{M}$ (aquí, $0^d, 1^d$, etc denotan informalmente datos de $\mathfrak{M}$). Veremos algun otro ejemplo de terminación de programas en la siguiente sección. Como hemos dicho en el capítulo 1, la terminación de programas imperativos en el marco de la lógica no estándar ha sido investigada por Sain [Sai 87]. ■

CALCULO ADMISIBLE

Hemos dicho que para ALRF$_H$ vamos a demostrar la corrección y completitud de un cálculo análogo al aritmético. Pero evidentemente, podemos obtener un cálculo finitario correcto y completo para ALRF, directamente del cálculo π dado para NLRF, que muestra que ALRF es esencialmente una lógica de primer orden.

13- TEOREMA

ALRF admite un cálculo finitario correcto y completo. Por lo tanto, verifica el teorema de compacidad y el teorema de Löwenheim-Skolem.

Demostración

Basta con tomar un cálculo finitario correcto y completo para NLRF (como el presentado en el teorema 5 de la sección 2) y añadirle el conjunto de axiomas de admisibilidad; es decir, $\Gamma \vdash Ax$ para todo $Ax \in A_\Sigma$. ■

Este cálculo no permite razonar de forma natural sobre el operador μ (recuerdese los comentarios a la demostración del teorema 5 de la sección 2). Ahora presentamos un cálculo orientado a la sintaxis que elimina esta dificultad y que tiene una aplicación interesante que veremos en la siguiente sección. A este cálculo lo denotamos por A y consiste de los siguientes axiomas y reglas.

Axiomas y reglas de primer orden

Como en el cálulo π para NLRF añadiendo todos los axiomas de admisibilidad (cfr. definición 1); es decir:

$\Gamma \vdash Ax$ para todo $Ax \in A_{\Sigma}$.

Reglas para la descomposición de las expresiones

Como en el cálculo π para NLRF pero omitiendo las reglas para el operador μ, junto con las siguientes dos nuevas reglas para este operador:

Regla de Invariancia

$$(I^{\bullet})\ \frac{\Gamma, G[\lambda \vec{x}:d.\varepsilon y:d.E(\vec{x}:d, y:d)/X](\vec{x}:d) \ni y:d \vdash E(\vec{x}:d, y:d) \qquad \Gamma \vdash \neg E(\vec{t}, t)}{\Gamma \vdash \neg(\mu X.G)(\vec{t}) \ni t}$$

Regla de Convergencia

$$(C^{\bullet})\ \frac{\Gamma,\ E'(k:n, \vec{x}:d, y:d) \vdash G[\lambda \vec{x}:d.\varepsilon y:d\ \exists l:n(l:n < k:n \wedge E'(l:n, \vec{x}:d, y:d))/X](\vec{x}:d) \ni y:d \qquad \Gamma \vdash \exists k:n E'(k:n, \vec{t}, t)}{\Gamma \vdash (\mu X.G)(\vec{t}) \ni t}$$

Como en el cálculo aritmético, en estas dos últimas reglas hemos usado notación adicional. $E(\vec{x}{:}d, y{:}d)$ representa una fórmula de primer orden extendido cuya variables libres incluyen a $\vec{x}{:}d, y{:}d$. La notación $\vec{t}, t$ se refiere a términos de género d. $E(\vec{t}, t)$ es una abreviatura de $E(\vec{x}{:}d, y{:}d)[\vec{t}/\vec{x}{:}d, t/y{:}d]$. G representa una expresión funcional positiva en la variable funcional X. La notación de (C^*) debe entenderse de forma similar.

El siguiente resultado muestra que ambas reglas reflejan la semántica de la recursión adecuadamente. La idea es que E y E' son fórmulas que expresan una función. De esta forma, intuitivamente la primera premisa de (I^*) afirma que $T(E) \lhd E$ y la segunda afirma que la tupla $\langle \vec{t}, t \rangle$ no pertenece al grafo de E, de lo que se concluye que tampoco pertenece al grafo de fix(T). Análogamente, la primera premisa de (C^*) afirma que $E'_k \lhd T(E'_{k+1})$ y la segunda que $\langle t, t \rangle$ es una tupla del grafo de $\exists k E'_k$, de lo que se concluye que también pertenece al grafo de fix(T). La corrección de este razonamiento se demuestra a continuación. Pero primero, definimos el concepto de deducibilidad formal en este cálculo.

El cálculo estructurado es también completo. La idea clave de la demostración es esencialmente la misma que en la demostración de completitud aritmética del capítulo anterior, análoga a la de la demostración de Cook [Coo 78] para la completitud relativa de un cálculo de verificación parcial o a la de la demostración de Harel [Har 79], [Har 84] para la completitud aritmética de la lógica dinámica. Los axiomas y reglas son adecuadas a la sintaxis de los programas (en nuestro caso expresiones y expresiones funcionales) y permiten reducir afirmaciones sobre ellos a otras afirmaciones más sencillas sobre partes sintácticas menores. Los axiomas o reglas necesarios siempre se pueden aplicar porque el lenguaje de primer orden es suficientemente expresivo. En el caso de Cook y Harel esto es posible porque se fija una estructura expresiva (respectivamente, aritmética). En nuestro caso, es el poder de los axiomas de admisibilidad.

Como siempre, denotamos por $\Phi \vdash_A \varphi$ a la relación de deducción formal en este cálculo. La definición es la dada en la sección anterior para el cálculo π para NLRF.

14- TEOREMA *(Corrección y Completitud Admisible)*

Para todo conjunto de fórmulas $\Phi \cup \{\varphi\}$ de ALRF:

$$\Phi \vdash_A \varphi \iff \Phi \vdash_A \varphi.$$

Demostración

CORRECCION: $\Phi \vdash_A \varphi \Rightarrow \Phi \vdash_A \varphi$.

Por inducción sobre la profundidad N del árbol de deducción, demostramos que la raíz de cualquier árbol de deducción es correcta. Decimos que una secuencia $\Gamma \vdash \varphi$ es correcta, si $\Gamma \vdash_A \varphi$.

N=0: Hay que comprobar que todas las instancias de los axiomas $\Gamma \vdash \varphi$ del cálculo A son correctas; es decir $\Gamma \vdash_A \varphi$. Todos los axiomas salvo los de admisibilidad son los del cálculo n para NLRF. Y los de admisibilidad son correctos por definición de $\vdash_A$.

N>0: Hay que comprobar que todas las instancias de las reglas del cálculo verifican que la conclusión es correcta si lo son las premisas. Todas las reglas son las mismas que las del cálculo n para NLRF, salvo las reglas del operador μ.

Fijemos una estructura admisible $\mathfrak{M}$. Consideremos una instancia de (I^*) y otra de (C^*). Supongamos que sus premisas son correctas y probamos que sus conclusiones también lo son. Para esto, tomemos un contexto Δ y una valoración δ sobre $\mathfrak{M}$ y consideramos:

(a) El operador monótono y continuo $T: \mathfrak{M}\mathfrak{I}_D^{(n)} \longrightarrow \mathfrak{M}\mathfrak{I}_D^{(n)}$ dado por
$$T(X) = G^{\mathfrak{M}}(\Delta[X/X], \delta)$$

(b) El menor punto fijo definible de T en $\mathfrak{M}$ bajo Δ, δ:
$$Y = (\mu X.G)^{\mathfrak{M}}(\Delta, \delta)$$

(c) La función no determinista $Z \in \mathfrak{M}\mathfrak{I}_D^{(n)}$ definida por
$$y \in Z(\vec{x}) \text{ iff } E(\vec{x}:d, y:d)^{\mathfrak{M}}(\Delta, \delta[\vec{x}/\vec{x}:d, y:d])$$

(d) para cada número interno k de $\mathfrak{M}$, $Z_k \in \mathfrak{N}\mathfrak{Y}^{(n)}$ definida por

$$y \in Z_k(x) \text{ iff } E'(k:n, \vec{x}:d, y:d)^{\mathfrak{M}}(\Delta, \delta[k/k:n, \vec{x}/\vec{x}:d, y/y:d])$$

(e) Los datos d_i, d definidos como $d_i = t_i^{\mathfrak{M}}(\Delta, \delta)$, $1 \leq i \leq n$; $d = t^{\mathfrak{M}}(\Delta, \delta)$

Supongamos que $\mathfrak{M} \models \Gamma$:

Para (I^*) Tenemos que probar que $\mathfrak{M} \models \neg(\mu X.G)(\vec{t}) \ni t$. El hecho de que la primera premisa sea correcta en $\mathfrak{M}$ significa que $T(Z) \ll Z$. Por el lema 9, se sigue que $Y \ll Z$. El hecho de que la segunda premisa sea correcta en $\mathfrak{M}$ implica que $d \notin Z(\vec{d})$. Por tanto, $d \notin Y(\vec{d})$, y la conclusión es correcta.

