UNA SEMÁNTICA CONSTRUCTIVA PARA LA LÓGICA DE SEGUNDO ORDEN AF2

FAVIO EZEQUIEL MIRANDA PEREA, LOURDES DEL CARMEN GONZÁLEZ HUESCA, AND ARACELI LILIANA REYES CABELLO

A la memoria de José Alfredo Amor Montaño

Resumen. La lógica AF2 es un sistema de deducción natural con codificación de pruebas para la lógica de predicados de segundo orden cuya característica principal radica en permitir el razonamiento ecuacional mediante la llamada igualdad de Leibniz. El objetivo de este artículo es presentar a detalle dicha lógica así como darle una semántica constructiva, en el sentido de la llamada interpretación de Brouwer-Heyting-Kolmogorov, donde el significado de una fórmula se interpreta como un conjunto de pruebas formales, las cuales se representan mediante términos del cálculo lambda.

1. Introducción

La lógica de predicados de segundo orden es una extensión de la bien conocida lógica de primer orden que permite la cuantificación no sólo sobre individuos sino también sobre propiedades de los mismos. En particular la lógica conocida como aritmética funcional de segundo orden, denotada AF2, sobre la que trata este trabajo es un sistema deductivo para la lógica de segundo orden desarrollado por Leivant y Krivine en [5, 4], cuyas características principales son el incluir un sistema de códificación de pruebas y un mecanismo de razonamiento ecuacional los cuales permiten la extracción de programas a partir de especificaciones lógicas mediante el llamado paradigma de programación con pruebas de Krivine-Parigot [3, 10]. Este método de síntesis de programas se basa fuertemente en una noción de tipo basada en una semántica clásica (ver [3, 10]) o bien en una interpretación sintáctica de realización (ver [3, 7]). Además de presentar a cierto detalle a la lógica AF2, el propósito de este artículo es desarrollar en su lugar y de manera rigurosa una semántica para AF2 similar a la esbozada en [11], la cual se basa fuertemente en la semántica intensional para la lógica constructiva conocida como interpretación de Brouwer-Heyting-Kolmogorov (BHK).

El desarrollo propuesto es el siguiente: mientras que la definición de lógica de primer orden está universalmente determinado, para el caso de segundo orden no existe realmente un consenso, por lo que debemos puntualizar la sintaxis particular de lo que entenderemos aquí por lógica de segundo orden, esto se hará en la sección 2. La interpretación BHK se presenta en la sección 3 mientras que el sistema de deducción natural sobre el que se basa nuestra semántica constructiva se desarrolla en la sección 4. El cálculo lambda fungirá como sistema de codificación de pruebas para la lógica y se discute en la sección 5. El sistema deductivo AF2 para la lógica

2010 *Mathematics Subject Classification.* Primary 03B15, 03B40, 03F07; Secondary 68N18.
Agradecemos el apoyo del proyecto PAPIIT-UNAM IN108810.

de segundo orden se define en la sección 6 donde también se muestran algunos ejemplos del método de extracción de programas. La principal aportación de este artículo es la semántica constructiva desarrollada en la sección 7 siendo el resultado principal el teorema de correctud. La existencia de ciertos modelos requeridos por dicho teorema se garantiza en la sección 8. Nuestra exposición termina mostrando la relación de nuestros modelos con los modelos clásicos (sección 9), así como dando algunos comentarios finales en la sección 10.

Iniciamos este artículo advirtiendo que si bien tratamos de ser autocontenidos, suponemos conocidos todos los conceptos acerca de la lógica de primer orden involucrados en la exposición y cuya definición se omite. Cualquier duda puede consultarse por ejemplo en [2].

2. Sintaxis de la lógica de segundo orden

A continuación presentamos detalladamente la sintaxis de la lógica de segundo orden.

- *Términos*: se definen de la manera usual a partir de variables de primer orden y de una signatura fija Σ que incluye símbolos de función f de un índice o número de argumentos fijo.
$$r, s, t ::= x \mid f(t_1, \ldots, t_n)$$
Obsérvese que esta definición incluye el caso en que el índice de un símbolo de función f sea cero, en cuyo caso se trata de un símbolo de constante.
- *Fórmulas*: se definen de la manera usual como sigue:
$$A, B, C ::= \mathcal{P}(\vec{t}) \mid A \to B \mid A \wedge B \mid A \vee B \mid \forall x A \mid \forall X A$$
donde las variable x y X se consideran ligadas en las fórmulas $\forall x A$ y $\forall X A$ respectivamente. En adelante adoptamos algunas veces la notación de punto en un cuantificador para declarar que su alcance se considera lo más lejos posible sintácticamente, por ejemplo la proposición $\forall x.A \to B$ significa $\forall x(A \to B)$ y no $(\forall x A) \to B$.
- *Predicados*: los predicados pueden ser variables de segundo orden X, símbolos de predicado P de una signatura fija Σ o bien predicados por comprensión $\mathcal{F}$.
$$\mathcal{P} ::= X \mid P \mid \mathcal{F}$$
Un predicado por comprensión es una expresión de la forma $\mathcal{F} =_{def} \{\vec{x} \,|\, A\}$ donde A es una fórmula y $\vec{x}$ es un vector de variables de primer orden, usualmente libres en A. El índice o número de argumentos de $\mathcal{F}$ se define como la longitud del vector $\vec{x}$. Esta clase de predicados representa al conjunto de tuplas $\vec{t}$ tales que $A[\vec{x} := \vec{t}]$ es válida. A. En particular, para cualquier tupla de términos $\vec{s}$, la fórmula atómica $\mathcal{F}\vec{s}$ se define como la fórmula $A[\vec{x} := \vec{s}]$. La operación de sustitución de variables por términos se define más adelante.

  Es importante observar que los predicados y la fórmula se definen simultaneamente de manera inductiva. Más aún, como es bien sabido, la lógica de segundo orden es suficientemente expresiva y permite definir los conectivos $\wedge, \vee$, así como los cuanticadores existenciales, mediante $\forall$ e $\to$. En nuestro caso los consideramos como primitivos porque las construcciones semánticas para estos conectivos, que presentamos más adelante, son de interés general.

Por último se observa que la negación está ausente, por lo que en realidad estamos tratando con una lógica minimal, esto se debe a que nos interesan los aspectos constructivos de la lógica, los cuales desaparecen con la negación, para una discusión de esto véase [9].

2.1. Sustitución. Dado que contamos con dos clases de variables y en ambas está presente un mecanismo de ligado es conveniente definir con cuidado las operaciones de sustitución. Los conjuntos $Var(t)$ de variables de primer orden de un término t, $Var_2(\mathcal{P})$ de variables de segundo orden de un predicado $\mathcal{P}$ y $FV(A)$ de variables libres de primer y segundo orden de una fórmula A se definen de la manera natural.

Definición 2.1. La sustitución de variables de primer orden $\vec{x} = x_1, \ldots, x_n$ por términos $\vec{t} = v_1, \ldots, v_n$ en un término r, denotada $r[\vec{x} := \vec{t}]$ se define como la sustitución textual de cada presencia de x_i por t_i en r.

Definición 2.2. La *sustitución* de variables de primer orden $\vec{x}$ por términos $\vec{t}$ en la fórmula A, denotada $A[\vec{x} := \vec{t}]$ se define recursivamente como sigue:

$$\begin{array}{rcl}
\mathcal{Q}(r_1, \ldots, r_m)[\vec{x} := \vec{t}\,] & = & \mathcal{Q}\big(r_1[\vec{x} := \vec{t}\,], \ldots, r_m[\vec{x} := \vec{t}\,]\big) \\
(A \wedge B)[\vec{x} := \vec{t}\,] & = & \big(A[\vec{x} := \vec{t}\,] \wedge B[\vec{x} := \vec{t}\,]\big) \\
(A \vee B)[\vec{x} := \vec{t}\,] & = & \big(A[\vec{x} := \vec{t}\,] \vee B[\vec{x} := \vec{t}\,]\big) \\
(A \to B)[\vec{x} := \vec{t}\,] & = & \big(A[\vec{x} := \vec{t}\,] \to B[\vec{x} := \vec{t}\,]\big) \\
(\forall x A)[\vec{x} := \vec{t}\,] & = & \forall x\big(A[\vec{x} := \vec{t}\,]\big) \text{ si } x \notin \vec{x} \cup Var(\vec{t}) \\
(\forall X A)[\vec{x} := \vec{t}\,] & = & \forall X\big(A[\vec{x} := \vec{t}\,]\big)
\end{array}$$

La condición en el caso de las cuantificaciones de primer orden siempre puede cumplirse al renombrar la variable ligada x en la fórmula original. Esto no causa problema alguno pues consideramos iguales a dos fórmulas que difieren únicamente en los nombres de sus variables ligadas. Esta convención se conoce como α-equivalencia, por ejemplo $\forall x \mathcal{P}(x)$ y $\forall y \mathcal{P}(y)$ son fórmulas α-equivalentes. Obsérvese que si queremos que la operación de sustitución defina a una función total, en lugar de fórmulas, es necesario manipular clases de α-equivalencia de fórmulas, situación que no hacemos explícita. Lo mismo sucede para el caso de la sustitución en cuantificaciones de segundo orden definida más abajo.

Definición 2.3. La *sustitución* de una variable de segundo orden X por el predicado $\mathcal{P}$ en el predicado $\mathcal{Q}$, denotada $\mathcal{Q}[X := \mathcal{P}]$ se define como sigue:

$$\begin{array}{rcl}
X[X := \mathcal{P}\,] & = & \mathcal{P} \\
Y[X := \mathcal{P}\,] & = & Y \\
P[X := \mathcal{P}\,] & = & P \\
\{\vec{x} \mid A\}[X := \mathcal{P}\,] & = & \{\vec{x} \mid A[X := \mathcal{P}]\}
\end{array}$$

Definición 2.4. La *sustitución* de una variable de segundo orden X por el predicado $\mathcal{P}$ en la fórmula A, denotada $A[X := \mathcal{P}]$ se define recursivamente como sigue:

$$\begin{array}{rcl}
\mathcal{Q}(r_1,\ldots,r_m)[X := \mathcal{P}\,] & = & \mathcal{Q}[X := \mathcal{P}](r_1,\ldots,r_m) \\
(A \to B)[X := \mathcal{P}\,] & = & \big(A[X := \mathcal{P}\,] \to B[X := \mathcal{P}\,]\big) \\
(A \wedge B)[X := \mathcal{P}\,] & = & \big(A[X := \mathcal{P}\,] \wedge B[X := \mathcal{P}\,]\big) \\
(A \vee B)[X := \mathcal{P}\,] & = & \big(A[X := \mathcal{P}\,] \vee B[X := \mathcal{P}\,]\big) \\
(\forall x A)[X := \mathcal{P}\,] & = & \forall x\big(A[X := \mathcal{P}\,]\big) \\
(\forall Y A)[X := \mathcal{P}\,] & = & \forall Y\big(A[X := \mathcal{P}\,]\big) \text{ si } Y \notin \{X\} \cup Var_2(\mathcal{P})
\end{array}$$

A continuación mostramos algunos ejemplos de la expresividad de la lógica de segundo orden:

Ejemplo 2.1. *Los conectivos lógicos $\wedge, \vee$, los cuantificadores existenciales así como los números naturales y su principio de inducción se definen como sigue:*

- *La conjunción: sean A, B dos fórmulas y X una variable de segundo orden de índice 0 que no figura en A ni en B.*

$$A \wedge B =_{def} \forall X.(A \to B \to X) \to X$$

- *La disyunción: sean A, B dos fórmulas y X una variable de segundo orden de índice 0 que no figura en A ni en B.*

$$A \vee B =_{def} \forall X.(A \to X) \to (B \to X) \to X$$

- *La cuantificación existencial de primer orden: sean A una fórmula y X una variable de segundo orden de índice cero tal que $X \notin FV(A)$.*

$$\exists x A =_{def} \forall X\big(\forall x(A \to X) \to X\big)$$

- *La cuantificación existencial de segundo orden: sean A una fórmula y Y una variable de segundo orden de índice cero tal que $Y \notin FV(A)$.*

$$\exists X A =_{def} \forall Y\big(\forall X(A \to Y) \to Y\big)$$

- *Los números naturales: se definen como el predicado más pequeño $\mathbb{N}$ cerrado bajo el cero 0 y la función sucesor s. Esta idea se captura con el siguiente predicado por comprensión:*

$$\mathbb{N}(y) =_{def} \{y | \forall X.\forall x(X(x) \to X(s(x))) \to X(0) \to X(y)\}$$

- *Postulado de inducción de Peano:*

$$\forall X.\forall x(X(x) \to X(s(x))) \to X(0) \to \forall x.\mathbb{N}(x) \to X(x)$$

Para una explicación del origen de esta definiciones sugerimos consultar [9].

