UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Agregando polimorfismo a una lógica que identifica proposiciones isomorfas

AUTORES: Cristian Fabián Sottile

DIRECTOR: Alejandro Díaz-Caro

CODIRECTOR: Claudia Pons

ASESOR PROFESIONAL:

CARRERA: Licenciatura en Informática

Resumen

Tanto los sistemas de tipos como los sistemas de pruebas distinguen elementos que tienen diferente forma aunque tengan el mismo significado, como pueden ser las pruebas de las conjunciones $A \wedge B$ y $B \wedge A$, por lo cual una prueba de una no constituye una prueba de la otra, a pesar de que se puede demostrar mediante la existencia de un isomorfismo que dichas proposiciones son equivalentes. Sistema I es un cálculo lambda simplemente tipado con pares, extendido con una teoría ecuacional obtenida a partir de los isomorfismos de tipos existentes entre los tipos simples con pares, de forma tal que las proposiciones con mismo significado son equivalentes. En este trabajo proponemos una extensión de Sistema I hacia polimorfismo, añadiendo al sistema de tipos el cuantificador universal y sus isomorfismos relacionados.

Palabras Clave

Cálculo Lambda
Teoría de tipos
Isomorfismos de tipos
Polimorfismo
Sistema de pruebas

Conclusiones

Expusimos un problema de rigidez en los sistemas de tipos en lenguajes de programación, y en la relación de derivación en los sistemas de pruebas. Estos impiden que, dadas dos proposiciones equivalentes distintas, una prueba de una constituya una prueba de la otra. Los sistemas módulo isomorfismos se abstraen de las formas de las proposiciones y se centran en su significado. Sistema I es uno de ellos. En este trabajo definimos Sistema I Polimórfico, una extensión del mismo hacia polimorfismo, y probamos su correctitud mediante la propiedad de preservación de tipos.

Trabajos Realizados

Analizamos el trabajo realizado en torno a la familia de sistemas módulo isomorfismos, en particular Sistema I, revisando las técnicas que se aplicaron en su desarrollo. Propusimos una extensión a dicho sistema con polimorfismo, a la que llamamos Sistema I Polimórfico. Describimos las diferentes características particulares del mismo: el conjunto de isomorfismos a identificar, y las consecuencias en la relación de tipado, reducción y equivalencia entre tipos y términos. Expusimos los motivos de las decisiones de inclusión y exclusión de isomorfismos y de equivalencias entre tipos y términos. Probamos la correctitud del cálculo mediante la propiedad de preservación de tipos, que debido a la cantidad de isomorfismos internalizados no resulta trivial.

Trabajos Futuros

Adición de conectivas, como puede ser un elemento neutro para la conjunción $\top$.
Prueba de terminación, aplicando la técnica utilizada para demostrar normalización fuerte en Sistema I.
Prueba de progresión mediante la incorporación de la η-expansión, entre otras reglas.
Extender la implementación existente de Sistema I siguiendo el diseño de Sistema I Polimórfico.
Desarrollo de bibliotecas o extensiones para lenguajes como Haskell, Coq o Agda.

Fecha de la presentación: Febrero de 2020

Agregando polimorfismo a una lógica que identifica proposiciones isomorfas

Cristian Fabián Sottile

Tabla de contenidos

Parte I

Introducción y preliminares

Capítulo 1

Introducción

1.1 Motivación

El tipo de un programa constituye una especificación formal de una porción de su propósito, y en este sentido todos los programas de un cierto tipo comparten dicha porción de propósito. Por ejemplo, en el caso de los lenguajes de primer orden, cuando decimos que la función + tiene tipo $(Entero \times Entero) \to Entero$, estamos especificando que a partir de un par de valores de tipo Entero, denota un Entero. Asimismo, decimos que la función − tiene tipo $(Entero \times Entero) \to Entero$, lo que resulta en la misma especificación parcial que para +. Debido al ampliamente estudiado isomorfismo de Curry (ver, por ejemplo [3] página 9), sabemos que hay definiciones alternativas para las funciones + y − que tienen tipo $Entero \to Entero \to Entero$; esta especificación parcial establece que a partir de un Entero, denotan una función que, a partir de un Entero, denota un Entero. Es interesante observar que en estas dos especificaciones formales aparece la misma idea: denotar un valor de tipo Entero a partir de dos valores de tipo Entero. ¿Cuál es la diferencia entonces entre estas dos especificaciones?