Para (C^*) Tenemos que probar que $\mathfrak{M} \models (\mu X.G)(\vec{t}) \ni t$. Puesto que la primera premisa es correcta, sabemos que $Z_k \ll T(\bigsqcup_{l<k} Z_l)$ para todo número interno k, donde l y < tienen también significados internos. Como todas las Z_k están uniformemente definidas por $E'(k:n, \vec{x}:d, y:d)$ y $\mathfrak{M}$ es admisible, por inducción sobre los números internos podemos concluir que $Z_k \ll Y$ para todo número interno k, aplicando el axioma de inducción a la fórmula

$$E'(k:n, \vec{x}:d, y:n) \longrightarrow \exists k:n\ BE_F(k:n, \vec{x}:d, y:d).$$

En efecto:

k=0: $Z_0 = T(\bot) \ll Y$, puesto que $T(\bigsqcup_{l<0} Z_l) = T(\bot)$ y $Y = fix(T)$.
k>0: $Z_k \ll T(\bigsqcup_{l<k} Z_l) = \bigsqcup_{l<k} T(Z_l) \ll \bigsqcup_{l<k} T(Y) = T(Y) = Y$ por hipotesis de inducción y monotonía de T.

Por otro lado, el hecho de que la segunda premisa sea correcta implica que $d \in Z_k(\vec{d})$ para algun número interno k. Por lo tanto, $d \in Y(\vec{d})$ y la conclusión es correcta.

COMPLETITUD: $\Phi \models_A \varphi \Rightarrow \Phi \vdash_A \varphi$.

Obviamente, esta cálculo es completo para NEL1 y por tanto, es suficiente demostrar los siguientes tres puntos (donde (2) y (3) juegan un papel auxiliar):

(1) Para toda $\varphi \in NFOR_\Sigma$: $\vdash_A \varphi \leftrightarrow E_\varphi$

(2) Para toda $M \in NEXP_\Sigma$: $\vdash_A M\exists y{:}\sigma \leftrightarrow E_M(y{:}\sigma)$

(3) Para toda $F \in FEXP_\Sigma^{(n)}$: $\vdash_A F(\vec{x}{:}d)\exists y{:}d \leftrightarrow E_F(\vec{x}{:}d, y{:}d)$

En efecto, si $\Phi \vdash_A \varphi$ entonces $E_\Phi \vdash_A E_\varphi$, y por la completitud de primer orden de A, $E_\Gamma \vdash_A E_\varphi$ con $E_\Gamma \subseteq E_\Phi$ finito, luego aplicando (1) y (MP) se obtiene $\Phi \vdash_A \varphi$.

Para demostrar (1)-(3), se procede por inducción simultánea sobre la función de complejidad $|.|$ sobre φ, M y F.

Veamos el caso de (3) cuando F es $\mu X.G$; los demás casos son sencillos. Demostramos:

(a) $\neg E_F(\vec{x}{:}d, y{:}d) \vdash_A \neg F(\vec{x}{:}d)\exists y{:}d$

(b) $E_F(\vec{x}{:}d, y{:}d) \vdash_A F(\vec{x}{:}d)\exists y{:}d$

Para esto usaremos las reglas (I^*) y (C^*).

(a) Usamos la rgla (I^*) de forma que como $E(\vec{x}{:}d, y{:}d)$ tomamos $\exists k{:}n\ BE_F(k{:}n, \vec{x}{:}d, y{:}d)$, es decir, $E_F(\vec{x}{:}d, y{:}d)$, cfr. sección 2. Con esto, la demostración formal para (a) puede construirse como sigue:

$$\dfrac{\neg E_F(\vec{x}{:}d, y{:}d) \vdash \neg F(\vec{x}{:}d)\exists y{:}d \quad |-(I^*)}{\dfrac{\neg E_F(\vec{x}{:}d, y{:}d) \vdash \neg E_F(\vec{x}{:}d, y{:}d)}{(SP)} \qquad \dfrac{\dfrac{\neg E_F(\vec{x}{:}d, y{:}d),\ G[\lambda\vec{x}{:}d.\varepsilon y{:}d\ E(\vec{x}{:}d, y{:}d)/X](\vec{x}{:}d)\exists y{:}d \vdash E(\vec{x}{:}d, y{:}d)}{\neg E_F(\vec{x}{:}d, y{:}d) \vdash \neg G[\lambda\vec{x}{:}d.\varepsilon y{:}d\ E(\vec{x}{:}d, y{:}d)/X](\vec{x}{:}d)\exists y{:}d}}{\neg E_F(\vec{x}{:}d, y{:}d) \vdash \neg E_{G[\lambda\vec{x}{:}d.\varepsilon y{:}d\ E(\vec{x}{:}d, y{:}d)/X]}(\vec{x}{:}d, y{:}d)}\ (a.1)}$$

Por la completitud de A para primer orden, basta con demostrar:

$$\vdash_A \neg E_F(\vec{x}:d, y:d) \rightarrow \neg E_{G[\lambda\vec{x}:d.\,\varepsilon y:d\ E(\vec{x}:d,y:d)/X]}(\vec{x}:d, y:d)$$

es decir

$$(a.1)\ \vdash_A E_{G[\lambda\vec{x}:d.\,\varepsilon y:d\ E(\vec{x}:d,y:d)/X]}(\vec{x}:d, y:d) \rightarrow E_F(\vec{x}:d, y:d)$$

Sean un contexto Δ y una valoración δ sobre $\mathfrak{M}$ fijadas. Consideremos T, Y y Z, definidas como en la demostración de la corrección. Para que (a.1) sea cierta en $\mathfrak{M}$ bajo Δ,δ es suficiente que

(a.2) $T(Z) \subseteq Z$

Además en este caso se da la igualdad. En efecto:

Como E es E_F, sabemos que $Z=(\mu X.G)^{\mathfrak{M}}(\Delta,\delta)$ (recordar la traducción para definir la semántica en la sección 2), es decir, Z=Y. Por el teorema 10, Y es el menor punto fijo definible de T, y por tanto se verifica T(Y)=Y, es decir (a.2). Inspeccionando la demostración del lema 9, podemos ver que la formalización de esta argumentación usa los axiomas unit y conc, además de la aritmética de Presburger.