2.2. La igualdad de Leibniz. En lógica constructiva el símbolo de igualdad puede ser considerado primitivo aunque lo usual para los sistemas de primer orden y para aplicaciones, es tratarlo como un símbolo de predicado binario distinguido, el cual satisface ciertos axiomas no lógicos, además de las propiedades de reflexividad, simetría y transitividad. En el caso de la lógica de segundo orden la igualdad es definible y permite el razonamiento ecuacional directo como veremos en esta sección.

Definición 2.5. Dados dos términos r y s definimos la relación de igualdad de Leibniz, denotada $r = s$, como la cerradura universal de primer orden de la fórmula

$$\forall X.X(r) \to X(s).$$

Cualquier fórmula de la forma $r = s$ se llamará ecuación. Se observa que una ecuación es una fórmula con una única variable ligada de segundo orden y en donde las variables de primer orden están libres, aunque se consideran ligadas universalmente.

Más adelante mostraremos que esta definición de igualdad cumple las propiedades esperadas. En particular la lógica AF2 se sirve ampliamente del razonamiento ecuacional provisto por la igualdad de Leibniz.

A continuación discutimos la semántica constructiva intensional que será la guia para definir una semántica formal.

3. La interpretación BHK

La lógica clásica se basa en la noción primitiva de *verdad*. La verdad de una fórmula es una propiedad absoluta de la misma, en el sentido de que no depende de ningún razonamiento, entendimiento o acción. Un enunciado declarativo libre de ambigüedades es verdadero o no verdadero, es decir, falso, independientemente de si conocemos su valor de verdad, o de si lo probamos o verificamos en cualquier forma posible. Por otra parte, en muchas aplicaciones no basta con saber cual es la solución a un problema, sino que necesitamos construirla, de manera que nos gustaría separar los métodos de prueba que proporcionan soluciones, de aquellos que no lo hacen. Por lo tanto, pragmáticamente tiene sentido considerar un enfoque constructivo de la lógica. Esta lógica, llamada constructiva o intuicionista se basa en la idea de que la verdad significa demostrabilidad (Brouwer 1907,1918). Para entender la lógica intuicionista, debemos olvidar la noción clásica (tarskiana) de verdad. Los juicios acerca de una fórmula ya no se basan en un valor de verdad asignado a ella, sino en la habilidad para construirla mediante una prueba explícita. Como consecuencia uno no debe definir los conectivos mediante tablas de verdad. En su lugar, debemos explicar el significado de fórmulas compuestas en términos de sus construcciones. Tal explicación se da de manera intuitiva mediante la llamada interpretación de Brouwer-Heyting-Kolmogorov (BHK) que es reconocida ampliamente como la semántica intensional de la lógica intuicionista.

Definición 3.1. La *interpretación BHK* se define recursivamente como sigue:

- Una prueba de una variable proposicional p se supone conocida y dada por un contexto previamente definido.
- Una prueba de $A \wedge B$ es un par $\langle p, q\rangle$ donde p es una prueba de A y q es una prueba de B.
- Una prueba de $A \vee B$ es una prueba p de A o una prueba q de B, junto con una etiqueta que indique cual de las dos fórmulas se está probando.
- Una prueba de $A \to B$ es un método funcional f que transforma cualquier prueba p de A en una prueba $f(p)$ de B.

Ejemplo 3.1. *Algunos ejemplos son:*

- *La función identidad id es una prueba de $A \to A$, pues dada cualquier prueba p de A, $id(p) = p$ es una prueba de A.*
- *Si f es una prueba de $A \to B$ y g es una prueba de $B \to C$ entonces $g \circ f$ es una prueba de $A \to C$. Por lo tanto La función F, dada por $F(f)(g) = g \circ f$ es una prueba de $(A \to B) \to (B \to C) \to A \to C$*

- *La función swap definida como* $swap(\langle p, q\rangle) = \langle q, p\rangle$ *es una prueba de* $A \wedge B \to B \wedge A$.
- *Una prueba de* $A \to A \vee B$ *está dada por la función* $f(p) = \mathsf{inl}\, p$. *Obsérvese que* inl *es la etiqueta que indica que la prueba de* $A \vee B$ *es en realidad una prueba de* A.

Observamos que la discusión anterior sólo se refiere a la lógica proposicional, la versión más común de la interpretación BHK para los cuantificadores es la siguiente:

- Una prueba de $\forall x A$ es un método funcional que transforma a cada objeto a en una prueba de $A[x := a]$.
- Una prueba de $\forall X A$ es un método funcional que transforma a cada relación R en una prueba de $A[X := R]$.

sin embargo nosotros usaremos una versión donde los cuantificadores no tienen contenido computacional (ver [1]), lo cual significa, para nuestros propósitos, lo siguiente:

- Una prueba de $\forall x A$ es una prueba de $A(a)$ que no depende de ninguna propiedad del objeto a, es decir, una prueba de A paramétrica en a.
- Una prueba de $\forall X A$ es una prueba de $A(R)$ que no depende de ninguna propiedad de la relación R, es decir, una prueba de A paramétrica en R.

La semántica intuitiva anterior consta de afirmaciones metalógicas pero puede formalizarse de diversas maneras, siendo las más usadas la semántica de álgebras de Heyting y la semántica de mundos posibles de Kripke, ambas equivalentes. Sin embargo en estas semánticas, el concepto de prueba no está involucrado directamente en la semántica. En contraste, nuestro objetivo es presentar una semántica donde la interpretación $\mathcal{I}(A)$ de una fórmula A será un conjunto de pruebas que incluya a las pruebas de A. Es decir, una interpretación es una función $\mathcal{I} : \mathsf{Form} \to \mathsf{Prb}$ donde Prb es el conjunto de (representaciones de) pruebas de la lógica y tal que si p es una prueba de A entonces se cumpla que $p \in \mathcal{I}(A)$.

Para lograr este objetivo primero tenemos que dar una noción formal de prueba, a esto nos dedicamos en la siguiente sección.

4. Deducción natural

Como sistema deductivo para la lógica de segundo orden escogemos la deducción natural con hipótesis localizadas, es decir, con contextos de fórmulas.

Definición 4.1. Un *contexto* es un conjunto de fórmulas $\Gamma = \{A_1, \ldots, A_n\}$.

En el manejo de contextos convenimos en omitir las llaves de conjunto así como la operación de unión escribiendo Γ, Δ en vez de $\Gamma \cup \Delta$ y Γ, A en vez de $\Gamma \cup \{A\}$. Mas aún, en un contexto de la forma Γ, A suponemos que A no figura en Γ. El conjunto de variables libres del contexto $\Gamma = \{A_1, \ldots, A_n\}$ se define como $FV(\Gamma) =_{def} FV(A_1) \cup \ldots \cup FV(A_n)$.

Definición 4.2. Un *secuente* es una expresión de la forma $\Gamma \vdash A$ donde Γ es un contexto y A es una fórmula. En particular el secuente $\vdash A$ significa $\varnothing \vdash A$.

El secuente $\Gamma \vdash A$ expresa la relación de derivabilidad de la fórmula A a partir de las hipótesis dadas en Γ. Esta relación se define formalmente mediante las siguientes reglas de inferencia entre secuentes.

Definición 4.3. La relación de *derivabilidad* $\Gamma \vdash A$ se define recursivamente como sigue:

- Regla de inicio:
$$\frac{}{\Gamma, A \vdash A}\ (Hip)$$
- Implicación:
$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ (\to I) \qquad \frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}\ (\to E)$$
- Conjunción:
$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ (\wedge I)\ \ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ (\wedge_1 E)\ \ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ (\wedge_2 E)$$
- Disyunción
$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ (\vee_1 I)\ \ \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ (\vee_2 I)$$
$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ (\vee E)$$
- Cuantificación universal de primer orden:
$$\frac{\Gamma \vdash A \quad x \notin FV(\Gamma)}{\Gamma \vdash \forall x A}\ (\forall I) \qquad \frac{\Gamma \vdash \forall x A}{\Gamma \vdash A[x := t]}\ (\forall E)$$
- Cuantificación universal de segundo orden:
$$\frac{\Gamma \vdash A \quad X \notin FV(\Gamma)}{\Gamma \vdash \forall X A}\ (\forall^2 I) \qquad \frac{\Gamma \vdash \forall X A}{\Gamma \vdash A[X := \mathcal{P}]}\ (\forall^2 E)$$

Una derivación de un secuente particular se define como sigue:

Definición 4.4. Una *derivación* del secuente $\Gamma \vdash A$ es una sucesión finita de secuentes $\Gamma_1 \vdash A_1, \ldots, \Gamma_n \vdash A_n$ tal que:

- $\Gamma_i \vdash A_i$ es instancia de la regla (Hip) ó
- $\Gamma_i \vdash A_i$ es conclusión de alguna regla de inferencia tal que las premisas necesarias figuran antes en la sucesión.
- $\Gamma \vdash A$ es el último elemento de la sucesión.

Obsérvese que a cada conectivo o cuantificador le corresponden reglas que lo introducen, denotadas con I, así como reglas que lo eliminan, denotadas con E. Esta simetría o dualidad en las reglas es de gran importancia y proporciona un determinismo en la relación de derivabilidad. Cada fórmula compuesta puede ser derivada usando una única regla de introducción. Así mismo, la información de cada fórmula compuesta puede ser utilizada para derivar otras fórmulas mediante una única regla de eliminación.

La lógica es monótona como lo asegura la siguiente

Proposición 4.1. *La siguiente regla de inferencia es derivable:*
$$\frac{\Gamma \vdash A}{\Gamma, B \vdash A}\ (Mon)$$

Demostración. Inducción sobre $\vdash$. □

4.1. Reglas para la igualdad. En este apartado mostramos que la igualdad de Leibniz cumple las propiedades usuales de la igualdad por lo que permite el razonamiento ecuacional común en matemáticas.

Proposición 4.2. *La igualdad de Leibniz es una relación de equivalencia compatible con funciones , es decir, se cumple lo siguiente para cualquier contexto* Γ*:*

- *Reflexividad* (Ref)*:* $\Gamma \vdash r = r$
- *Transitividad* (Trn)*: Si* $\Gamma \vdash r = s$ *y* $\Gamma \vdash s = t$ *entonces* $\Gamma \vdash r = t$
- *Simetría* (Sim)*: Si* $\Gamma \vdash r = s$ *entonces* $\Gamma \vdash s = r$
- *Compatibilidad* (Com)*: Si* $\Gamma \vdash r_i = s_i$*, para toda* $1 \leq i \leq n$ *entonces* $\Gamma \vdash f(r_1, \ldots, r_n) = f(s_1, \ldots, s_n)$

Demostración. Es claro que se cumplen la reflexividad y la transitividad. Para mostrar la simetría definimos el predicado por comprensión $\mathcal{F} =_{def} \{x | x = r\}$, donde $x \notin Var(r)$, y mostramos que $\Gamma \vdash \mathcal{F}(s)$.

1.	$\Gamma \vdash \forall X.X(r) \to X(s)$	(Hip) y def. de $r = s$.
2.	$\Gamma \vdash \mathcal{F}(r) \to \mathcal{F}(s)$	$(\forall^2 E),\ 1,\ [X := \mathcal{F}]$
3.	$\Gamma \vdash \mathcal{F}(r)$	$((\text{Ref}))$
4.	$\Gamma \vdash \mathcal{F}(s)$	$(\to E), 2, 3$

Para la compatibilidad: mostramos el caso para $n = 2$, el caso general resulta de una sencilla inducción. Sean $\mathbb{E} = \{r_1 = s_1, r_2 = s_2\}, \mathcal{F}_1 =_{def} \{x | f(r_1, r_2) = f(x, r_2)\}$, donde $x \notin Var(r_2)$, $\mathcal{F}_2 =_{def} \{y | f(s1, r2) = f(s1, y)\}$ donde $y \notin Var(s_1)$.