Los tipos de los programas describen clases de problemas. Por ejemplo, $(Entero \times Entero) \to Entero$ describe la clase de los problemas que pueden resolverse a partir de un par de números enteros. De igual manera, debido a la propia definición del isomorfismo de Curry, $Entero \to Entero \to Entero$ describe la clase de los problemas que pueden resolverse a partir de dos números enteros. La diferencia radica en la forma en que al programa se le proveen los dos enteros. En el primer caso es mediante un argumento de tipo par de números enteros; el resultado se conforma a partir de la

combinación de un programa y un valor de tipo Par de números enteros. En el segundo caso es mediante dos argumentos de tipo Entero; el resultado se conforma mediante la combinación de un programa y dos valores (separados) de tipo Entero. Llamaremos *forma* a la manera sintáctica en que se deben combinar los programas para obtener resultados, la cual está determinada por el propio tipo del programa.

Desde una perspectiva semántica, resulta de interés considerar a los programas como soluciones más allá de sus formas. Dentro de este esquema, tanto el caso de $(Entero \times Entero) \to Entero$ como el de $Entero \to Entero \to Entero$ pasan a solucionar la misma clase de problemas: los que a partir de dos números enteros, sin importar si los mismos son obtenidos juntos o separados, denotan un número entero.

Establecimos entonces que cada tipo describe una clase de problemas, y vimos que existe un nivel más de abstracción en estas, que se obtiene al ignorar las formas y observar lo que denominaremos *fondo* (haciendo referencia a su comportamiento) o simplemente significado. El inconveniente de buscar abstraerse de las formas es que en ellas nos basamos para garantizar la correctitud de nuestros lenguajes de programación, ya que los sistemas de tipos clásicos no admiten la combinación de programas de maneras no especificadas explícitamente; por ejemplo, las construcciones no contemplan recibir dos argumentos separados en donde se esperaba un par.

Para poder abstraernos de la forma y centrarnos en el significado de los programas, es necesario establecer cuáles son las formas que son combinables y cómo combinarlas. Para ello nos valdremos de la noción de isomorfismo entre tipos. En términos conceptuales, dos tipos son isomorfos cuando se pueden transformar términos de un tipo a otro sin perder información, es decir manteniendo el significado. Un ejemplo sencillo es la conmutatividad de los pares. El tipo $A \times B$ es isomorfo al tipo $B \times A$, dado que cualquier par $\langle x, y \rangle$ se puede transformar en $\langle y, x \rangle$, y no se perdió información. Un ejemplo algo más complejo, que es con el que trabajamos en los párrafos anteriores, es el isomorfismo de Curry, que relaciona a los tipos $(A \times B) \to C$ y $A \to (B \to C)$. Las funciones $curry : ((A \times B) \to C) \to (A \to (B \to C))$ y $uncurry : (A \to (B \to C)) \to ((A \times B) \to C)$ sirven de testigo para el mismo. Los isomorfismos entonces representan una relación de fondo entre tipos que tienen diferentes formas.

Continuando con el ejemplo de la suma, y considerando

$$suma : Entero \to Entero \to Entero$$
$$suma' : (Entero \times Entero) \to Entero$$
$$x, y : Entero$$

podemos construir las siguientes expresiones válidas, en donde (1) y (2) respetan las formas de *suma* y *suma'*, y (3) y (4) visualizan el fondo compartido y adaptan las formas aprovechando el isomorfismo de Curry.

$$suma\ x\ y \tag{1.1}$$
$$suma'\ \langle x, y\rangle \tag{1.2}$$
$$curry\ suma'\ x\ y \tag{1.3}$$
$$uncurry\ suma\ \langle x, y\rangle \tag{1.4}$$