(b) Usamos la regla (C^*) de forma que como $E'(k:n,\vec{x}:d,y:d)$ tomamos la fórmula $BE_F(k:n,\vec{x}:d,y:d)$, cfr. sección 2. Con esto la demostración formal de (b) puede construirse como sigue:

$$\dfrac{\dfrac{}{E_F(\vec{x}:d,y:d) \vdash F(\vec{x}:d)\ni y:d}\ (C^*)}{E_F(\vec{x}:d,y:d) \vdash \exists k:n\ E'(k:n,\vec{x}:n,y:n)}\ (SP)$$

$$E_F(\vec{x}:d,y:d),\ E'(k:n,\vec{x}:d,y:d) \vdash G[\lambda\vec{x}:d.\,\varepsilon y:d\ \exists l:n(l:n<k:n \wedge E'(l:n,\vec{x}:d,y:d))/X](\vec{x}:d)\ni y:d$$

$$E_F(\vec{x}:d,y:d),\ E'(k:n,\vec{x}:d,y:d) \vdash E_{G[\lambda\vec{x}:d.\,\varepsilon y:d\ \exists l:n(l:n<k:n \wedge E'(l:n,\vec{x}:d,y:d))/X]}(\vec{x}:d,y:d)\ (b.1)$$

Por la completitud de A para primer orden, basta con demostrar:

(b.1) $\vdash_A E'(k{:}n, \vec{x}{:}d, y{:}d) \rightarrow$

$$E_{G[\lambda \vec{x}:d.\varepsilon y:d\ \exists l:n(l:n<k:n \wedge E'(l:n, \vec{x}:d, y:d))/X]}(\vec{x}:d, y:d)$$

Sean un contexto Δ y una valoración δ sobre $\mathfrak{M}$ fijadas. Consideremos T, Y y Z_k (para todo número interno k), definidas como en la demostración de la corrección. Para que (b.1) sea cierta en $\mathfrak{M}$ bajo Δ, δ es suficiente que:

(b.2) $Z_k \ll T(\bigsqcup_{l<k} Z_l)$ para todo número interno k, donde l, < tienen también significado interno

Además en este caso se da la igualdad. En efecto:

Como E' es BE_F, el grafo de cada Z_k consiste exactamente de aquellas tuplas $\langle \vec{x}, y \rangle$ las cuales tienen una T-derivación interna de longitud k+1 (recordar la construcción de BE_F). Luego por definición de T-derivación interna se verifica que $Z_0 = T(\perp)$, $\bigsqcup_{l<k} Z_l = Z_{k-1}$ (puesto que una derivación que produce $\langle \vec{x}, y \rangle$ se convierte en otra de mayor longitud que también produce $\langle \vec{x}, y \rangle$, sin más que repetir alguno de sus elementos en el mismo lugar donde aparecen) y que $T(Z_{k-1}) = Z_k$. Luego se verifica (b.2). La formalización de esta argumentación usa la estructura lógica de BE_F, y la aritmética de Presburger. ∎

15- EJEMPLO

Cerramos esta sección con un ejemplo muy sencillo que ilustra el razonamiento formal en este cálculo. Supongamos que Σ incluye dos constantes cero y uno y un símbolo de función binaria sum. Sea $\mathfrak{D}$ una Σ-estructura de un género de dominio $\mathbb{N}$ y donde cero, uno, y sum se interpretan de forma natural. Consideremos el conjunto Φ de Σ-fórmulas de primer orden extendido válidas en $\mathfrak{D}$, y la expresión funcional

$$\text{ZERO} = \mu X.\lambda x{:}d\ (x{:}d = \text{cero} \rightarrow \text{cero} \cup \neg\, x{:}d = \text{cero} \rightarrow X(\text{dif}(x{:}d, \text{uno})))$$

Veamos que $\Phi \vdash_{\overline{A}}$ ZERO(x:d)∋cero

Obsérvese que este teorema trivial no es demostrable ni en LRF ni en NLRF.

La idea es especificar mediante una fórmula de primer orden extendido, el grafo de Zero. Evidentemente, un par <x,y>∈grafo(Zero) si y=0 y x puede generarse a partir de las constantes 0 y 1 y la función + de la estructura pseudoaritmética $\mathfrak{M}_D$.

Para obtener un árbol de derivación para $\Phi \vdash_A$ Zero(x:d)∋cero, usaremos la regla (C^*) de forma que

$$E'(k{:}n, x{:}d, y{:}d) = eq(k{:}n, x{:}d) \wedge y{:}d = cero$$

siendo

$$eq(k{:}n, x{:}d) = \exists u{:}s(u{:}s[0{:}n] = cero \wedge u{:}s[k{:}n] = x{:}d \wedge \forall l{:}n(l{:}n<k{:}n \longrightarrow u{:}s[l{:}n+1] = sum(u{:}s[l{:}n], uno)))$$

$$G = \lambda x{:}d.\ (x{:}d=cero \rightarrow cero \cup \neg\, x{:}d = cero \longrightarrow X(dif(x{:}d, uno)))$$

$$G[\lambda x{:}d.\varepsilon y{:}d\ \exists l{:}n\ (l{:}n<k{:}n \wedge E'(l{:}n, x{:}d, y{:}d))/X] = \lambda x{:}d.\ (x{:}d=cero \rightarrow cero \cup \neg x{:}d=cero \rightarrow (\lambda x{:}d.\ \varepsilon y{:}d\ \exists l{:}n\ l{:}n<k{:}n \wedge E'(l{:}n, x{:}d, y{:}d))(dif(x{:}d, uno)))$$

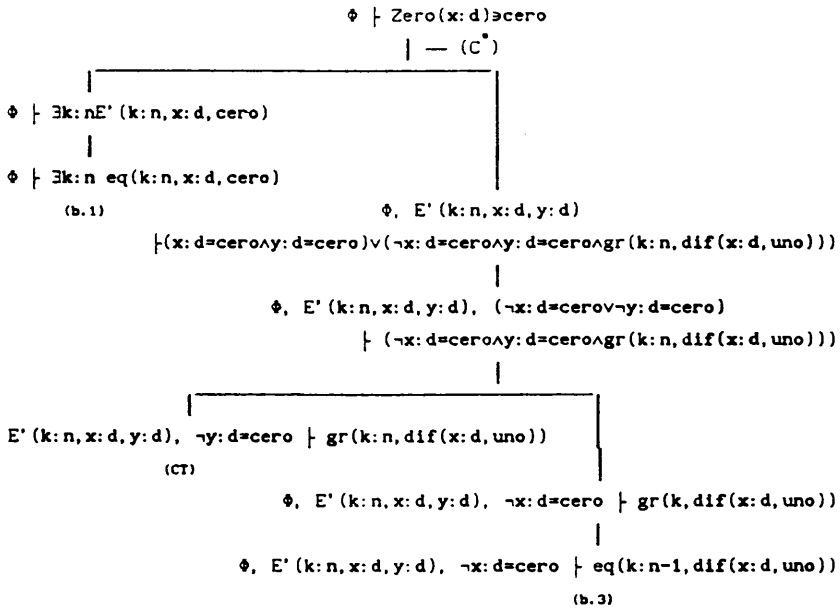
y

$$\vdash G[\lambda x{:}d.\varepsilon y{:}d\ \exists l{:}n\ (l{:}n<k{:}n \wedge E'(l{:}n, x{:}d, y{:}d))/X](x{:}d) \ni y{:}d \leftrightarrow (x=cero \wedge y=cero) \vee (\neg x=cero \wedge y=cero \wedge gr(k{:}n, dif(x{:}d, uno)))$$

con

$$gr(k{:}n, x{:}d) = \exists l{:}n\ l{:}n<k{:}n \wedge eq(k{:}n, x{:}d).$$

Por tanto:



Razonemos que:

(b.1) $\Phi \vdash_A \exists k{:}n\ eq(k{:}n, x{:}d, y{:}d)$

(b.3) $\Phi \vdash_A (E'(k{:}n, x{:}d, y{:}d) \wedge \neg x{:}d{=}cero \rightarrow eq(k{:}n{-}1, dif(x{:}d, uno)))$

(b.1) Basta con tomar $k{=}x \in N$ y $u{=}\langle 0, 1, \ldots, x{-}1, x\rangle$.

(b.3) Basta con tomar u tal que $u^C{=}$ concat(u,<x>), siendo u^C la secuencia que existe para eq(k:n,x:d), segun E'(k:n,x:d,y:d). ■

4.- APLICACION: COMPLETITUD DEL CALCULO ARITMETICO

Como se ha dicho en el capítulo 1, una característica muy atractiva del estudio no estándar de las lógicas de programas es que sirve de herramienta para probar resultados estándar. En esta última sección presentamos una aplicación de este tipo, mediante una técnica de traducción, podemos demostrar la completitud aritmética del cálculo α dado para LRF en la sección 3.2 a partir del cálculo estructurado de la sección anterior. Esta técnica puede aplicarse para otros cálculos, como el aritmético para DL citado en el capítulo 1 (cfr. Harel [Har 79]).