1.	$\Gamma \vdash \mathcal{F}_1(r_1)$	(Ref)
2.	$\Gamma \vdash \forall X.X(r_1) \to X(s_1)$	(Hip) y def. de $r_1 = s_1$.
3.	$\Gamma \vdash \mathcal{F}_1(r_1) \to \mathcal{F}_1(s_1)$	$(\forall^2 E),\ 2,\ [X := \mathcal{F}_1]$
4.	$\Gamma \vdash \mathcal{F}_1(s_1)$	$(\to E),\ 1, 3$
5.	$\Gamma \vdash \mathcal{F}_2(r_2)$	(Ref)
6.	$\Gamma \vdash \forall X.X(r_2) \to X(s_2)$	(Hip) y def. de $r_2 = s_2$.
7.	$\Gamma \vdash \mathcal{F}_2(r_2) \to \mathcal{F}_2(s_2)$	$(\forall^2 E),\ 6,\ [X := \mathcal{F}_2]$
8.	$\Gamma \vdash \mathcal{F}_2(s_2)$	$(\to E),\ 5, 7$
9.	$\Gamma \vdash f(r_1, r_2) = f(s_1, s_2)$	$(\text{Trn})),\ 4, 8$

□

El razonamiento ecuacional común se formaliza mediante la siguiente relación de inferencia.

Definición 4.5. Dado un conjunto de ecuaciones $\mathbb{E}$, decimos que la ecuación $r = s$ se deriva a partir de $\mathbb{E}$, lo cual denotamos con $\mathbb{E} \rhd r = s$, si y sólo si existe una derivación mediante las siguientes reglas de inferencia:

- (Eq-Ax). $\mathbb{E} \rhd r = s$, si $r = s$ es un caso particular de una ecuación en $\mathbb{E}$, es decir, $r = s$ se obtiene de una ecuación de $\mathbb{E}$ instanciando sus variables.
- Reflexividad, simetria, transitividad o compatibilidad con funciones:

$$\frac{}{\mathbb{E} \rhd r = r}\ (\text{Eq-Ref}) \qquad \frac{\mathbb{E} \rhd r = s}{\mathbb{E} \rhd s = r}\ (\text{Eq-Sim})$$

$$\frac{\mathbb{E} \rhd r = s \quad \mathbb{E} \rhd s = t}{\mathbb{E} \rhd r = t}(\text{Eq-Trn})$$

$$\frac{\mathbb{E} \rhd r_1 = s_1 \quad \ldots \quad \mathbb{E} \rhd r_n = s_n}{\mathbb{E} \rhd f(r_1, \ldots, r_n) = f(s_1, \ldots, s_n)}(\text{Eq-Fun})$$

La siguiente proposición muestra que el razonamiento ecuacional usual es permitido por igualdad de Leibniz, puesto que las reglas de la definición 4.5 son derivables en la lógica de segundo orden.

Proposición 4.3. *Si* $\mathbb{E} \rhd r = s$ *entonces* $\mathbb{E} \vdash r = s$.

Demostración. Inducción sobre $\rhd$, utilizando la proposición 4.2 □

5. El cálculo λ

El cálculo λ fue inventado en la decada de los 1930 por Alonso Church. Este cálculo fue inicialmente parte de un sistema más complejo que pretendia ser un fundamento matemático para la lógica. En 1935 Kleene y Rosser, estudiantes de Church, descubrieron que dicho sistema era inconsistente. Sin embargo el subsistema de términos conocido hoy como cálculo λ fue desde entonces estudiado independientemente como un modelo de cómputo. Este cálculo se ocupa de funciones únicamente, en particular modela funciones que toman otras funciones como argumento o que devuelven funciones como resultados. En términos de lenguajes de programación el cálculo λ es un prototipo extremadamente simple de un lenguaje funcional puro de orden superior. Esta sección da una muy breve introducción al cálculo λ sin tipos, para un estudio profundo del mismo sugerimos revisar el libro [13].

La sintaxis del cálculo λ puro es la siguiente:

$$e ::= x \mid \lambda x e \mid e e$$

Es decir, un λ-término es una expresión de alguna de las siguientes formas:

- Variables: la variable x representa a una función cualquiera.
- *Abstracción lambda*: Una expresión de la forma $\lambda x e$ se llama *abstracción lambda* o simplemente *abstracción*. La idea aquí es que $\lambda x e$ indica una abstracción de los valores particulares de la variable x en la expresión e lo cual causa que e se considere una función de x, es decir, la expresión $\lambda x e$ define anónimamente a la función la función $x \mapsto e$ que asocia a cada valor x la expresión e . El punto en una abstracción es, como en el caso de los cuantificadores de la lógica, un uso particular de la llamada sintaxis de orden superior en lenguajes de programación y denota el ligado de x en e, además indica que el alcance del ligado para x se extiende a la derecha tanto como sea posible. Por ejemplo, $\lambda x.xy$ significa $\lambda x.(xy)$ y no $(\lambda x.x)y$. Para facilitar la escritura de abstracciones usaremos la siguiente convención
$$\lambda x_1 x_2 \ldots x_n.e =_{def} \lambda x_1.\lambda x_2.\ldots.\lambda x_n.e$$
- *Aplicación*: la expresión $e_1 e_2$ representa a la aplicación de la función e_1 al argumento e_2. La aplicación se asocia a la izquierda de manera que $e_1\, e_2\, e_3$ significa $(e_1\, e_2)\, e_3$ y nunca $e_1\, (e_2\, e_3)$. Obsérvese que no hay un operador explícito para la aplicación, ésta se denota simplemente mediante la escritura secuencial de dos expresiones.

Es importante observar que los λ-términos son las únicas expresiones válidas en el cálculo λ y que el operador λ se comporta de manera similar a un cuantificador en la lógica usual, es decir, en la abstracción $\lambda x.e$ el operador λ liga a la variable x en la expresión e por lo que se utiliza la noción de variables libres y ligadas usual en

lógica. En particular se tiene que el conjunto de variables libres de una abstracción se define como $FV(\lambda xe) = FV(e) \setminus \{x\}$. Por ejemplo en $(\lambda xy.xzy)(\lambda z.zy)u$ las dos presencias de x son ligadas así como las dos primeras presencias de y y las dos últimas presencias de z mientras que el resto de las presencias de cada variable son libres.

5.1. Semántica Operacional. El sistema de λ-términos es dinámico, a diferencia del sistema de términos de una lógica, en el sentido de que los términos operan entre si de acuerdo a una relación binaria llamada semántica operacional. Esta semántica denotada $\rightarrow_\beta$ se define como la cerradura de la siguiente regla, conocida universalmente como β-reducción, bajo todos los constructores de término.

$$(\lambda xe)t \mapsto_\beta e[x := t]$$

Lo que esta regla nos dice es que la acción de aplicar la función $\lambda x.e$ al argumento t consiste en sustituir el parámetro x por el término t en el término e. Se observa entonces que todo paso de evaluación es simplemente una substitución definida de la manera usual mediante el uso de la α-equivalencia (renombre de variables ligadas) para evitar la captura de variables libres:

- $x[x := r] = r$.
- $y[x := r] = y$ si $x \neq y$.
- $(ts)[x := r] = t[x := r]s[x := r]$.
- $(\lambda yt)[x := r] = \lambda y.t[x := r]$ donde s.p.g. $y \neq x$ y $y \notin FV(r)$.

Dos relaciones de importancia, derivadas de $\rightarrow_\beta$ son su cerradura reflexiva-transitiva, denotada $\rightarrow_\beta^\star$ y su cerradura reflexiva-simetrica-transitiva, denotada $=_\beta$ y llamada β-equivalencia.

Ejemplo 5.1. *Los siguientes son ejemplos de β-reducciones y β-equivalencia.*

- $(\lambda xx)y \rightarrow y$
- $(\lambda xy)t \rightarrow y$
- $(\lambda x.x(xy))t \rightarrow t(ty)$
- $(\lambda x.x(\lambda xx))(ur) \rightarrow (ur)(\lambda xx)$
- $(\lambda x.(\lambda y.yx)z)v \rightarrow (\lambda y.yv)z$
- *Si* $I =_{def} \lambda xx$ *entonces* $III \rightarrow^\star I$ *e* $III =_\beta II$
- *Si* $S =_{def} \lambda xyz.xz(yz)$ *y* $K =_{def} \lambda xy.x$ *entonces* $SKK \rightarrow^\star I$ *y* $SKK =_\beta II$.

El cálculo lambda es lo suficientemente poderoso para representar tipos de datos como booleanos y números naturales. Veamos de que manera.

Ejemplo 5.2 (Booleanos)**.** *Considerense los siguientes λ-términos.*

- $\mathsf{true} := \lambda x\lambda y.x$
- $\mathsf{false} := \lambda x\lambda y.y$
- $\mathsf{test} := \lambda b.\lambda t\lambda e.bte$
- $\mathsf{not} := \lambda z.z\,\mathsf{false}\,\mathsf{true}$
- $\mathsf{and} := \lambda x\lambda y\lambda z\lambda w.x(yzw)w$

Es facil cerciorarse de lo siguiente:

- $\mathsf{test}\,\mathsf{true}\,e_1\,e_2 \rightarrow^\star e_1$
- $\mathsf{test}\,\mathsf{false}\,e_1\,e_2 \rightarrow^\star e_2$

por lo que el término test *se comporta como el condicional booleano*

- $\mathsf{not}\,\mathsf{true} \rightarrow^\star \mathsf{false}$

- $\mathsf{not}\,\mathsf{false} \to^\star \mathsf{true}$
- $\mathsf{and}\,\mathsf{false}\,b \to^\star \mathsf{false}$
- $\mathsf{and}\,\mathsf{true}\,b \to^\star b$

Ejemplo 5.3 (Números naturales (numerales de Church))**.** *Los λ-términos conocidos como numerales de Church se definen como sigue*

- $\bar{0} := \lambda s.\lambda z.z$
- $\bar{1} := \lambda s.\lambda z.sz$
- $\bar{2} := \lambda s.\lambda z.s(sz)$
- $\bar{3} := \lambda s.\lambda z.s(s(sz))$
- $\bar{n} := \lambda s.\lambda z.\underbrace{s(\ldots(s}_{n\ veces}\,z)\ldots)$

Las operaciones aritméticas básicas se definen como sigue, y la semántica operacional del cálculo lambda permite verificar que se comportan de la forma esperada.

- $\mathsf{suc} := \lambda n.\lambda s.\lambda z.s(nsz)$ *tal que*
$$\forall n \in \mathbb{N}\big(\mathsf{suc}\,\bar{n} \to^\star \overline{n+1}\big)$$
- $\mathsf{suma} := \lambda n\lambda m.m(\lambda u\lambda f\lambda z.f(ufz))n$ *tal que*
$$\forall n, m \in \mathbb{N}\big(\mathsf{suma}\,\bar{n}\,\bar{m} \to^\star \overline{n+m}\big)$$
- $\mathsf{prod} := \lambda m.\lambda n.m(\mathsf{suma}\ n)\bar{0}$ *tal que*
$$\forall n, m \in \mathbb{N}\big(\mathsf{prod}\,\bar{n}\,\bar{m} \to^\star \overline{n*m}\big)$$
- $\mathsf{iszero} := \lambda m.m(\lambda x.\mathsf{false})\mathsf{true}$ *tal que*
$$\mathsf{iszero}\,\bar{0} \to^\star \mathsf{true} \qquad \forall n \in \mathbb{N}\big(\mathsf{iszero}\,(\overline{n+1}) \to^\star \mathsf{false}\big)$$

Acerca de como surgen estas definiciones podemos mencionar que si bien son conocidas desde la introducción del cálculo lambda en la decada de los 1930, pueden justificarse mediante el mecanismo de extracción de programas de AF2 discutido en la sección 6.3.

5.2. Extensión con pares e inyecciones. Nuestro objetivo es utilizar el cálculo lambda como un sistema de codificación o representación de pruebas en la lógica, para de esta manera poder definir una semántica formal basada en la interpretación BHK. Para esto necesitamos extender el cálculo lambda puero con términos que permitan representar las pruebas que involucran conjunciones y disyunciones. Esta extensión se define a continuación.