Debido a la estrecha relación entre la programación y la lógica, que se analizará en detalle en el Capítulo 4, al hablar de tipos y términos estamos hablando también de proposiciones y pruebas. Desde la perspectiva de la lógica, continuando con el ejemplo de la currificación, una prueba de la implicación $(p \wedge q) \Rightarrow r$ necesitará de una prueba del antecedente $p \wedge q$, que a su vez necesitará de una prueba de p y de una prueba de q. De igual manera, una prueba de la implicación $p \Rightarrow (q \Rightarrow r)$ requerirá tanto de una prueba de p como de una prueba de q. Ocurre entonces que cualquier prueba de $(p \wedge q) \Rightarrow r$ permitirá probar $p \Rightarrow (q \Rightarrow r)$, y viceversa. Pero dado que son evidentemente distintas, una prueba de una no constituye en sí misma una prueba de la otra.

Así como en programación, la forma de una prueba consiste en la manera en que puede ser combinada con otras pruebas, y está caracterizada por la proposición correspondiente a la prueba; es por eso que una prueba de $(\tau \wedge \sigma) \Rightarrow \gamma$ no es una prueba de $\tau \Rightarrow (\sigma \Rightarrow \gamma)$. Del mismo modo, lo que llamamos fondo en programación tiene aquí la misma acepción: qué significa una prueba de la proposición. En el caso que venimos tratando, $(\tau \wedge \sigma) \Rightarrow \gamma$ significa que se puede probar r a partir de tener pruebas de p y q, y $\tau \Rightarrow (\sigma \Rightarrow \gamma)$ también singnifica que se puede probar r a partir de tener pruebas de p y q.

El isomorfismo de Curry vale también en la lógica y relaciona a las proposiciones $(A \wedge B) \Rightarrow C$ y $A \Rightarrow (B \Rightarrow C)$. Dos proposiciones son isomorfas cuando se puedan transformar pruebas de una en pruebas de la otra y viceversa, a través de una biyección.

Esta tesina se inscribe en una serie de estudios [1,11,13,27] sobre sistemas que internalizan isomorfismos y permiten que aquellos tipos o proposiciones que sean isomorfos sean considerados equivalentes.

1.2 El cálculo lambda como método de estudio

La computación surge a partir de esfuerzos de la comunidad académica matemática por encontrar respuestas a ciertas preguntas. Siendo los lenguajes de programación sistemas formales, resulta natural que los mismos se estudien mediante el uso de lógica, matemática, y métodos formales en general.

El cálculo lambda es un lenguaje sencillo con solo tres construcciones (variables, abstracciones y aplicaciones) diseñado en la década de 1930, que constituyó uno de los primeros modelos de computación y fue históricamente utilizado para el estudio de los lenguajes de programación funcionales. Existen diversas extensiones y restricciones que retratan diferentes propiedades del mismo. Por ejemplo, el cálculo lambda simplemente tipado incorpora un conjunto de reglas tal que los términos válidos solo son aquellos que se construyan mediante ellas, apuntando a que no sea posible construir programas que no tengan sentido. Esta restricción ha servido de base para muchos de los cálculos que se estudian en la actualidad, incluyendo el sistema en el que se basa este trabajo.

1.3 Nuestro aporte: una extensión polimórfica de Sistema I

Los sistemas módulo isomorfismos [1,11,13,27] son cálculos lambda tipados, a la vez que sistemas de pruebas, que trabajan módulo isomorfismos. En particular, Sistema I es un cálculo lambda simplemente tipado con pares. Dado que establecimos que los isomorfismos caracterizan la equivalencia de significado entre tipos y términos de distinta forma, este sistema ignora las formas de los programas y se centra puramente en el significado, permitiendo efectuar combinaciones tradicionalmente incompatibles como las expuestas previamente. Al mismo tiempo, en su carácter de sistema lógico y entendiendo a los isomorfismos como igualadores de significado de proposiciones, Sistema I ignora la forma de las proposiciones y se centra en su significado. Las consecuencias van desde la comodidad de quien programa o realiza pruebas, que podrá atender únicamente el significado de sus programas o pruebas sin ocuparse de que las formas sean compatibles, hasta la capacidad de optimizar programas o pruebas a partir de las nuevas formas de combinación introducidas.