En primer lugar, como los símbolos usados para el género n en las estructuras pseudoaritméticas coinciden con los exigidos para las estructuras aritméticas, modificamos estos símbolos para que no colapsen.

1- NOTACION

El concepto de Σ-estructura aritmética se introdujo en la definición 1 de la sección 3.2. En particular Σ tiene que contener dos símbolos de constantes para el cero y el uno, dos símbolos de función binarias para la suma y el acceso a componentes de secuencias y un símbolo de predicado para distinguir a los números naturales. Para no confundir estos símbolos con sus análogos en estructuras pseudoaritméticas, los denotaremos en esta sección por: 0_d, 1_d, $+_d$, $comp_d$ y nat_d. ■

Ahora no debe confundirse el concepto de estructura aritmética con el de estructura aritmética heterogénea de la definición 2 de la sección 1.

Una Σ-estructura aritmética puede verse como una Σ_{pa}-estructura de tres géneros:

2- DEFINICION

Sea $\mathfrak{D}$ una Σ-estructura aritmética. La vista de tres géneros de $\mathfrak{D}$ es la Σ_{pa}-estructura

$$\mathfrak{M}_{\mathfrak{D}}^{ar} = \langle \mathfrak{N}, \mathfrak{D}, D, comp^{\mathfrak{M}_{\mathfrak{D}}^{ar}} \rangle$$

donde $\mathfrak{N}$ es estándar, $S=D$, y $comp^{\mathfrak{M}_{\mathfrak{D}}^{ar}}(i,u) = comp_d^{\mathfrak{D}}(i,u)$ para cualquier $i\in N$, $u\in S$. Obsérvese que los datos internos de $\mathfrak{M}_{\mathfrak{D}}^{ar}$ son los mismos objetos que las secuencias internas, e incluyen a los números naturales. ■

En la sección 2, remarcamos que la semántica de NLRFc es esencialmente equivalente a la de LRFc sobre estructuras aritméticas heterogéneas. Necesitamos una versión precisa de un resultado similar para estructuras aritméticas. Para este propósito, definimos las traducciones sintácticas del lenguaje de LRF al de NLRF y viceversa.

3- DEFINICION

Sea Σ una signatura aritmética. Tenemos que definir las traducciones sintácticas por recursión simultánea sobre fórmulas, expresiones y expresiones funcionales como sigue:

(a) Traducción $(.)^{ar}$ desde LRF a ALRF

(a.1) $\varphi^{ar}\in NFFOR_\Sigma$ para toda $\varphi\in FFOR_\Sigma$

$(t_1=t_2)^{ar} = (t_1^{ar}=t_2^{ar})$

$p(M_1,\ldots,M_n)^{ar} = p(M_1^{ar},\ldots,M_n^{ar})$

$(M\ni t)^{ar} = (M^{ar}\ni t^{ar})$

$(\neg\varphi)^{ar} = \neg\,\varphi^{ar}$

$(\varphi\vee\psi)^{ar} = (\varphi^{ar}\vee\psi^{ar})$

$(\exists x\varphi)^{ar} = \exists x{:}d\;\varphi^{ar}$

(a.2) $M^{ar}\in NEXP_\Sigma$ para toda $M\in EXP_\Sigma$

$x^{ar} = x{:}d$

$F(M_1,\ldots,M_n)^{ar} = F^{ar}(M_1^{ar},\ldots,M_n^{ar})$

$(M\cup N)^{ar} = (M^{ar}\cup N^{ar})$

$(\varphi\rightarrow M)^{ar} = (\varphi^{ar}\rightarrow M^{ar})$

$(\varepsilon x\;\varphi)^{ar} = \varepsilon x{:}d\;\varphi^{ar}$

(a.3) $F^{ar} \in NFEXP_{\Sigma}^{(n)}$ para toda $F \in FEXP_{\Sigma}^{(n)}$

$f^{ar} = f$

$X^{ar} = X$

$(\lambda x_1 \ldots x_n . M)^{ar} = \lambda x_1 : d \ldots x_n : d . M^{ar}$

$(\mu X . G)^{ar} = \mu X . G^{ar}$

(b) Traducción $(.)^d$ desde ALRF a LRF

Suponemos fijada una traducción de variables una a una

$(.)^d : VAR_n \cup VAR_d \cup VAR_s \longrightarrow VAR$,

La traducción de los Σ_{pa}-términos de género s se define por:

$(x:s)^d$: definida por la traducción de variables fijada

La traducción de los Σ_{pa}-términos de género n se define por:

$(x:n)^d$: definida por la traducción de variables fijada

$0^d = 0_d$

$1^d = 1_d$

$(t_1 + t_2)^d = t_1^d +_d t_2^d$

(b.1) $\varphi^d \in FFOR_{\Sigma}$ para toda $\varphi \in NFFOR_{\Sigma}$

$(t_1 = t_2)^d = (t_1^d = t_2^d)$

$p(M_1, \ldots, M_n)^d = p(M_1^d, \ldots, M_n^d)$

$(M \ni T)^d = (M^d \ni t^d)$

$(\neg \varphi)^d = \neg \varphi^d$

$(\varphi \vee \psi)^d = (\varphi^d \vee \psi^d)$

$(\exists x:n \; \varphi)^d = \exists (x:n)^d \; (nat(x:n)^d \wedge \varphi^d)$

$(\exists x:d \; \varphi)^d = \exists (x:d)^d \; \varphi^d$

$(\exists x:s \; \varphi)^d = \exists (x:s)^d \; \varphi^d$

(b.2) $M^d \in EXP_{\Sigma}$ para toda $M \in NEXP_{\Sigma}$

$(x:n)^d$: definida por la traducción de variables fijada

$F(M_1, \ldots, M_n)^d = F^d(M_1^d, \ldots, M_n^d)$

$(M \cup N)^d = (M^d \cup N^d)$

$(\varphi \longrightarrow M)^d = (\varphi^d \longrightarrow M^d)$

$(\varepsilon x:d \; \varphi)^d = \varepsilon (x:d)^d \; \varphi^d$

(b.3) $F^d \in FEXP_{\Sigma}^{(n)}$ para toda $F \in NFEXP_{\Sigma}^{(n)}$

$comp = comp_d$

$f^d = f$ para $f \neq comp$

$X^d = X$

$(\lambda x_1 : d \ldots x_n : d . \; M)^d = \lambda (x_1 : d)^d \ldots (x_n : d)^d . M^d$

$(\mu X . G)^d = \mu X . G^d$ ■

El significado de la traducción se explica en el siguiente resultado.

4- LEMA *(Lema de Traducción)*

Para una Σ-estructura aritmética $\mathfrak{D}$, $\mathfrak{M}_{\mathfrak{D}}^{ar}$ es admisible. Además:

(a) Para un contexto Δ y una valoración δ sobre $\mathfrak{D}$, existen sus correspondientes Δ^{ar}, δ^{ar} sobre $\mathfrak{M}_{\mathfrak{D}}^{ar}$ tal que:

(a.1) Para $\varphi \in FFOR_{\Sigma}$: $\mathfrak{D} \models \varphi(\Delta,\delta)$ sii $\mathfrak{M}_{\mathfrak{D}}^{ar} \models \varphi^{ar}(\Delta^{ar},\delta^{ar})$

(a.2) Para $M \in EXP_{\Sigma}$: $M^{\mathfrak{D}}(\Delta,\delta) = (M^{ar})^{\mathfrak{M}_{\mathfrak{D}}^{ar}}(\Delta^{ar},\delta^{ar})$

(a.3) Para $F \in FEXP_{\Sigma}^{(n)}$: $F^{\mathfrak{D}}(\Delta,\delta) = (F^{ar})^{\mathfrak{M}_{\mathfrak{D}}^{ar}}(\Delta^{ar},\delta^{ar})$