- *λ-términos*: agregamos términos para las operaciones de par ordenado, proyecciones, inyecciones y análisis de casos.
$$e ::= \ldots \mid \langle e, e\rangle \mid \mathsf{fst}\,e \mid \mathsf{snd}\,e \mid \mathsf{inl}\,e \mid \mathsf{inr}\,e \mid \mathsf{case}(r, x.s, y.t)$$
donde en el último caso la notación $x.s$ $(y.t)$ denota un ligado de la respectiva variable en el término que está después del punto.
- *Semántica operacional*: la relación $\mapsto_\beta$ se extiende como sigue:
$$\begin{array}{rcl} \mathsf{fst}\langle r, s\rangle & \mapsto_\beta & r \\ \mathsf{snd}\langle r, s\rangle & \mapsto_\beta & s \\ \mathsf{case}(\mathsf{inl}\,r, x.s, y.t) & \mapsto_\beta & s[x := r] \\ \mathsf{case}(\mathsf{inr}\,r, x.s, y.t) & \mapsto_\beta & t[y := r] \end{array}$$

Las reglas que involucran pares y proyecciones son claras; las reglas para el constructor de casos reflejan el análisis de una inyección de r, de acuerdo a si es izquierda (inl) o derecha (inr) eligiendo el caso correcto s o t mediante la sustitución del parámetro x o y por el valor particular analizado r.

6. La lógica AF2

A grandes rasgos podemos decir que la lógica AF2 es un sistema deductivo para la lógica de segundo orden que cuenta con codificación de pruebas y razonamiento ecuacional. Para dar los detalles de su definición primero tenemos que fijar nuestro sistema de codificación o representación de pruebas para lo cual nos serviremos del cálculo lambda.

6.1. Codificación de pruebas. En la teoría de la demostración, rama de la lógica que se encarga de estudiar la estructura de las pruebas, comparar e identificar pruebas, así como de distinguir una prueba de otra. De esta manera resulta natural elegir una notación conveniente para las pruebas para lo cual usaremos el cálculo λ. A cada prueba de una fórmula A, se le asociará un código único mediante un término del cálculo lambda, esta asociación se denota con $t : A$. Ahora bien, como las pruebas dependen de hipótesis particulares en realidad a partir de la relación binaria $\Gamma \vdash A$ obtenemos una nueva relación ternaria $\Gamma \vdash t : A$ cuyo significado es que la fórmula A es derivable a partir de las hipótesis Γ, siendo el λ-término t un código de prueba para la derivación original $\Gamma \vdash A$. En esta nueva relación los contextos de hipótesis son de la forma $\Gamma = \{x_1 : A_1, \ldots, x_n : A_n\}$, es decir, en este caso no sólo suponemos una fórmula A_i sino que también suponemos que existe una prueba x_i de A_i, la cual por ser desconocida se codifica mediante una variable del cálculo λ. Por ejemplo para la implicación las reglas son:

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (\to I) \qquad \frac{\Gamma \vdash f : A \to B \quad \Gamma \vdash t : A}{\Gamma \vdash f\, t : B}\ (\to E)$$

En este momento es de gran relevancia observar que en el nuevo sistema deductivo figuran tanto los términos de la lógica como los λ-términos. Sin embargo utilizamos la misma notación para los términos de la lógica y para los λ-términos. Por ejemplo en el secuente $\vdash \lambda x.x : P(x) \to P(x)$ las presencias de x antes de los dos puntos no tienen relación alguna con las de la implicación que está después de los dos puntos.

Al utilizar al cálculo λ como un sistema notacional para codificar las pruebas, su semántica operacional juega un papel importante, puesto que permite simplificar pruebas redundantes. Considérese el problema de construir una prueba de $\Gamma \vdash B$, dadas las derivaciones $\Gamma \vdash \lambda x.e : A \to B$ y $\Gamma \vdash r : A$. La solución que salta a la vista es aplicar el modus ponens (regla $(\to E)$) con lo que obtenemos la derivación $\Gamma \vdash (\lambda x.e)r : B$. Sin embargo dado el determinismo de las reglas de inferencia y la prueba dada $\Gamma \vdash \lambda x.e : A \to B$, sabemos que previamente debio obtenerse la prueba $\Gamma, x : A \vdash e : B$ puesto que la regla usada como último paso de la derivación fue $(\to I)$ necesariamente. Ahora bien, obsérvese que como ya tenemos una prueba $\Gamma \vdash r : A$, no es necesario suponer $x : A$ puesto que en la prueba codificada por e podemos sustituir cada prueba x de A por la prueba dada r, obteniendo así una prueba de B sin necesidad de usar el modus ponens, a saber le prueba $\Gamma \vdash e[x := r] : B$. Este proceso es general por lo que siempre podemos reducir la prueba $(\lambda x.e)r$ a la prueba $e[x := r]$ por lo que ambas pruebas pueden

considerarse equivalentes, pero esta equivalencia es parte de la β-equivalencia, es decir $(\lambda x.e)r =_\beta e[x := r]$. Las mismas consideraciones son válidas para la extensión con pares e inyecciones. Con esta idea podemos ya definir el sistema deductivo AF2.

6.2. Definición de la lógica AF2. La sintaxis de la lógica AF2 es la presentada en la sección 2, mientras que su sistema deductivo se obtiene del sistema de deducción natural de la sección 4 mediante la anotación de pruebas con λ-términos.

- Regla de inicio:

$$\frac{}{\Gamma, x : A \vdash x : A}\ (Var)$$

- Reglas para los conectivos:

$$\frac{\Gamma, x : A \vdash r : B}{\Gamma \vdash \lambda x r : A \to B}\ (\to I) \qquad \frac{\Gamma \vdash r : A \to B \quad \Gamma \vdash s : A}{\Gamma \vdash rs : B}\ (\to E)$$

$$\frac{\Gamma \vdash r : A \quad \Gamma \vdash s : B}{\Gamma \vdash \langle r, s\rangle : A \wedge B}\ (\wedge I) \quad \frac{\Gamma \vdash s : A \wedge B}{\Gamma \vdash \mathsf{fst}\, s : A}\ (\wedge E_L) \quad \frac{\Gamma \vdash s : A \wedge B}{\Gamma \vdash \mathsf{snd}\, s : B}\ (\wedge E_R)$$

$$\frac{\Gamma \vdash r : A}{\Gamma \vdash \mathsf{inl}\, r : A \vee B}\ (\vee I_L) \quad \frac{\Gamma \vdash r : B}{\Gamma \vdash \mathsf{inr}\, r : A \vee B}\ (\vee I_R)$$

$$\frac{\Gamma \vdash r : A \vee B \quad \Gamma, x : A \vdash s : C \quad \Gamma, y : B \vdash t : C}{\Gamma \vdash \mathsf{case}(r, x.s, y.t) : C}\ (\vee E)$$

- Reglas para los cuantificadores:

$$\frac{\Gamma \vdash t : A \quad x \notin FV(\Gamma)}{\Gamma \vdash t : \forall x A}\ (\forall I) \qquad \frac{\Gamma \vdash t : \forall x A}{\Gamma \vdash t : A[x := r]}\ (\forall E)$$

$$\frac{\Gamma \vdash t : A \quad X \notin FV(\Gamma)}{\Gamma \vdash t : \forall X A}\ (\forall^2 I) \qquad \frac{\Gamma \vdash t : \forall X A}{\Gamma \vdash t : A[X := \mathcal{P}]}\ (\forall^2 E)$$

  Hasta aquí las reglas de AF2 son exactamente las versiones con pruebas anotadas del sistema de deducción natural para la lógica de segundo orden presentadas en la sección 4, obsérvese que en el caso de los cuantificadores el código t asociado a A es el mismo que para la cuantificación $\forall x A$ o $\forall X A$, esto refleja que las pruebas de cuantificaciones no tienen un contenido computacional. La última regla es característica de AF2 e incorpora el razonamiento ecuacional directamente en la lógica.
- Regla ecuacional:

$$\frac{\Gamma \vdash_{\mathbb{E}} t : A[x := r] \quad \mathbb{E} \rhd r = s}{\Gamma \vdash_{\mathbb{E}} t : A[x := s]}\ (Eq)$$

Veamos ahora algunos ejemplos de derivaciones en la lógica que conllevan la justificación de la definición de algunos de los λ-términos dados en los ejemplos 5.2 y 5.3.

6.3. Ejemplos de extracción de programas. En esta sección ejemplificamos el método de extracción de programas (λ-términos) a partir de pruebas en AF2 desarrollado en [4, 3, 10, 6]. La idea a grandes rasgos es la siguiente: sea $f : A \to B$ una función específicada mediante un conjunto de ecuaciones $\mathbb{E}$ y tal que existan tipos de datos formales (predicados de índice 1 con ciertas propiedades semánticas) $\mathcal{A}$ y $\mathcal{B}$ que representen a los conjuntos A y B. Para hallar un programa (λ-término) t que implemente a f basta con mostrar formalmente que $\vdash_{\mathbb{E}} t : \forall x.\mathcal{A}(x) \to \mathcal{B}(f(x))$.

Veamos algunos ejemplos.

Ejemplo 6.1 (Valores Booleanos)**.** *Sean* t, f *dos constantes representando a los valores booleanos. Definimos el predicado por comprensión correspondiente al tipo booleano como*

$$\mathbb{B}(x) = \{x \mid \forall X.X(\mathsf{t}) \to X(\mathsf{f}) \to X(x)\}$$

Esta definición es similar a la de los números naturales (ejemplo 2.1) y nos viene del principio de inducción para booleanos. Veamos de donde surgen los λ-términos para los valores de verdad del ejemplo 5.2.

1.	$x : X(\mathsf{t}), y : X(\mathsf{f}) \vdash X(\mathsf{t})$	(Var)
2.	$x : X(\mathsf{t}) \vdash \lambda y.x : X(\mathsf{f}) \to X(\mathsf{t})$	$(\to I), 1$
3.	$\vdash \lambda x.\lambda y.x : X(\mathsf{t}) \to X(\mathsf{f}) \to X(\mathsf{t})$	$(\to I), 2$
4.	$\vdash \lambda x.\lambda y.x : \forall X.X(\mathsf{t}) \to X(\mathsf{f}) \to X(\mathsf{t})$	$(\forall^2 I), 3$

Por lo tanto hemos probado $\vdash \mathsf{true} : \mathbb{B}(\mathsf{t})$*. Similarmente se obtiene que* $\vdash \mathsf{false} : \mathbb{B}(\mathsf{f})$.

El siguiente ejemplo hace uso crucial del razonamiento ecuacional.

Ejemplo 6.2 (Conjunción)**.** *Sean* con *un símbolo de función binario y* $\mathbb{E} = \{\mathsf{con}(\mathsf{t}, x) = x,\ \mathsf{con}(\mathsf{f}, x) = \mathsf{f}\}$ *y* $\mathcal{F} =_{def} \{z \mid X(\mathsf{con}(z, y))\}$*. Vamos a mostrar que*

$$\vdash \mathsf{and} : \forall x \forall y.\mathbb{B}(x) \to \mathbb{B}(y) \to \mathbb{B}(\mathsf{con}(x, y))$$

donde and *es el λ-término definido en el ejemplo 5.2. Sea* $\Gamma = \{x : \mathbb{B}(x), y : \mathbb{B}(y),\ z : X(\mathsf{t}), w : X(\mathsf{f})\}$.