(b) Para un contexto Δ y una valoración δ sobre $\mathfrak{M}_{\mathfrak{D}}^{ar}$, existen sus correspondientes Δ^{d}, δ^{d} sobre $\mathfrak{D}$ tal que:

(b.1) Para $\varphi \in NFFOR_{\Sigma}$: $\mathfrak{M}_{\mathfrak{D}}^{ar} \models \varphi(\Delta,\delta)$ sii $\mathfrak{D} \models \varphi^{d}(\Delta^{d},\delta^{d})$

(b.2) Para $M \in NEXP_{\Sigma}$: $M^{\mathfrak{M}_{\mathfrak{D}}^{ar}}(\Delta,\delta) = (M^{d})^{\mathfrak{D}}(\Delta^{d},\delta^{d})$

(b.3) Para $F \in NFEXP_{\Sigma}^{(n)}$: $F^{\mathfrak{M}_{\mathfrak{D}}^{ar}}(\Delta,\delta) = (F^{d})^{\mathfrak{D}}(\Delta^{d},\delta^{d})$

(c) $(.)^{d}$ se comporta como la inversa de $(.)^{ar}$. Más exactamente:

(c.1) Para $\varphi \in FFOR_{\Sigma}$: $(\varphi^{ar})^{d} = \varphi$

(c.2) Para $M \in EXP_{\Sigma}$: $(M^{ar})^{d} = M$

(c.3) Para $F \in FEXP_{\Sigma}^{(n)}$: $(F^{ar})^{d} = F$

supuesto que se toma una traducción adecuada de variables (que depende de φ, M o F, respectivamente).

Demostración

Sea $\mathfrak{D}$ aritmética. Entonces $\mathfrak{M}_{\mathfrak{D}}^{ar}$ es admisible, porque los números internos de $\mathfrak{M}_{\mathfrak{D}}^{ar}$ son los estándar, y las secuencías internas de $\mathfrak{M}_{\mathfrak{D}}^{ar}$ se comportan como los códigos de las secuencias finitas estándar.

Para (a), es suficiente tomar $\Delta^{ar}=\Delta$ y $\delta^{ar}(x:d)=\delta(x)$ para toda $x\in VAR$. Entonces, (a.1) - (a.3) puede comprobarse mediante una inducción trivial.

Para (b), la argumentación es similar, pero tomando $\Delta^{d}=\Delta$ y $\delta^{d}((x:\sigma)^{d})=\delta(x:\sigma)$ para toda $x:\sigma\in VAR_n \cup VAR_d \cup VAR_s$.

Finalmente, para (c) es suficiente escoger una traducción de variables con la propiedad de que $(x:d)^{d}=x$ para toda variable x que aparezca en φ, M o F, respectivamente. Obsérvese que solo hay un número finito de estas variables. ■

El siguiente lema técnico es necesario para la demostración de la completitud aritmética del teorema 6.

5- LEMA

Sea t un Σ_{pa}-término de género σ y F una Σ_{pa}-expresión funcional de NLRF:
(a) Para todo Σ_{pa}-término t_1 de género σ, $t_1[t/x:\sigma]^{d} = t_1^{d}[t^{d}/(x:\sigma)^{d}]$
(b) Para toda Σ_{pa}-fórmula, Σ_{pa}-expresión o Σ_{pa}-expresión funcional Θ,

$\Theta[t/x:\sigma]^{d} = \Theta^{d}[t^{d}/(x:\sigma)^{d}]$
$\Theta[F/X]^{d} = \Theta^{d}[F^{d}/X]$

Demostración

(a) Para cada género σ, por inducción sobre los términos.

(b) Por inducción simultánea sobre las fórmulas, expresiones y expresiones funcionales.

Veamos el caso $(\exists x:n\ \varphi)[t/y:n]$ para $x\neq y$:
$(\exists x:n\ \varphi)[t/y:n]^{d}$ = por definición
$\exists x:n\ \varphi[t/y:n]^{d}$ = por hipótesis de inducción
$\exists(x:n)^{d}\ (nat((x:n)^{d}) \wedge \varphi^{d}[t^{d}/(y:n)^{d}])$ = por definición
$\exists(x:n)^{d}\ (nat((x:n)^{d}) \wedge \varphi^{d})[t^{d}/(y:n)^{d}]$ = por definición
$(\exists x:n\ \varphi)^{d}[t^{d}/(y:n)^{d}]$ como se quería demostrar. ■

Ahora estamos en posición de demostrar el resultado principal de esta sección.

6- TEOREMA *(Corrección y Completitud Aritmética)*

El cálculo α de la sección 3.2 es aritméticamente correcto y completo para la lógica estándar LRFc. Es decir: para una Σ-estrucutra aritmética $\mathfrak{D}$ y una fórmula de LRFc $\varphi \in FFOR_{\Sigma}$, se verifica

$$\mathfrak{D} \models \varphi \iff ETh(\mathfrak{D}) \vdash_{\alpha} \varphi$$

Demostración

Fijemos una estructura aritmética $\mathfrak{D}$.

(a) Corrección: $Eth(\mathfrak{D}) \vdash_{\alpha} \varphi \Rightarrow \mathfrak{D} \models \varphi$

Ya demostrado en la sección 3.2.

(b) Completitud: $\mathfrak{D} \models \varphi \Rightarrow Eth(\mathfrak{D}) \vdash_{\alpha} \varphi$

Supongamos que $\mathfrak{D} \models \varphi$. Por el lema 4, sabemos que $\mathfrak{M}_{\mathfrak{D}}^{ar}$ es admisible y que $\mathfrak{M}_{\mathfrak{D}}^{ar} \models \varphi^{ar}$. Por el lema de traducción, $(\varphi^{ar})^{d} = \varphi$. Consideremos la teoría de primer orden extendida de $\mathfrak{M}_{\mathfrak{D}}^{ar}$ en el lenguaje de tres géneros de LRFc que es:

$$ETh(\mathfrak{M}_{\mathfrak{D}}^{ar}) = \{ E \in FOR_{\Sigma_{pa}} \;/\; \mathfrak{M}_{\mathfrak{D}}^{ar} \models E \}$$

Puesto que la semántica de ALRF se ha definido a traves de una traducción sintáctica al lenguaje de primer orden extendido (cfr. definición 3 de la sección 2), $\mathfrak{M}_{\mathfrak{D}}^{ar} \models \varphi^{ar}$ implica que φ^{ar} es consecuencia lógica de la teoría de primer orden extendido de $\mathfrak{M}_{\mathfrak{D}}^{ar}$ en ALRF: $ETh(\mathfrak{M}_{\mathfrak{D}}^{ar}) \vdash_{A} \varphi^{ar}$ y por tanto de su cierre universal, $ETh(\mathfrak{M}_{\mathfrak{D}}^{ar})^{\forall} \vdash_{A} \varphi^{ar}$. Entonces, por el teorema de completitud admisible, sabemos que $ETh(\mathfrak{M}_{\mathfrak{D}}^{ar})^{\forall} \vdash_{\alpha} \varphi^{ar}$. Ahora, puede verse que $(.)^{d}$ traducida esta deducción formal a otra deducción formal que prueba $ETh(\mathfrak{D}) \vdash_{\alpha} (\varphi^{ar})^{d} = \varphi$. Más precisamente demostramos para toda secuencia $\Gamma \models \varphi$ en la que no aparecen variables libres de género n:

(*) si $\Gamma\vdash\varphi$ tiene un árbol de deducción $\mathfrak{A}$ en A "adecuado", entonces $\Gamma^d\vdash\varphi^d$ tiene un árbol de deducción $\mathfrak{A}^*$ en α, donde $\Gamma^d = \{ \psi^d / \psi\in\Gamma \}$

Donde por un árbol "adecuado" $\mathfrak{A}$ para una secuencia $\Gamma\vdash\varphi$ sin variables libres de género n, entendemos que cada vez que se aplica en $\mathfrak{A}$ la regla:

$$(\exists\text{-C})\quad \frac{\Gamma \vdash \varphi[t/x:n]}{\Gamma \vdash \exists x:n\varphi}$$

obsérvese que solo se refiere a variables de género n

el término t (de género n) es cerrado y los símbolos por los que puede estar formado son: +, 0, 1 y las constantes de C_n (constantes nuevas de género n) que aparecen en $\Gamma \vdash \exists x:n\varphi$.

Una vez demostrado esto, podemos razonar como sigue:
$ETh(\mathfrak{M}_{\mathfrak{D}}^{ar})^{\forall}\vdash_{\overline{A}} \varphi^{ar}$, implica por definición que $\Gamma\vdash\varphi^{ar}$ tiene un árbol $\mathfrak{A}$ de deducción en A, para $\Gamma\subseteq ETh(\mathfrak{M}_{\mathfrak{D}}^{ar})^{\forall}$ finito. En $\Gamma\vdash\varphi^{ar}$ no aparecen variables libres de género n (en Γ no aparecen variables libres, y en φ^{ar} no aparecen variables de género n), y podemos suponer sin pérdida de generalidad que este árbol de deducción es "adecuado". Luego $\Gamma^d\vdash(\varphi^{ar})^d=\varphi$ tiene un árbol de deducción $\mathfrak{A}^*$ en α, para $\Gamma^d\subseteq ETh(\mathfrak{D})$ finito por el lema de traducción, como queríamos demostrar.

Veamos la demostración de (*). Por inducción sobre la profundidad N del árbol $\mathfrak{A}$. Puesto que los axiomas y reglas de ambos cálculos tiene los mismos nombres, usamos subíndices para diferenciarlos.
N=0: Distinguimos cada axioma de A.

$(SP)_A$ $\Gamma\vdash\varphi$ por que $\varphi\in\Gamma$. Entonces $\varphi^d\in\Gamma^d$ y $\Gamma^d\vdash\varphi^d$ es un axioma de $\alpha(\mathfrak{D})$.

$(ID)_A$ $\Gamma\vdash t=t$. Entonces $\Gamma^d\vdash t^d=t^d$ es un axioma de $\alpha(\mathfrak{D})$.

$(ffA)_A$ $\Gamma, ff\vdash\varphi$. Entonces $\Gamma^d, ff\vdash\varphi^d$ es un axioma de $\alpha(\mathfrak{D})$.
Obsérvese que estamos usando el mismo símbolo para denotar a las fórmulas $\exists x:\sigma(\neg\, x:\sigma=x:\sigma)$ y $\exists x(\neg\, x=x)$ en NLRF y LRF respectivamente.

$(Ax)_A$ $\Gamma\vdash Ax$ siendo $Ax\in A_{\Sigma}$ un axioma de admisibilidad. Entonces $\Gamma^d\vdash Ax^d$ es un axioma de $\alpha(\mathfrak{D})$.

N>0: La raíz de $\mathfrak{A}$ tiene uno o dos subárboles $\mathfrak{A}_1$, dependiendo de la regla que se haya aplicado, de profundidad menor que N, $\mathfrak{A}_1$ es árbol de deducción en A para su raiz $\Gamma_1 \vdash \varphi_1$ y existe una regla en el cálculo A de forma que $\Gamma \vdash \varphi$ es la conclusión y $\Gamma_1 \vdash \varphi_1$ las premisas de la regla.. En $\Gamma_1 \vdash \varphi_1$ no aparecen variables libres de género n y $\mathfrak{A}_1$ es "adecuado", ya que en $\Gamma \vdash \varphi$ no aparecen variables libres de género n y $\mathfrak{A}$ es adecuado. Luego por hipótesis de inducción, existe $\mathfrak{A}_1^*$ árbol de deducción en α para $\Gamma_1^d \vdash \varphi_1^d$.

Tenemos que considerar cada regla separadamente, pero agruparemos aquellas para las que el razonamiento sea análogo.

(1) Reglas de primer orden para el cuantificador $\exists$.

$$(\exists\text{-A})_A \quad \frac{\Gamma,\ \varphi[c/x:n] \vdash \psi}{\Gamma,\ \exists x:n\varphi \vdash \psi} \quad \text{(c nueva) en el caso en que x sea de género n.}$$

Por el lema 5: $\Gamma^d, \varphi[c/x:n]^d \vdash \psi^d = \Gamma^d, \varphi^d[c^d/(x:n)^d] \vdash \psi^d$.
por definición: $\Gamma^d, (\exists x:n\varphi)^d \vdash \psi^d = \Gamma^d,\ \exists(x:n)^d(\mathrm{nat}((x:n)^d) \wedge \varphi^d) \vdash \psi^d$.
Entonces, aplicando la regla $(\exists\text{-A})_\alpha$:

$$\Gamma^d,\ \exists(x:n)^d(\mathrm{nat}((x:n)^d) \wedge \varphi^d) \vdash \psi^d$$
$$\Big|$$
$$\Gamma^d,\ \mathrm{nat}(c^d) \wedge \varphi^d[c^d/(x:n)^d] \vdash \psi^d$$
$$\Big|$$
$$\Gamma^d,\ \mathrm{nat}(c^d),\ \varphi^d[c^d/(x:n)^d] \vdash \psi^d$$

hipótesis de inducción

Obsérvese que cada vez que se añade una nueva constante de género n en $\mathfrak{A}$, en $\mathfrak{A}^*$ queda constancia de que esta constante es natural.

$$(\exists\text{-C})_A \quad \frac{\Gamma \vdash \varphi[t/x:n]}{\Gamma \vdash \exists x:n\varphi} \quad \text{en el caso en que x es una variable de género n.}$$

Por el lema 5: $\Gamma^d \vdash \varphi[t/x:n]^d = \Gamma^d \vdash \varphi^d[t^d/(x:n)^d]$.
Por definición: $\Gamma^d \vdash (\exists x:n\varphi)^d = \Gamma^d \vdash \exists(x:n)^d(nat((x:n)^d) \wedge \varphi^d)$.

Entonces, aplicando la regla (∃-C)α:

$$\dfrac{\dfrac{\Gamma^d \vdash \exists(x:n)^d(nat((x:n)^d) \wedge \varphi^d)}{\Gamma^d \vdash nat(t^d) \wedge \varphi^d[t^d/(x:n)^d]}}{\dfrac{\Gamma^d \vdash nat(t^d)}{\Gamma^* \vdash \chi \rightarrow nat(t^d)}\ ETh(\exists)} \qquad \dfrac{}{\Gamma^d \vdash \varphi^d[t^d/(x:n)^d]}\ \text{hipótesis de inducción}$$

siendo $\chi = \bigwedge nat(c)$, para toda c que aparece en t^d y $\Gamma^* = \Gamma^d \setminus (nat(c)/c$ que aparece en $t^d\}$. Obsérvese que podemos asegurar que $nat(c) \in \Gamma^d$ para toda c que aparece en t^d, por que hemos exigido que t sea un término "adecuado", según el concepto de árbol de deducción adecuado.

(2) Reglas de primer orden en A salvo las del cuantificador ∃.

Cada regla $(.)_A$ se adecua directamente a su correspondiente regla (.)α. Por ejemplo:

$$(SUS)_A\ \dfrac{\Gamma \vdash \varphi[t/x:\sigma]}{\Gamma, t=t' \vdash \varphi[t'/x:\sigma]}$$

Por el lema 5: $\Gamma^d \vdash \varphi[t/x:\sigma]^d = \Gamma^d \vdash \varphi^d[t^d/(x:\sigma)^d]$.
Por definición: $\Gamma^d, (t=t')^d \vdash \varphi[t'/x:\sigma]^d = \Gamma^d, t^d=t'^d \vdash \varphi^d[t'^d/(x:\sigma)^d]$.
Entonces, aplicando la regla (SUS)α:

$$\dfrac{\dfrac{\Gamma^d, t^d=t'^d \vdash \varphi^d[t'^d/(x:\sigma)^d]}{\Gamma^d \vdash \varphi^d[t^d/(x:\sigma)^d]}}{\text{hipótesis de inducción}}$$

(3) Reglas para $(\ni)$, (PD), $(\cup)$, $(\to)$, (ε), (AP) y (λ) en **A**.