1.	$\Gamma \vdash y : \mathbb{B}(y)$	(Var)
2.	$\Gamma \vdash y : X(\mathsf{t}) \to X(\mathsf{f}) \to X(y)$	$(\forall^2 E), 1, [X := X]$
3.	$\Gamma \vdash z : X(\mathsf{t})$	(Var)
4.	$\Gamma \vdash yz : X(\mathsf{f}) \to X(y)$	$(\to E), 2, 3$
5.	$\Gamma \vdash w : X(\mathsf{f})$	(Var)
6.	$\Gamma \vdash yzw : X(y)$	$(\to E), 4, 5$
7.	$\Gamma \vdash_{\mathbb{E}} yzw : X(\mathsf{con}(\mathsf{t}, y))$	$(Eq), 6$
8.	$\Gamma \vdash x : \mathbb{B}(x)$	(Var)
9.	$\Gamma \vdash x : \mathcal{F}(\mathsf{t}) \to \mathcal{F}(\mathsf{f}) \to \mathcal{F}(x)$	$(\forall^2 E), 1, [X := \mathcal{F}]$
10.	$\Gamma \vdash x(yzw) : \mathcal{F}(\mathsf{f}) \to \mathcal{F}(x)$	$(\to E), 9, 7$
11.	$\Gamma \vdash w : X(\mathsf{con}(\mathsf{f}, y))$	$(Eq), 5$
12.	$\Gamma \vdash x(yzw)w : \mathcal{F}(x)$	$(\to E), 10, 11$
13.	$x : \mathbb{B}(x), y : \mathbb{B}(y),\ z : X(\mathsf{t}) \vdash \lambda w.x(yzw)w : X(\mathsf{f}) \to \mathcal{F}(x)$	$(\to I), 12$
14.	$x : \mathbb{B}(x), y : \mathbb{B}(y) \vdash \lambda z.\lambda w.x(yzw)w : X(\mathsf{t}) \to X(\mathsf{f}) \to \mathcal{F}(x)$	$(\to I), 12$
15.	$x : \mathbb{B}(x), y : \mathbb{B}(y) \vdash \lambda z.\lambda w.x(yzw)w : \forall X.X(\mathsf{t}) \to X(\mathsf{f}) \to X(\mathsf{con}(x, y))$	$(\forall E), 14$ *y def. de* $\mathcal{F}$
16.	$x : \mathbb{B}(x) \vdash \lambda y.\lambda z.\lambda w.x(yzw)w : \mathbb{B}(y) \to \mathbb{B}(\mathsf{con}(x, y))$	$(\to I), 15$ *y def. de* $\mathbb{B}$
17.	$\vdash \lambda x.\lambda y.\lambda z.\lambda w.x(yzw)w : \mathbb{B}(x) \to \mathbb{B}(y) \to \mathbb{B}(\mathsf{con}(x, y))$	$(\to I), 16$
18.	$\vdash \lambda x.\lambda y.\lambda z.\lambda w.x(yzw)w : \forall y.\mathbb{B}(x) \to \mathbb{B}(y) \to \mathbb{B}(\mathsf{con}(x, y))$	$(\forall I), 17$
19.	$\vdash \mathsf{and} : \forall x.\forall y.\mathbb{B}(x) \to \mathbb{B}(y) \to \mathbb{B}(\mathsf{con}(x, y))$	$(\forall I), 18$

Pasamos ahora a mostrar las derivaciones de los númerales de Church y de la suma de números naturales dadas en el ejemplo 5.3.

Ejemplo 6.3 (Numerales de Church)**.** *La representación de los números naturales en el cálculo lambda dada en el ejemplo 5.3 se obtiene similarmente al caso de los booleanos, veamos como ejemplo la extracción del numeral* $\bar{2}$*, es decir, mostremos que* $\vdash \bar{2} : \mathbb{N}(s(s(0)))$*, donde la definición de* $\mathbb{N}$ *es la dada en el ejemplo 2.1*

1.	$x : \forall x(X(x) \to X(s(x))),\ y : X(0) \vdash x : \forall x(X(x) \to Xs(x))$	(Var)
2.	$x : \forall x(X(x) \to X(s(x))),\ y : X(0) \vdash x : X(0) \to X(s(0))$	$(\forall E)\ 1,\ [x := 0]$
3.	$x : \forall x(X(x) \to X(s(x))),\ y : X(0) \vdash x : X(s(0)) \to X(s(s(0)))$	$(\forall E)\ 1,\ [x := s(0)]$
4.	$x : \forall x(X(x) \to X(s(x))),\ y : X(0) \vdash y : X(0)$	(Var)
5.	$x : \forall x(X(x) \to X(s(x))),\ y : X(0) \vdash xy : X(s(0))$	$(\to E)\ 2, 4$
6.	$x : \forall x(X(x) \to X(s(x))),\ y : X(0) \vdash x(xy) : X(ss(0))$	$(\to E)\ 3, 5$
7.	$x : \forall x(X(x) \to X(s(x))) \vdash \lambda y.x(xy) : X(0) \to X(s(s(0)))$	$(\to I)\ 6$
8.	$\vdash \lambda x.\lambda y.x(xy) : \forall x(X(x) \to X(s(x)) \to X(0) \to X(s(s(0)))$	$(\to I)\ 7$
9.	$\vdash \bar{2} : \mathbb{N}(s(s(0)))$	$(\forall^2 I)\ 8$

En el siguiente ejemplo el uso de razonamiento ecuacional es indispensable.

Ejemplo 6.4 (Suma de números naturales)**.** *Sean* sum *un símbolo de función binario,* $\mathbb{E} = \{\mathsf{sum}(x, 0) = x,\ \mathsf{sum}(x, s(y)) = s(\mathsf{sum}(x, y))\}$*. Vamos a mostrar que*

$$\vdash \mathsf{suma} : \forall x.\forall y.\mathbb{N}(x) \to \mathbb{N}(y) \to \mathbb{N}(\mathsf{sum}(x, y))$$

donde suma *es el* λ*-término definido en el ejemplo 5.3. Para la prueba necesitaremos del predicado por comprensión* $\mathcal{F} =_{def} \{w \mid \mathbb{N}(\mathsf{sum}(x, w))\}$*.*

Veamos primero que

$$\vdash \lambda u\lambda f\lambda z.f(ufz) : \forall z.\mathcal{F}(z) \to \mathcal{F}(s(z)) \qquad (**)$$

para lo cual tomamos $\Gamma = \{u : \mathcal{F}(z),\ f : \forall x.X(x) \to X(s(x)),\ z : X(0)\}$*.*

1.	$\Gamma \vdash f : \forall x(X(x) \to X(s(x)))$	(Var)
2.	$\Gamma \vdash u : \mathbb{N}(\mathsf{sum}(x, z))$	(Var) *y def. de* $\mathcal{F}$
3.	$\Gamma \vdash z : X(0)$	(Var)
4.	$\Gamma \vdash u : \forall x(X(x) \to X(s(x))) \to X(0) \to X(\mathsf{sum}(x, z))$	$(\forall^2 E), 2, [X := X]$
5.	$\Gamma \vdash uf : X(0) \to X(\mathsf{sum}(x, z))$	$(\to E), 4, 1$
6.	$\Gamma \vdash ufz : X(\mathsf{sum}(x, z))$	$(\to E), 5, 3$
7.	$\Gamma \vdash f : X(\mathsf{sum}(x, z)) \to X(s(\mathsf{sum}(x, z)))$	$(\forall E), 1, [x := \mathsf{sum}(x, z)]$
8.	$\Gamma \vdash f(ufz) : X(s(\mathsf{sum}(x, z)))$	$(\to E), 7, 6$
9.	$\Gamma \vdash_{\mathbb{E}} f(ufz) : X(\mathsf{sum}(x, s(z)))$	$(Eq), 8$
10.	$u : \mathcal{F}(z),\ f : \forall x(X(x) \to X(s(x))) \vdash \lambda z.f(ufz) : X(0) \to X(\mathsf{sum}(x, s(z)))$	$(\to I), 9$
11.	$u : \mathcal{F}(z) \vdash \lambda f.\lambda z.f(ufz) : \forall x(X(x) \to X(s(x))) \to X(0) \to X(\mathsf{sum}(x, s(z)))$	$(\to I), 10$
12.	$u : \mathcal{F}(z) \vdash \lambda f.\lambda z.f(ufz) : \mathbb{N}(\mathsf{sum}(x, s(z)))$	$(\forall^2 I), 11$ *y def. de* $\mathbb{N}$
13.	$\vdash \lambda u.\lambda f.\lambda z.f(ufz) : \mathcal{F}(z) \to \mathcal{F}(s(z))$	$(\to I), 12$ *y def. de* $\mathcal{F}$
14.	$\vdash \lambda u.\lambda f.\lambda z.f(ufz) : \forall z.\mathcal{F}(z) \to \mathcal{F}(s(z))$	$(\forall I), 14$

Para obtener la derivación principal tomamos $\Gamma = \{x : \mathbb{N}(x), y : \mathbb{N}(y)\}$

1.	$\Gamma \vdash_{\mathbb{E}} \lambda u.\lambda f.\lambda z.f(ufz) : \forall z.\mathcal{F}(z) \to \mathcal{F}(s(z))$	(Mon) *en* $(**)$
2.	$\Gamma \vdash y : \forall X.\forall z(X(z) \to X(s(z))) \to X(0) \to X(y)$	(Var)
3.	$\Gamma \vdash y : \forall z(\mathcal{F}(z) \to \mathcal{F}(s(z))) \to \mathcal{F}(0) \to \mathcal{F}(y)$	$(\forall^2 E), 2, [X := \mathcal{F}]$
4.	$\Gamma \vdash y(\lambda u.\lambda f.\lambda z.f(ufz)) : \mathcal{F}(0) \to \mathcal{F}(y)$	$(\to E), 4, 1$
5.	$\Gamma \vdash x : \mathbb{N}(x)$	(Var)
6.	$\Gamma \vdash_{\mathbb{E}} x : \mathbb{N}(\mathsf{sum}(x, 0))$	$(Eq), 5$
7.	$\Gamma \vdash y(\lambda u.\lambda f.\lambda z.f(ufz))x : \mathcal{F}(y)$	$(\to E), 4, 6$ *y def. de* $\mathcal{F}$
8.	$x : \mathbb{N}(x) \vdash \lambda y.y(\lambda u.\lambda f.\lambda z.f(ufz))x : \mathbb{N}(y) \to \mathcal{F}(y)$	$(\to I), 7$
9.	$\vdash \lambda x\lambda y.y(\lambda u.\lambda f.\lambda z.f(ufz))x : \mathbb{N}(x) \to \mathbb{N}(y) \to \mathcal{F}(y)$	$(\to I), 8$
10.	$\vdash \lambda x\lambda y.y(\lambda u.\lambda f.\lambda z.f(ufz))x : \forall y.\mathbb{N}(x) \to \mathbb{N}(y) \to \mathcal{F}(y)$	$(\forall I), 9$
11.	$\vdash \mathsf{suma} : \forall x.\forall y.\mathbb{N}(x) \to \mathbb{N}(y) \to \mathbb{N}(\mathsf{sum}(x, y))$	$(\forall I), 10$ *y def. de* $\mathcal{F}$

Es relevante mencionar que para justificar diversos conceptos involucrados en la demostración de que el método de extracción presentado en los ejemplos anteriores funciona en general requerimos de una semántica como la prometida en este artículo.

Terminamos esta sección mencionando dos propiedades importantes que muestran las repercusiones de la semántica operacional del cálculo lambda en las derivaciones de la lógica.

Proposición 6.1 (Preservación de pruebas)**.** *Si* $\Gamma \vdash t : A$ *y* $t \to^\star_\beta t'$ *entonces* $\Gamma \vdash t' : A$.

Demostración. La prueba de esta importante propiedad no es trivial puesto que las reglas para los cuantificadores y el razonamiento ecuacional no son rastreables mediante los códigos de prueba. Una prueba puede consultarse en [4]. □

Estar propiedad implica que los reductos de un λ-término que codifica a una prueba de una fórmula A siguen siendo códigos para pruebas de A.

Proposición 6.2 (Normalización fuerte)**.** *Si* $\Gamma \vdash t : A$ *entonces* t *es un* λ*-término fuertemente normalizable, es decir, existe un* λ*-término* e *que no es* β*-reducible y tal que* $t \to^\star_\beta e$.

Demostración. La prueba consiste en mostrar que AF2 se encaja en la lógica proposicional de segundo orden (ver [9]) la cual se sabe que es fuertemente normalizable (ver [4]). □

Esta propiedad implica que si un λ-término resulta ser un código de prueba entonces puede simplificarse de manera exhaustiva, lo cual junto con la propiedad de preservación implica que una prueba de una fórmula A siempre puede simplificarse hasta conseguir una prueba llamada normal o canónica que resulta ser la prueba más simple de A.

A continuación definimos la semántica prometida.

7. Semántica constructiva para AF2

En esta sección presentamos la mayor aportación de este artículo, una semántica donde la interpretación $\mathcal{I}(A)$ de una fórmula A será un conjunto de códigos de prueba, es decir de λ-términos, que incluya a las pruebas de A.

7.1. Operaciones sobre conjuntos de λ-términos. Para definir el significado de los conectivos lógicos utilizaremos conjuntos de λ-términos módulo la relación de β-equivalencia. Por ejemplo consideramos iguales a los términos $(\lambda y.xy)w$ y xw. Tomando en cuenta Cesta convención denotamos con Λ al conjunto de todos los lb-términos. En realidad Λ es el conjunto de clases de equivalencia de λ-términos de acuerdo a la relación $=_\beta$, pero esto no se hace explícito.