Cada regla $(.)_{\mathbf{A}}$ se adecua directamente a su correspondiente regla $(.)\alpha$.

(4) Reglas para el operador μ.

$(I^{\circ})_{\mathbf{A}}$ Por el lema 5:

$\Gamma^d \vdash \neg E(\vec{t}, t)^d = \Gamma^d \vdash \neg E^d(\vec{t}^d, t^d)$ y
$\Gamma^d, (G[E/X](\vec{x}:d)\ni y:d)^d \vdash E^d(\vec{x}, y) = \Gamma^d,\ G^d[E^d/X](\vec{x}:d)\ni y:d \vdash E^d(\vec{x}, y)$.

Por definición:

$\Gamma^d \vdash (\neg(\mu X.G)(\vec{t})\ni t)^d = \Gamma^d \vdash \neg(\mu X.G^d)(\vec{t}^d)\ni t^d$.

Entonces, aplicando la regla $(I^{\circ})\alpha$:

$$\dfrac{\Gamma^d \vdash \neg(\mu X.G^d)(\vec{t}^d)\ni t^d}{\Gamma^d,\ G^d[E^d/X](\vec{x}:d)\ni y:d \vdash E^d(\vec{x}, y) \qquad\qquad \Gamma^d \vdash \neg E^d(\vec{t}^d, t^d)}$$

hipótesis de inducción

$(C^{\circ})_{\mathbf{A}}$ Por el lema 5:

$\Gamma^d \vdash (\exists k:n E'(k, \vec{t}, t))^d = \Gamma^d \vdash \exists(k:n)^d(\mathrm{nat}((k:n)^d) \wedge E'^d((k:n)^d, \vec{t}^d, t^d))$ y
$\Gamma^d, E'(k, \vec{t}, t)^d \vdash (G[\exists 1<kE'(1)/X](\vec{x})\ni y)^d = \Gamma^d, E'^d(k, \vec{x}, y) \vdash G^d[\exists 1<kE'^d(1)/X](\vec{x})\ni y$.

Por definición:

$\Gamma^d \vdash ((\mu X.G)(\vec{t})\ni t)^d = \Gamma^d \vdash (\mu X.G^d)(\vec{t}^d)\ni t^d$.

Entonces, aplicando la regla $(C^{\circ})\alpha$:

$$\Gamma^d \vdash (\mu X.G^d)(\vec{t}^d) \ni t^d$$

$$\Gamma^d,\ E'^d(k,\vec{x},y) \vdash G^d[\exists l<kE'^d(l)/X](\vec{x}) \ni y$$

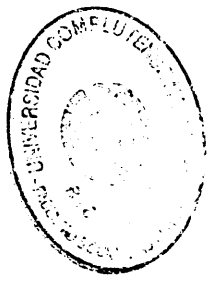
$$\Gamma^d \vdash \exists(k:n)^d(\mathrm{nat}((k:n)^d) \wedge E'^d((k:n)^d,\vec{t}^d,t^d))$$

hipótesis de inducción

Ahora, en $(C^*)\alpha$ aparece explicitamente $\mathrm{nat}((k:n)^d)$. ■

Obsérvese que la completitud admisible es un concepto más general que la aritmética, al que se puede acceder mediante la semántica no estándar de programas. De hecho, la completitud aritmética caracteriza la validez en una estructura aritmética fija $\mathfrak{D}$ en términos de probabilidad formal. Y la completitud admisible caracteriza la validez en una teoría de ALRF Φ fija en términos de probabilidad formal. Por lo que la completitud aritmética es un caso particular de la completitud admisible, cuando Φ se toma como $ETh(\mathfrak{D})$.

Por último, remarcar que el teorema 6 puede ser fácilmente adaptado a QDL y a otras lógicas de programas. Para QDL, esta afirmación está implícita en la inclusión en LRF, discutida en la sección 2.4.



CONCLUSIONES

Hemos presentado una lógica para programas funcionales, que reune conceptos fundamentales como μ-recursión, λ-abstracción, no determinismo, y funciones predefinidas, y hemos estudiado las vertientes estándar y no estándar.

Hemos demostrado que el sistema estándar LRF puede verse como un fragmento de $L\omega_1\omega$, bastante expresivo. En concreto, en este trabajo se expresan en LRF el μ-cálculo proposicional de Kozen y la lógica dinámica de primer orden para programas regulares de Harel.

Además, LRF admite un cálculo infinitario correcto y completo, del que se obtiene como corolario el teorema de Löwenheim-Skolem y que el problema de validez para LRF es Π^1_1-completo. También admite un cálculo finitario orientado a la sintaxis, aritméticamente correcto y completo. Basado en la capacidad de expresión de las estructuras aritméticas, en las que es posible definir, mediante una fórmula de primer orden, el menor punto fijo de un operador continuo.

En la vertiente no estándar, hemos presentado una primera lógica base NLRF, que se comporta como una lógica de primer orden, y que es un marco muy apropiado para el estudio no estándar de lógicas de programas. Trabajando en este marco, hemos desarrollado una lógica "admisible" ALRF más restrictiva, que excluye muchos de los modelos "arbitrarios" de NLRF pero mantiene el comportamiento de primer orden.

Para ALRF, hemos obtenido la caracterización semántica del menor punto fijo definible, que buscabamos al restringir NLRF. Y además, también hemos obtenido un cálculo orientado a la sintaxis correcto y completo para ALRF. Finalmente, hemos comprobado mediante ejemplos, que en ALRF se pueden demostrar terminación de programas y hemos demostrado que las derivaciones del cálculo para ALRF pueden traducirse a derivaciones del cálculo aritmético para LRF.

Nuestro objetivo principal en este trabajo ha sido proveer de un marco adecuado para lógicas de programas funcionales y clarificar la relación entre los enfoques estándar y no estándar. Actualmente, tenemos en mente algunas posibles lineas de continuación, para futuros trabajos. Por un lado, investigar más detalladamente la capacidad de expresión de LRF, intentando determinar que lógicas de programas conocidas pueden expresarse adecuadamente en LRF. Por otro lado, podía ser interesante extender LRF con más conceptos fundamentales para la programación funcional, como incluir tipos y funciones de orden superior. Finalmente, en otra dirección, NLRF y ALRF podían enriquecerse con constructores adecuados para describir diferentes conceptos de terminación e imparcialidad; cfr. [Har 79 y 84].

BIBLIOGRAFIA

[ANS 82] Andréka, H., Németi, I. and Sain, I.
"A Complete Logic for Reasoning about Programs via Nonstandard Model Theory, Parts I, II", TCS 17, 1982, 193-212 y 259-278.

[Apt 81] Apt, K.R.
"Ten Years of Hoare's Logic: a Survey-Part 1", ACM Trans. Prog. Lang. Syst. 3, 1981, 431-483.

[Ber 83] Berman, F.
"Nonstandard Models in Propositional Dynamic Logic", LNCS 148, 1983, 81-85.

[Bet 59] Beth, E.W.
"The Foundations of Mathematics", North-Holland, 1959.

[BF 85] Barwise, J. and Feferman, S. (Ed.)
"Model-theoretic Logics", Springer Verlag, 1985.

[CM 79] Cartwright, R. and McCarthy, J.
"Representation of Recursive Programs in First Order Logic", Stanford Art. Int. Memo AIM 324, 1979.

[Car 83] Cartwright, R.
"Non-standard Fixed Points in First Order Logic", LNCS 164, 1983, 86-100.

[Car 84] Cartwright, R.
"Recursive Programs as Definitions in First Order Logic", SIAM J. Comput. 13, 1984, 374-408.

[Coo 78] Cook, S.A.
"Soundness and Completeness of an Axiom System for Program Verification", SIAM J.Comput. 7, 1978, 70-90.