Definición 7.1. Sea $\mathcal{C} \subseteq \mathcal{P}(\Lambda)$. Definimos las operaciones $\Rightarrow, \oplus, \otimes : \mathcal{P}(\Lambda) \times \mathcal{P}(\Lambda) \to \mathcal{P}(\Lambda)$, como sigue:

$$\begin{array}{rcl} \mathcal{M} \Rightarrow \mathcal{N} & = & \{r \in \Lambda \mid \forall s \in \mathcal{M}.\ rs \in \mathcal{N}\} \\ \mathcal{M} \otimes \mathcal{N} & = & \{r \in \Lambda \mid \mathsf{fst}\, r \in \mathcal{M} \text{ y } \mathsf{snd}\, r \in \mathcal{N}\} \\ \mathcal{M} \oplus \mathcal{N} & = & \{r \in \Lambda \mid \forall \mathcal{P} \in \mathcal{C}\, \forall x\, \forall s \in \mathsf{S}_x(\mathcal{M}, \mathcal{P})\, \forall y\, \forall t \in \mathsf{S}_y(\mathcal{N}, \mathcal{P}).\ \mathsf{case}(r, x.s, y.t) \in \mathcal{P}\} \end{array}$$

donde para cada variable x y $\mathcal{M}, \mathcal{N} \in \mathcal{P}(\Lambda)$,

$$\mathsf{S}_x(\mathcal{M}, \mathcal{N}) = \{t \mid \forall s \in \mathcal{M}.\ t[x := s] \in \mathcal{N}\}.$$

Obsérvese que en la definición anterior el conjunto $\mathcal{C}$ es cualquier conjunto de conjuntos de λ-términos y que sólo se utiliza en la última operación. Veamos ahora algunas propiedades importantes de estas operaciones.

Proposición 7.1 (Correctud de las operaciones)**.** *Sean $\mathcal{M}, \mathcal{N} \in \mathcal{P}(\Lambda)$ y $\mathcal{C} \subseteq \mathcal{P}(\Lambda)$. Se cumple lo siguiente:*

1. *Si $t \in \mathsf{S}_x(\mathcal{M}, \mathcal{N})$ entonces $\lambda xt \in \mathcal{M} \Rightarrow \mathcal{N}$.*
2. *Si $r \in \mathcal{M} \Rightarrow \mathcal{N}$ y $s \in \mathcal{M}$ entonces $rs \in \mathcal{N}$.*
3. *Si $r \in \mathcal{M}$ y $s \in \mathcal{N}$ entonces $\langle r, s\rangle \in \mathcal{M} \otimes \mathcal{N}$.*
4. *Si $r \in \mathcal{M} \otimes \mathcal{N}$ entonces $\mathsf{fst}\, r \in \mathcal{M}$ y $\mathsf{snd}\, r \in \mathcal{N}$.*
5. *Si $r \in \mathcal{M}$ entonces $\mathsf{inl}\, r \in \mathcal{M} \oplus \mathcal{N}$.*
6. *Si $r \in \mathcal{M}$ entonces $\mathsf{inr}\, r \in \mathcal{M} \oplus \mathcal{N}$.*
7. *Si $r \in \mathcal{M} \oplus \mathcal{N}, \mathcal{P} \in \mathcal{C}, s \in \mathsf{S}_x(\mathcal{M}, \mathcal{P})$ y $t \in \mathsf{S}_y(\mathcal{N}, \mathcal{P})$ entonces $\mathsf{case}(r, x.s, y.t) \in \mathcal{P}$*

Demostración. Mostramos los casos para la implicación, para los casos restantes el razonamiento es similar:

1. Sean $t \in \mathsf{S}_x(\mathcal{M}, \mathcal{N})$ y $s \in \mathcal{M}$. Para mostrar que $(\lambda xt)s \in \mathcal{N}$ basta ver que $t[x := s] \in \mathcal{N}$, pero esto es inmediato de la definición de $\mathsf{S}_x(\mathcal{M}, \mathcal{N})$.
2. Sean $r \in \mathcal{M} \Rightarrow \mathcal{N}$ y $s \in \mathcal{M}$. De la definición de $\mathcal{M} \Rightarrow \mathcal{N}$ es claro que $rs \in \mathcal{N}$.

□

La proposición anterior nos permitirá demostrar de manera sencilla el teorema de correctud para AF2 siempre y cuando el conjunto $\mathcal{C}$ de conjuntos de λ-términos sea un conjunto coherente de acuerdo a la siguiente

Definición 7.2. Un conjunto $\mathcal{C} \subseteq \mathcal{P}(\Lambda)$ es coherente si y sólo si cumple las siguientes condiciones:

- $\varnothing, \Lambda \in \mathcal{C}$.
- $\mathcal{C}$ es cerrado bajo las siguientes operaciones:

$$\bigcap,\ \Rightarrow,\ \oplus,\ \otimes$$

Dado un conjunto coherente $\mathcal{C}$ y un conjunto cualquiera $M \neq \varnothing$, el conjunto de todas las funciones $f : M^n \to \mathcal{C}$ es de gran importancia para definir la interpretación de los predicados y se denotará con $\mathcal{C}_n$.

Ejemplo 7.1. *$\mathcal{P} = \mathcal{P}(\Lambda)$ es claramente un conjunto coherente. Más aún, es fácil cerciorarse de que el conjunto $\mathcal{B} = \{\varnothing, \Lambda\}$ también es coherente.*

Dado un conjunto coherente $\mathcal{C}$ y un conjunto cualquiera L de λ-términos, siempre existe un elemento de $\mathcal{C}$ que contiene a L y es mínimo, de acuerdo a la siguiente

Definición 7.3. Dado un conjunto $L \subseteq \Lambda$, definimos la cerradura coherente de L como

$$\mathsf{cl}(L) := \bigcap \{\mathcal{N} \in \mathcal{C} \mid L \subseteq \mathcal{N}\}$$

Es claro que $\mathsf{cl}(M)$ está bien definido, puesto que $\Lambda \in \mathcal{C}$, y que es el mínimo elemento de $\mathcal{C}$ que contiene a M. Este concepto será de gran importancia en la sección 8.

Ya estamos en posibilidad de definir los conceptos de modelo e interpretación.

7.2. $\mathcal{C}$-modelos y funciones de interpretación. El concepto de modelo con el que trabajaremos depende de un conjunto coherente $\mathcal{C}$ fijo previamente.

Definición 7.4. Dado un conjunto coherente $\mathcal{C}$, un $\mathcal{C}$-modelo para un lenguaje de segundo orden $\mathfrak{L}$ es un par $\mathcal{M} = \langle M, \mathcal{I}\rangle$ donde M es un conjunto no vacío e $\mathcal{I}$ es una función de interpretación para $\mathfrak{L}$ tal que:

- Para cada símbolo de función de índice n, $f \in \mathfrak{L}$:
$$\mathcal{I}(f) : M^n \to M,$$
- Para cada símbolo de predicado de índice n, $P \in \mathfrak{L}$:
$$\mathcal{I}(P) : M^n \to \mathcal{C}$$

Obsérvese que los símbolos de predicado se interpretan como funciones valuadas en $\mathcal{C}$, es decir como elementos de $\mathcal{C}_n$.

Como es usual la interpretación de términos y fórmulas depende del concepto de estado dado en la siguiente

Definición 7.5. Un estado o asignación a las variables es una función $\sigma : Var \to \Lambda \cup \mathcal{C}_n$ tal que $\sigma(x) \in M$ y $\sigma(X^{(n)}) \in \mathcal{C}_n$. Dados $r \in M$ o $\Phi \in \mathcal{C}_n$, los estados modificados $\sigma[x/r]$ and $\sigma[X/\Phi]$ se definen de la manera usual.

A continuación damos la definición de interpretación para términos, predicados y fórmulas, las cuales dependen de un $\mathcal{C}$-modelo $\mathcal{M} = \langle M, \mathcal{I}\rangle$ y de un estado σ fijos.

Definición 7.6. Dado un estado σ definimos la función de interpretación de términos

$$\mathcal{I}_\sigma : \mathsf{Term} \to M,$$

como sigue:

- $\mathcal{I}_\sigma(x) = \sigma(x)$
- $\mathcal{I}_\sigma(f(t_1, \ldots, t_n)) = \mathcal{I}(f)(\mathcal{I}_\sigma(t_1), \ldots, \mathcal{I}_\sigma(t_n))$

Definición 7.7. Dados un estado σ definimos la función de interpretación de predicados

$$\mathcal{I}_\sigma : \mathsf{Pred} \to \mathcal{C}_n,$$

como sigue:

- Variables de predicado: $\mathcal{I}_\sigma(X) = \sigma(X)$
- Símbolos de predicado: $\mathcal{I}_\sigma(P) = \mathcal{I}(P)$
- Predicados por comprensión: $\mathcal{I}_\sigma(\mathcal{F}) = \Phi_\mathcal{F}$
  donde si $\mathcal{F} =_{def} \{\vec{x}\,|A\}$ entonces definimos $\Phi_\mathcal{F} : M^n \to \mathcal{C}$ como
  $$\Phi_\mathcal{F}(\vec{m}\,) = \mathcal{I}_{\sigma[\vec{x}/\vec{m}\,]}(A),$$
  para toda $\vec{m} \in M^n$.

Definición 7.8 (Interpretación de fórmulas)**.** Dado un estado σ definimos la función de interpretación de fórmulas

$$\mathcal{I}_\sigma : \mathsf{Form} \to \mathcal{C},$$

como sigue:

$$\begin{array}{rcl}
\mathcal{I}_\sigma(P(t_1,\ldots,t_n)) & = & \mathcal{I}_\sigma(P)(\mathcal{I}_\sigma(t_1),\ldots,\mathcal{I}_\sigma(t_n)) \\
\mathcal{I}_\sigma(A \to B) & = & \mathcal{I}_\sigma(A) \Rightarrow \mathcal{I}_\sigma(B) \\
\mathcal{I}_\sigma(A \wedge B) & = & \mathcal{I}_\sigma(A) \otimes \mathcal{I}_\sigma(B) \\
\mathcal{I}_\sigma(A \vee B) & = & \mathcal{I}_\sigma(A) \oplus \mathcal{I}_\sigma(B) \\
\mathcal{I}_\sigma(\forall x A) & = & \bigcap\{\mathcal{I}_{\sigma[x/m]}(A) \mid m \in M\,\} \\
\mathcal{I}_\sigma(\forall X A) & = & \bigcap\{\mathcal{I}_{\sigma[X/\Phi]}(A) \mid \Phi \in \mathcal{C}_n\,\}
\end{array}$$

Los siguientes lemas resaltan propiedades de importancia y deben cumplirse en cualquier semántica correcta con respecto a un sistema deductivo.

Lema 7.1 (Coincidencia)**.** *Sean t un término, A una fórmula, y σ, σ' dos estados que coinciden en los conjuntos $Var(t)$ y $FV(A)$, de variables de t y de variables libres de A respectivamente. Entonces*

- $\mathcal{I}_\sigma(t) = \mathcal{I}_{\sigma'}(t)$.
- $\mathcal{I}_\sigma(A) = \mathcal{I}_{\sigma'}(A)$.

Demostración. Inducción sobre t y A respectivamente. □

Lema 7.2 (Sustitución)**.** *Sean t un término, A una fórmula, r un término y $\mathcal{P}$ un predicado. Entonces*

- $\mathcal{I}_\sigma(t[x := r]) = \mathcal{I}_{\sigma[x/\mathcal{I}_\sigma(r)]}(t)$
- $\mathcal{I}_\sigma(A[x := r]) = \mathcal{I}_{\sigma[x/\mathcal{I}_\sigma(r)]}(A)$
- $\mathcal{I}_\sigma(A[X := \mathcal{P}]) = \mathcal{I}_{\sigma[X/\mathcal{I}_\sigma(\mathcal{P})]}(A)$

Demostración. Inducción sobre t y A respectivamente. □

El objetivo anunciado acerca de nuestra semántica se probará en la siguiente sección

7.3. El teorema de adecuación o correctud. El teorema de adecuación o correctud garantizará que el significado de una fórmula A incluye a las pruebas de A. Dado que AF2 incluye razonamiento ecuacional debemos restringir la clase de modelos a aquellos que satisfacen las ecuaciones involucradas en una prueba particular de acuerdo a la siguiente

Definición 7.9. Sean $\mathcal{M} = \langle M, \mathcal{I}\rangle$ un $\mathcal{C}$-modelo y σ un estado de las variables. Decimos que la interpretación $\mathcal{I}_\sigma$ satisface a la ecuación $r = s$ si y sólo si $\mathcal{I}_\sigma(r) = \mathcal{I}_\sigma(s)$. Más aún, si $\mathbb{E}$ es un conjunto de ecuaciones, decimos que $\mathcal{I}_\sigma$ satisface a $\mathbb{E}$ si y sólo si $\mathcal{I}_\sigma$ satisface a cada caso particular de una ecuación de $\mathbb{E}$.