[EFT 84] Ebbinghaus, H.D., Flum, J. and Thomas, W.
"Mathematical Logic", Springer-Verlag, 1984.

[Eis 87] Eisenbach,S. (ed.)
"Functional Programming: Languages, Tools and Architectures", Ellis Horwood, 1987.

[End 72] Enderton, H.B.
"A Mathematical Introduction to Logic", Academic Press New York, 1972.

[Eng 67] Engeler, E.
"Algoritmic Properties of Structures", Math. Syst. Theory 1, 1967, 183-195.

[FL 77] Fischer, M.J. y Ladner, R.L.
"Propositional Modal Logic of Programs", Proc. 9 ACM Symp. on Theory of Computing, Boulder, Col., 1977.

[Flo 67] Floyd, R.W.
"Assigning Meaning to Programs", Proc. AMS Symp. Appl. Math. 19, Amer. Math. Soc., Providence, 1967, 19-31.

[Gil 85] Gil-Luezas, A.
"Un Estudio de las Lógicas Proposicionales de Programación: PDL, DPDL, SDPDL", Tesina de Licenciatura, Univ. Complutense de Madrid, 1985.

[Gil 89] Gil-Luezas, A.
"A Logic for Nondeterministic Functional Programs". Proceedings FCT'89. LNCS 380, 1989, 197-208.

[GHR 89] Gil-Luezas, A., Hortalá-González, M.T. y Rodriguez-artalejo, M.
"Standard Versus Nonstandard Semantics in Logics for Functional Programs", Informa Técnico del Dept. de Inf. y Aut. de la Univ. Complutense de Madrid, DIA/89/7.

[GB 83] Goguen, J.A. y Burstall, R.M.
"Introducing Institutions", LNCS 164, 1983, 221-255.

[Goe 85] Goerdt, A.
"Ein Hoare Kalkül für getypte λ-terme", Korrektheit, Vollständigkeit, Anwendungen, Dissertation, RWTH Aachen, 1985.

[Goe 87] Goerdt, A.
"Hoare Logic for Lambda-Terms as Basis of Hoare Logic for Imperative Languages", Procs. LICS'87, IEEE Computer Society Press 1987, 293-299.

[GMW 79] Gordon, M.J., Milner, A.J. and Wadsworth, C.P.
"Edinburgh LCF", LNCS 78 Springer Verlag, 1979.

[Haj 86] Hajek, P.
"A Simple Dynamic Logic", TCS 46, 1986, 239-259.

[Har 79] Harel, D.
"First Order Dynamic Logic", LNCS 68, 1979, Springer Verlag.

[Har 84] Harel, D.
"Dynamic Logic", D.Gabbay and F. Guenthner (ed.) Handbook of Philosophical Logic 2, Reidel P.C., 1984, 479-604.

[Hen 80] Henderson, P.
"Functional Programming: Application and Implementation", Prentice Hall, 1980.

[HMP 77] Harel, D., Meyer, A.R. y Pratt, V.R.
"Computability and Completeness in Logics of Programs", Proc. 9 ACM Symp. Theory of Comput., 1977, 261-268.

[Hoa 69] Hoare, C.A.R.
"An Axiomatic Basis for Computer Programming", Comm. of the ACM 12, 1969, 576-580.

[HR 85] Hortalá-González, M. T. y Rodriguez-Artalejo, M.
"Hoare's Logic for Nondeterministic Regular Programs: a Nonstandard Completeness Theorem", Proceedings ICALP 85. LNCS 194, 1985, 270-280.

[HR 89] Hortalá-González, M. T. y Rodriguez-Artalejo, M.
"Hoare's Logic for Nondeterministic Regular Programs: a Nonstandard Approach", TCS 68, n. 3, 1989, 277-302.

[Kei 71] Keisler, H.J.
"Model Theory for Infinitary Logic", North-Holland, 1971.

[Kle 67] Kleene, S.C.
"Mathematical Logic", John Wiley and Sons, 1967.

[Koz 83] Kozen,D.
"Results on the propositional μ-calculus", TCS 27, 1983, 333-354.

[KP 75] Kfoury, A.J. y Park, D.M.R.
"On the Termination of Program Schemas", Inf. and Control 29, 1975, 243-251.

[KT] Kozen, D. y Tiuryn, J.
"Logics of Programs", aparecerá como capítulo de Handbook of Theoretical Computer Science, (Managing Editor: Leeuwen, J.V.) y será publicado por North-Holland.

[Lan 63] Landin, P.J.
"The Mechanical Evaluation of Expressions", Comp. J. 6, 1963, 308-320.

[Lin 69] Lindström, P.
"On extensions of elementary logic", Theoria 35, 1969, 1-11.

[MaC 60] McCarthy, J.
"Recursive Functions of Symbolic Expressions and their Computation by Machine", Comm. Ass. Comp. Mach. 3, 1960, 184-195.

[Man 74] Manna, Z.
"Mathematical Theory of Computation", McGraw Hill, New York, 1974.

[Mir 71] Mirkowska, G.
"On Formalized Systems of Algorithmic Logic", Bull., Acad. Polon. Sci., Ser. Sci. Math. Astron. Phys. 19, 1971, 421-428.

[MM 83] Meyer, A.R. and Mitchell, J.C.
"Termination Assertions for Recursive Programs: Completeness and Axiomatic Definiability", Inf. and Control 56, 1983, 112-138.

[Mos 74] Moschovakis, Y.N.
"Elementary Induction on Abstract Structures", North-Holland, Amsterdam, 1974.

[Nau 66] Naur, P.
"Proof of Algorithms by General Snapshots", BIT 6, 1966, 310-316.

[Pas 86] Pasztor, A.
"Nonstandard Algorithmic and Dynamic Logic", J. Symbolic Comput. 2, 1986, 59-81.

[Pas 87] Pasztor, A.
"Recursive Programs and Denotational Semantics in Absolute Logics of Programs", Tech. Rep. FIU-SCS-87-1, Florida Int. Univ., 1987.

[Pey 87] Peyton Jones, S.L.
"The Implementation of Functional Programming Languages", Prentice Hall, 1987.

[Pnu 77] Pnueli, A.
"The Temporal Logic of Programs", Proc. 18 IEEE Symp. Found. Comput. Sci., 1977, 46-57.

[Pra 76] Pratt, V.R.
"Semantical Considerations on Floyd-Hoare Logic", Proc. 17 IEEE Symp. Found. Comput. Sci., 1976, 109-121.

[Roe 85] Roever, W. P. de
"The Quest for Compositionality", RUU-CS-85 2 Rijksuniv. Utrecht, 1985.

[Rog 67] Rogers, H.
"Theory of Recursive Functions and Effective Computability", McGraw-Hill, 1967.

[Sai 84] Sain, I.
"Structured Nonstandard Dynamic Logic", Zeit. Math. Log. Grund. Math. 30, 1984, 481-497.

[Sai 87] Sain, I.
"Total Correctness in Nonstandard Logics of Programs", TCS 50, 1987, 285-321.

[Sal 70] Salwicki, A.
"Formalized Algorithmic Languages", Bull. Acad. Polon. Sci., Ser. Sci. Math. Astron. Phys. 18, 1970, 227-232.

[Sho 67] Shoenfield, J.R.
"Mathematical Logic", Addison-Wesley Reading Mass., 1967.

[Smu 86] Smullyan, R.M.
"First-order Logic", Springer-Verlag, 1986.

[Sto 77] Stoy, J.
"Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory", MIT Press, 1977.

[Tar 55] Tarski, A.
"A Lattice Theoretical Fixpoint Theorem and its Applications", Pacific J. of Math. 5, 1955, 285-309.

[Thi 66] Thiele, H.
"Wissenschafts-Theoretische Untersuchungen in Algorithmischen Sprachen. Theorie der Graphschemata-Kalküle", Vab. Deutcher Verlag der Wissenschaften, Berlin, 1966.

[Tur 79] Turner, D.A.
"A New Implementation Technique for Applicative Languages", Softw. Pract. Exp. 9, 1979, 31-49.