Proposición 7.2. *Si $\mathcal{I}_\sigma$ satisface a $\mathbb{E}$ y $\mathbb{E} \rhd r = s$ entonces $\mathcal{I}_\sigma$ satisface a $r = s$.*

Demostración. Inducción sobre $\mathbb{E} \rhd r = s$. □

Es tiempo de probar nuestro teorema principal.

Teorema 7.1 (Adecuación o correctud)**.** *Sea $\mathcal{M} = \langle M, \mathcal{I}\rangle$ un $\mathcal{C}$-modelo tal que la interpretación $\mathcal{I}_\sigma$ satisface al conjunto de ecuaciones $\mathbb{E}$. Si $\Gamma \vdash_{\mathbb{E}} t : A$, con $\Gamma = \{x_1 : A_1, \ldots, x_n : A_n\}$ y para toda $1 \leq i \leq n$, $r_i \in \mathcal{I}_\sigma(A_i)$, entonces $t[\vec{x} := \vec{r}] \in \mathcal{I}_\sigma(A)$.*

Demostración. Sea $\vec{r} = r_1, \ldots, r_n$ como requiere el teorema. Hacemos inducción sobre $\Gamma \vdash_{\mathbb{E}} t : A$. El caso base correspondiente a la regla (Var) es claro. Veamos los casos para la implicación, los cuantificadores y el razonamiento ecuacional.

- Regla $(\to I)$. Sea $\Gamma \vdash \lambda xt : A \to B$. Como $\Gamma, x : A \vdash t : B$, entonces por H.I. tenemos que para toda $s \in \mathcal{I}_\sigma(A)$, se tiene que $t[\vec{x}, x := \vec{r}, s] \in \mathcal{I}_\sigma(B)$. Queremos mostrar que $(\lambda xt)[\vec{x} := \vec{r}] \in \mathcal{I}_\sigma(A \to B)$, es decir, que $(\lambda x.t[\vec{x} := \vec{r}]) \in \mathcal{I}_\sigma(A) \Rightarrow \mathcal{I}_\sigma(B)$. De acuerdo a la parte 1 del lema 7.1 basta ver que $t[\vec{x} := \vec{r}] \in \mathsf{S}_x(\mathcal{I}_\sigma(A), \mathcal{I}_\sigma(B))$. Sea $s \in \mathcal{I}_\sigma(A)$, queremos ver que $t[\vec{x} := \vec{r}][x := s] \in \mathcal{I}_\sigma(B)$. Pero por propiedades de sustitución sabemos que $t[\vec{x} := \vec{r}][x := s] = t[\vec{x}, x := \vec{r}, s]$ por lo que lo requerido es inmediato de la hipótesis de inducción.
- Regla $(\to E)$. Sea $\Gamma \vdash ts : B$ la conclusión de $\Gamma \vdash t : A \to B$ y $\Gamma \vdash s : A$. Por H.I. tenemos que $t[\vec{x} := \vec{r}] \in \mathcal{I}_\sigma(A \Rightarrow B)$ y que $s[\vec{x} := \vec{r}] \in \mathcal{I}_\sigma(A)$, de donde, por la definición de $\mathcal{I}_\sigma(A \Rightarrow B)$ y la segunda parte del lema 7.1, concluimos que $ts[\vec{x} := \vec{r}] = t[\vec{x} := \vec{r}]s[\vec{x} := \vec{r}] \in \mathcal{I}_\sigma(B)$.
- Regla $(\forall I)$. Sea $\Gamma \vdash t : \forall xA$ obtenida de $\Gamma \vdash t : A$, donde $x \notin FV(\Gamma)$. Como $x \notin FV(\Gamma)$ entonces, por el lema de coincidencia $\mathcal{I}_\sigma(A_i) = \mathcal{I}_{\sigma[x/m]}(A_i)$, para cualquier $m \in M$, de donde $r_i \in \mathcal{I}_{\sigma[x/m]}(A_i)$ para toda $1 \leq i \leq n$ y, por H.I., concluimos $t[\vec{x} := \vec{r}] \in \mathcal{I}_{\sigma[x/m]}(A)$, para toda $m \in M$, lo cual implica que $t[\vec{x} := \vec{r}] \in \mathcal{I}_\sigma(\forall xA)$.
- Regla $(\forall E)$. Sea $\Gamma \vdash t : A[x := s]$ obtenida de $\Gamma \vdash t : \forall xA$. Por H.I. sabemos que $t[\vec{x} := \vec{r}] \in \mathcal{I}_\sigma(\forall xA)$, lo cual implica en particular que $t[\vec{x} := \vec{r}] \in \mathcal{I}_{\sigma[x/\mathcal{I}_\sigma(s)]}(A)$, puesto que $\mathcal{I}_\sigma(s) \in M$. Finalmente el lema de coincidencia nos lleva a $t[\vec{x} := \vec{r}] \in \mathcal{I}_\sigma(A[x := s])$
- Regla $(\forall^2 I)$. Sea $\Gamma \vdash t : \forall XA$ obtenida de $\Gamma \vdash t : A$, donde $X \notin FV(\Gamma)$. Como $X \notin FV(\Gamma)$ entonces, por el lema de coincidencia $\mathcal{I}_\sigma(A_i) = \mathcal{I}_{\sigma[X/\Phi]}(A_i)$, para cualquier $\Phi \in \mathcal{C}_n$, de donde $r_i \in \mathcal{I}_{\sigma[X/\Phi]}(A_i)$ para toda $1 \leq i \leq n$ y, por H.I., concluimos $t[\vec{x} := \vec{r}] \in \mathcal{I}_{\sigma[X/\Phi]}(A)$, para toda $\Phi \in \mathcal{C}_n$, lo cual implica que $t[\vec{x} := \vec{r}] \in \mathcal{I}_\sigma(\forall XA)$.
- Regla $(\forall^2 E)$. Sea $\Gamma \vdash t : A[X := \mathcal{P}]$ obtenida de $\Gamma \vdash t : \forall XA$. Por H.I. sabemos que $t[\vec{x} := \vec{r}] \in \mathcal{I}_\sigma(\forall XA)$, lo cual implica en particular que $t[\vec{x} := \vec{r}] \in \mathcal{I}_{\sigma[X/\mathcal{I}_\sigma(\mathcal{P})]}(A)$, puesto que $\mathcal{I}_\sigma(\mathcal{P}) \in \mathcal{C}_n$. Finalmente el lema de coincidencia nos lleva a $t[\vec{x} := \vec{r}] \in \mathcal{I}_\sigma(A[X := \mathcal{P}])$.
- Regla (Eq). Sea $\Gamma \vdash t : A[x := r]$ obtenida de $\Gamma \vdash t : A[x := s]$ y $\mathbb{E} \rhd r = s$. Por H.I. tenemos que $t[\vec{x} := \vec{r}] \in \mathcal{I}_\sigma(A[x := r])$. Como $\mathcal{I}_\sigma$ satisface a $\mathbb{E}$ entonces $\mathcal{I}_\sigma(r) = \mathcal{I}_\sigma(s)$ por lo que $\mathcal{I}_{\sigma[x/\mathcal{I}_\sigma(r)]}(A) = \mathcal{I}_{\sigma[x/\mathcal{I}_\sigma(s)]}(A)$, de donde

por el lema de sustitución $\mathcal{I}_\sigma(A[x := r]) = \mathcal{I}_\sigma(A[x := s])$ y por lo tanto $t[\vec{x} := \vec{r}] \in \mathcal{I}_\sigma(A[x := s])$

□

Corolario 7.1. *Sea $\mathcal{M} = \langle M, \mathcal{I}\rangle$ un $\mathcal{C}$-modelo tal que la interpretación $\mathcal{I}_\sigma$ satisface al conjunto de ecuaciones $\mathbb{E}$. Si $\vdash_{\mathbb{E}} t : A$ entonces $t \in \mathcal{I}_\sigma(A)$*

Por lo tanto si A es un teorema entonces cualquier prueba t de A es parte del significado $\mathcal{I}_\sigma(A)$ de A. Esto sucede siempre y cuando garanticemos la existencia de un $\mathcal{C}$-modelo para el caso en que la prueba de A no requiera razonamiento ecuacional, como es el caso de todos los teoremas de la lógica de segundo orden sin igualdad; en otro caso se requiere la existencia de un $\mathcal{C}$-modelo que satisfaga al conjunto de ecuaciones $\mathbb{E}$ involucrado en la prueba de A. En las siguientes secciones argumentamos acerca de la existencia de tales modelos.

8. Modelos sintácticos para la igualdad

En este apartado construimos para cada fórmula A, cuya prueba involucra a un conjunto de ecuaciones $\mathbb{E}$, un $\mathcal{C}$-modelo sintáctico y una interpretación que lo satisfaga, es decir, un modelo cuyo universo es esencialmente un conjunto de términos del lenguaje de segundo orden en cuestión.

Definición 8.1. Para cada juicio $\mathfrak{J} =_{def} \Gamma \vdash_{\mathbb{E}} t : A$ se define un $\mathcal{C}$-modelo $\mathcal{M}_{\mathfrak{J}} = \langle M, \mathcal{I}\rangle$ como sigue:

- Sea $\approx_{\mathbb{E}}$ la siguiente relación binaria sobre términos del lenguaje

$$r \approx_{\mathbb{E}} s \Leftrightarrow_{def} \mathbb{E} \rhd r = s.$$

  De la definición 4.5 es claro que esta es una relación de equivalencia.
- El universo de $\mathcal{M}_{\mathfrak{J}}$ es el conjunto $M = \mathsf{Term}_{\mathcal{L}}/\approx_{\mathbb{E}}$, de clases de equivalencia $[t]$ de la relación $\approx_{\mathbb{E}}$.
- La función de interpretación $\mathcal{I}$ se define como sigue:
  - $f^{\mathcal{I}} : M^n \to M,\ f^{\mathcal{I}}([t_1], \ldots, [t_n]) =_{def} [f(t_1, \ldots, t_n)]$
  - $P^{\mathcal{I}} : M^n \to \mathcal{C},\ P^{\mathcal{I}}([t_1], \ldots, [t_n]) =_{def} \mathsf{cl}(\{s \in \Lambda \mid \Gamma \vdash_{\mathbb{E}} s : P(t_1, \ldots, t_n)\})$

Es facil ver que la función de interpretación está bien definida y por lo tanto $\mathcal{M}_{\mathfrak{J}}$ es un $\mathcal{C}$-modelo. Es importante observar que la operación cerradura coherente (def. 7.3) es indispensable para garantizar que $P^{\mathcal{I}} \in \mathcal{C}_n$.

El siguiente lema nos muestra que en $\mathcal{M}_{\mathfrak{J}}$, la interpretación de términos se realiza mediante una sustitución.

Lema 8.1. *Sean σ un estado y $r \in \mathsf{Term}_{\mathfrak{L}}$ tales que $Var(r) = \vec{x}$. Si $\sigma(x_i) = [s_i]$ entonces $\mathcal{I}_\sigma(r) = \big[r[\vec{x} := \vec{s}]\big]$.*

Demostración. Inducción sobre r. □

Ahora podemos definir la interpretación que satisface al conjunto $\mathbb{E}$.

Proposición 8.1. *Para cada juicio $\mathfrak{J} =_{def} \Gamma \vdash_{\mathbb{E}} t : A$ existe un estado σ en el $\mathcal{C}$-modelo $\mathcal{M}_{\mathfrak{J}}$ tal que la interpretación $\mathcal{I}_\sigma$ satisface al conjunto de ecuaciones $\mathbb{E}$.*

Demostración. Definimos el estado σ de $\mathcal{M}_{\mathfrak{J}}$, como sigue:

- Para cualquier variable de primer orden x, $\sigma(x) = [x]$.

- Para cualquier variable de segundo orden X, $\sigma(X) = \Phi$, donde

$$\Phi([t_1], \ldots, [t_n]) =_{def} \mathsf{cl}(\{s \in \Lambda \mid \Gamma \vdash_{\mathbb{E}} s : X(t_1, \ldots, t_n)\}).$$

Obsérvese que la operación de cerradura coherente (def. 7.3) es indispensable para garantizar que $\Phi \in \mathcal{C}_n$. Es fácil ver que σ es una función bien definida y por lo tanto es un estado. Más aún, $\mathcal{I}_\sigma$ satisface a $\mathbb{E}$, puesto que si $r = s$ es un caso particular de una ecuación de $\mathbb{E}$ entonces $\mathbb{E} \rhd r = s$ y por lo tanto $r \approx_{\mathbb{E}} s$ lo cual implica que $[r] = [s]$. Pero si $Var(r) = \vec{x}$ y $Var(s) = \vec{y}$, entonces por definición de σ y por el lema 8.1 tenemos

$$\mathcal{I}_\sigma(r) = \left[r[\vec{x} := \vec{x}]\right] = [r] = [s] = \left[s[\vec{y} := \vec{y}]\right] = \mathcal{I}_\sigma(s)$$

□

En la siguiente sección mostramos que los modelos clásicos de la lógica de segundo orden corresponden biúnivocamente a cierta clase de $\mathcal{C}$-modelos.

9. Relación con la semántica clásica

Resulta de interés observar que para el conjunto coherente $\mathcal{B} = \{\varnothing, \Lambda\}$ la interpretación de una fórmula A en un $\mathcal{B}$-modelo es uno de dos elementos por lo que A puede considerarse verdadera si $\mathcal{I}_\sigma(A) = \Lambda$ y falsa cuando $\mathcal{I}_\sigma(A) = \varnothing$. De esta observación se intuye que hay una relación entre los $\mathcal{B}$-modelos y los modelos clásicos, la cual mostraremos formalmente en esta sección.

Recordemos la definición de modelo clásico para la lógica de segundo orden.

Definición 9.1. Un modelo clásico para un lenguaje de segundo orden $\mathfrak{L}$ es una terna $\mathfrak{M} = \langle M, \mathcal{R}, \mathcal{J}\rangle$ donde M es un conjunto no vacío, $\mathcal{R} = \{\mathcal{R}_n\}_{n\in\mathbb{N}}$ es una familia de familias de relaciones de índice n sobre M, es decir, se cumple que para toda $n \in \mathbb{N}$, $\mathcal{R}_n \subseteq \mathcal{P}(M^n)$, y $\mathcal{J}$ es una función de interpretación para $\mathfrak{L}$ tal que:

- Para cada símbolo de función de índice n, $f \in \mathfrak{L}$:

$$\mathcal{J}(f) : M^n \to M,$$

- Para cada símbolo de predicado de índice n, $P \in \mathfrak{L}$:

$$\mathcal{J}(P) \in \mathcal{R}_n$$

En el caso en que $\mathcal{R}_n = \mathcal{P}(M^n)$ para toda $n \in \mathbb{N}$, diremos que el modelo $\mathfrak{M}$ es un modelo pleno.

Se observa que la única diferencia en la definición de interpretación para $\mathcal{C}$-modelos y modelos clásicos está en el caso de los predicados. Las nociones de estado e interpretación, así como la relación de satisfacción ($\models$) en modelos clásicos son las usuales y pueden consultarse por ejemplo en [2].

Veamos ahora como definir un modelo y estado clásicos asociados a un $\mathcal{B}$-modelo y estado dados.

Definición 9.2. Para cada $\mathcal{B}$-modelo $\mathcal{M} = \langle M, \mathcal{I}\rangle$, definimos un modelo clásico $\mathfrak{M}^c = \langle \Lambda, \mathcal{R}, \mathcal{J}\rangle$ asociado a $\mathcal{M}$ como sigue:

- El universo de $\mathfrak{M}$ es el conjunto Λ de λ-términos
- Para cada $n \in \mathbb{N}$, $\mathcal{R}_n = \mathcal{P}(\Lambda^n)$, es decir, el modelo es pleno.

- La función de interpretación $\mathcal{J}$ está dada por:
  - Para cada símbolo de función $f \in \mathfrak{L}$ de índice n, $\mathcal{J}(f) = \mathcal{I}(f)$.
  - Para cada símbolo de función $P \in \mathfrak{L}$ de índice n,

$$\mathcal{J}(P) = \{(t_1, \ldots, t_n) \in \Lambda^n \mid \mathcal{I}(P)(t_1, \ldots, t_n) = \Lambda\}.$$

Definición 9.3. Sean $\mathcal{M} = \langle M, \mathcal{I}\rangle$ un $\mathcal{B}$–modelo y σ un estado en $\mathcal{M}$. Definimos el estado clásico asociado a σ, denotado σ^c, como sigue:

- $\sigma^c(x) = \sigma(x)$
- $\sigma^c(X^{(n)}) = \{(t_1, \ldots, t_n) \in \Lambda^n \mid \sigma(X)(t_1, \ldots, t_n) = \Lambda\}$

Ahora ya podemos mostrar formalmente que todo $\mathcal{B}$-modelo es equivalente con respecto a la noción de satisfacción con su modelo clásico asociado.

Definición 9.4. Si $\mathcal{M} = \langle M, \mathcal{I}\rangle$ es un $\mathcal{B}$-modelo y σ es un estado diremos que A se satisface en $\mathcal{M}$ en el estado σ, denotado $\mathcal{M} \Vdash_\sigma A$, si y sólo si $\mathcal{I}_\sigma(A) = \Lambda$.

Proposición 9.1. *Sea $\mathcal{M} = \langle M, \mathcal{I}\rangle$ un $\mathcal{B}$-modelo. Para cualquier fórmula A y estado σ se cumple que:*

$$\mathcal{M} \Vdash_\sigma A \text{ si y sólo si } \mathfrak{M}^c \models_{\sigma^c} A$$

Demostración. Inducción sobre A. □

De manera que hemos probado formalmente que los $\mathcal{B}$-modelos son esencialmente los modelos clásicos plenos cuyo universo es el conjunto Λ de λ-términos β-equivalentes.

10. Comentarios Finales

En este artículo hemos presentado a la lógica AF2, un sistema deductivo para la lógica de segundo orden cuyas principales características son que permite el razonamiento ecuacional mediante la igualdad de Leibniz y que cuenta con un mecanismo de codificación de pruebas que permite la extracción de programas (λ-términos) a partir de derivaciones de ciertas especificaciones de funciones definidas mediante un conjunto de ecuaciones, usualmente recursivas. Este conjunto es en esencia un programa funcional por lo que la lógica AF2 resulta más que adecuada como un prototipo de lenguaje de programación funcional que adicionalmente cuenta con el rigor dado por la estructura lógica. En particular cualquier programa extraido de una derivación en la lógica estará libre de ciclos divergentes gracias a la propiedad de normalización fuerte. Ademas de dar a conocer las capacidades de esta lógica así como una muy breve introducción al cálculo lambda, nuestro principal propósito fue desarrollar una semántica constructiva para la misma, a la luz de la semántica intensional de la lógica constructiva conocida como interpretación de Brouwer-Heyting-Kolmogorov (BHK). En esta semántica la interpretación de una fórmula A es un conjunto de λ-términos que captura a las pruebas de A en el sentido de que incluye a aquellos λ-términos que son códigos de prueba de A. Esto queda garantizado por el teorema de correctud discutido aquí, por lo que nuestra semántica es constructiva en el sentido dado por la interpretación BHK. Especial énfasis hemos puesto tanto en justificar detalladamente el mecanismo de razonamiento ecuacional con la igualdad de Leibniz, como en mostrar mediante ejemplos el mecanismo de extracción de programas, cuya justificación técnica si bien no presentada aquí (ver [4, 3, 10, 6]) se basa en una noción de tipo de datos formal que involucra de manera imprescindible a nuestra semántica. Con respecto a la igualdad

es importante remarcar que si bien hemos mostrado que el razonamiento ecuacional es válido, esto no implica necesariamente que pudimos haber utilizado la igualdad como una operación primitiva, pues en tal caso tendríamos que haber definido el significado de la fórmula atómica $t = s$ como un conjunto de λ-términos que incluya a sus codigos de prueba, pero no es claro como codificar en el cálculo lambda a está clase de pruebas puramente ecuacionales. La parte final del artículo se encargo de mostrar la existencia de los modelos requeridos por el teorema de correctud, así como de puntualizar la relación existente entre ciertos modelos constructivos y los modelos clásicos.

Para finalizar queremos mencionar que frecuentemente el sistema AF2 es considerado en la literatura como un sistema tipado de reescritura de términos cuyos tipos son las fórmulas de la lógica de segundo orden, en este sentido AF2 no es más que un caso particular del cálculo lambda tipado. Esto no es sino un ejemplo de la llamada correspondencia de Curry-Howard (véase [13]). En este artículo hemos querido presentar a AF2 desde el punto de vista puramente lógico y no desde la perspectiva de los sistemas de tipos para lenguajes de programación, por ser este enfoque más accesible a los interesados en las aplicaciones computacionales de la lógica, pero que estén familiarizados principalmente con la lógica matemática en su sentido más puro.

Referencias

[1] Ulrich Berger. Program Extraction from Normalization Proofs. Lecture Notes in Computer Science, 664, p91-106. Springer 1993.

[2] Herbert B. Enderton. "Una introducción matemática a la lógica". *IIF-UNAM.*, 2004.

[3] J.L. Krivine, M. Parigot. Programming with Proofs. In *Journal of Information Processing and Cybernetics EIK (Formerly Elektronische Informationsverarbeitung und Kybernetik)* 26(3) pp. 149-167. 1990.

[4] J.L. Krivine. Lambda-Calculus, Types and Models. Ellis Horwood Series in Computers and their Applications. Ellis Horwood, Masson 1993.

[5] D. Leivant. Reasoning about Functional Programs and Complexity Classes associated with Type Disciplines. *Proceedings of 24th Annual Symposium on Foundations of Computer Science* pp.460-469 IEEE Computer Science Press. 1983.

[6] Favio E. Miranda Perea. On Extensions of AF2 with Monotone and Clausular (Co)inductive Definitions. Dissertation. *Ludwig-Maximilians-Universität München.*, 2004.

[7] Favio E. Miranda Perea. Realizability for Monotone and Clausular (Co)inductive Definitions. Electronic Notes in Theoretical Computer Science Vol 123. Elsevier Science Holland 2005

[8] Favio E. Miranda Perea. *Two Extensions of System* F *with (Co)iteration and Primitive (Co)recursion Principles.* Por aparecer en Theoretical Informatics and Applications. 2009.

[9] Favio E. Miranda Perea. *La lógica proposicional de segundo orden.* Memorias de la Sociedad Matemática Mexicana. Vol. 40, Aportaciones Matemáticas 2009.

[10] Michel Parigot. *Recursive Programming with Proofs.* Theoretical Computer Science **No.** 94 (1992). pp. 335-356.

[11] C. Raffalli. Data Types, Infinity and Equality in System AF2. Lecture Notes in Computer Science, 832, p280-294. Springer. 1993.

[12] Helmut Schwichtenberg, Anne S. Troelstra. "Basic Proof Theory". *Cambridge University Press.* Cambridge Tracts in Theoretical Computer Science **No.** 43., 1996.

[13] Morten H. Sørensen, Pawel Urzyczyn. "Lectures on the Curry-Howard Isomorphism". *Elsevier.* Studies in Logic and the Foundations of Mathematics. **Vol.** 149., 2006.

[14] Anne S. Troelstra, Dirk van Dalen. "Constructivism in Mathematics, An Introduction, vol. 1". *North-Holland.* Studies in Logic and the Foundations of Mathematics. **Vol.** 121., 1988.

Departamento de Matemáticas, Facultad de Ciencias UNAM, Circuito Exterior S/N, Cd. Universitaria 04510, México D.F., México
E-mail address: favio@ciencias.unam.mx

Current address, (L. González Huesca): Laboratoire Preuves, Programmes et Systèmes, Equipe πr^2. INRIA 23 avenue d'Italie, CS 81321 - 75214 Paris Cedex 13, Francia
E-mail address: luglzhuesca@ciencias.unam.mx

Academia de Informática, Colegio de Ciencia y Tecnología, Universidad Autónoma de la Ciudad de México, Plantel San Lorenzo Tezonco. Prolongación San Isidro 151. San Lorenzo Tezonco, Iztapalapa, 09790 México D.F., México
E-mail address: liliana.mar@gmail.com