En términos de expresividad, un siguiente paso natural para un lenguaje de programación simplemente tipado consiste en la adición de polimorfismo.

Desde el punto de vista lógico, esto significa llevarlo de ser un lenguaje capaz de expresar especificaciones formales en primer orden, a uno capaz de expresar especificaciones en segundo orden. La propuesta de este trabajo es agregar polimorfismo a Sistema I, produciendo un nuevo sistema que llamaremos Sistema I Polimórfico. Esto conlleva agregar: el constructor $\forall$ a nivel de tipos; la abstracción y aplicación de tipos a nivel de términos; los isomorfismos faltantes correspondientes al fragmento de la lógica con $\Rightarrow$, $\wedge$ y $\forall$ tanto a nivel de tipos como a nivel de términos; y una prueba de correctitud para el nuevo cálculo. Los aportes de esta tesina fueron publicados [27] en el XXV Congreso Argentino de Ciencias de la Computación (CACIC 2019).

1.4 Estructura del trabajo

En los capítulos 2 y 3 introducimos el cálculo lambda y dos sistemas de tipos: Tipos simples y Sistema F. En el Capítulo 4 analizamos la correspondencia existente entre la programación y la lógica, un aspecto clave de este trabajo. En el Capítulo 5 introducimos el concepto de isomorfismo en los sistemas de tipos, y las caracterizaciones puntuales de los sistemas que nos competen. En el Capítulo 6 introducimos la familia de sistemas módulo isomorfismos en la cual se inscribe el sistema desarrollado en esta tesina. En el Capítulo 7 introducimos Sistema I Polimórfico, un sistema módulo isomorfismos basado en Sistema I y Sistema F.

Capítulo 2

Introducción al cálculo lambda

2.1 Orígenes

Tal como explica Philip Wadler en [28], a comienzos del siglo XX, a partir de la demostración de que la lógica formal podía expresar parte de la matemática, Hilbert y sus colegas se convirtieron en los principales propulsores de la lógica formal, intentando establecer un fundamento lógico firme para la matemática. Propusieron una serie de problemas que hoy se conoce como *Hilbert's Program* (el Programa de Hilbert), entre los cuales, para nuestro interés como computólogos, destaca el llamado *Entscheidungsproblem* (Problema de la Decisión). El mismo consiste en el desarrollo de un procedimiento *efectivamente calculable* para determinar la verdad o falsedad de cualquier sentencia matemática. Este problema suponía la completitud de la matemática, es decir que toda sentencia o su negación posee una prueba, y no requería una definición precisa de "efectivamente calculable". En 1930 Gödel probó que la aritmética es incompleta, y para probar que el Problema de la Decisión era indecidible sí era necesaria una definición formal de efectivamente calculable. En 1936, Alonzo Church introdujo el cálculo lambda, que constituyó, junto a las funciones recursivas de Gödel y las máquinas de Turing de Turing, una de las primeras definiciones formales de un procedimiento efectivamente calculable. Es también la base teórica de los lenguajes de programación del paradigma funcional, y al día de hoy mantiene su vigencia como artefacto de estudio de las características y propiedades de distintos sistemas.

2.2 Definición

El cálculo lambda es una teoría de funciones, es decir una formalización de las funciones matemáticas, que considera su aspecto computacional (para una introducción más completa al cálculo lambda, ver por ejemplo [2, 14]). Es un sistema formal basado en la abstracción y aplicación de funciones que permite representar todas las funciones computables, y dado que estas son las únicas construcciones disponibles, todos los datos y estructuras de interés se representan mediante ellas. Los elementos semánticos a los que harán referencia las expresiones del lenguaje serán *variables*, *funciones* (o *abstracciones*) y *aplicaciones*. Los elementos sintácticos que se usarán para representarlos serán:

- Para las *variables*: las letras w, x, y, z.
- Para las abstracciones: un símbolo λ, una variable seguida de un punto que constituirá el *parámetro* de la función, y una expresión que constituirá el *cuerpo* de la función. La idea es que una abstracción *liga* a su parámetro con su cuerpo, es decir, establece una conexión directa entre la variable y los usos de dicha variable en el cuerpo, que estará dada por el eventual reemplazo del parámetro formal por el parámetro actual en el cuerpo. Por ejemplo, $\lambda x.x$ es la función que liga x en x.
- Para la aplicación: la yuxtaposición de las expresiones. Por ejemplo, si r y s son expresiones, entonces rs es la aplicación de la función r al argumento s. Llamaremos a las expresiones λ-términos (o simplemente términos) y usaremos las letras r, s, t, u, v para referenciarlos.

La gramática que define a los términos bien formados entonces será

$$t ::= x \mid \lambda x.t \mid tt$$

donde x pertenece Var, el conjunto infinito de variables.

Podemos ver que el cálculo no nos permite producir funciones de varios parámetros de manera directa. En cambio, una función de n parámetros se escribe como una función que toma un parámetro y da como resultado una función de $n-1$ parámetros. Así, por ejemplo, suponiendo que contamos con una forma de codificar el operador de suma, la función que suma sus dos parámetros se escribiría $\lambda x.(\lambda y.(x+y))$. Consecuentemente, la aplicación de una función de varios parámetros definida en el cálculo lambda se llevará a cabo mediante la aplicación sucesiva de sus argumentos; es decir, una función de n parámetros se aplicará con un parámetro, resultando en una función

de $n-1$ parámetros, que se aplicará con el segundo parámetro, resultando en una función de $n-2$ parámetros. Siguiendo el ejemplo de la función que suma, suponiendo que contamos con los números naturales además del operador de suma, la aplicación consistirá en $((\lambda x.(\lambda y.(x+y)))1)2$. Tomaremos de aquí en adelante la convención de que los paréntesis asocian a derecha en la abstracción de funciones y a izquierda en la aplicación de funciones a argumentos, por lo tanto reescribiremos los términos anteriores como $\lambda x.\lambda y.x+y$ y $(\lambda x.\lambda y.x+y)\ 1\ 2$.

Ejemplos de términos válidos:

- x: una variable.
- $\lambda x.x$: la función que toma un parámetro x y da como resultado x. La llamaremos *id*.
- $(\lambda x.x)y$: la aplicación de *id* a y.
- $\lambda x.\lambda y.x$: la función que toma un parámetro x y un parámetro y, ignora y y da como resultado x. La llamaremos *true*.
- $(\lambda x.\lambda y.x)wz$: la aplicación de *true* a w y z.
- $\lambda x.\lambda y.xy$: la función que toma un parámetro x y un parámetro y, y aplica x a y. La llamaremos *apply*.
- $(\lambda x.\lambda y.xy)(\lambda x.x)x$: la aplicación de *apply* a *id* y x.

En el cálculo lambda los nombres elegidos para las variables no tienen relevancia. Existe una equivalencia entre términos llamada *α-equivalencia* que expresa esta característica, de forma tal que, por ejemplo, la función $\lambda x.x$ es considerada equivalente a $\lambda y.y$. Usualmente, los cálculos lambda se consideran módulo α-equivalencia.

2.3 Computación

Definiremos primero la función que denota las variables que se encuentran libres en el término recibido como parámetro. Una variable ligada es aquella que se encuentra en un término que es el cuerpo de una función que liga dicha variable. Una variable libre es una variable que no se encuentra en el cuerpo de una función que la liga. Por ejemplo, la variable x está ligada en

$\lambda x.x$, y la variable y está libre en $\lambda x.y$. La función fv (por las siglas de *free variables*) se define entonces como sigue:

$$\begin{aligned} fv(x) &= \{x\} \\ fv(\lambda x.r) &= fv(r) - \{x\} \\ fv(rs) &= fv(r) \cup fv(s) \end{aligned}$$

El primer caso establece que las variables libres de una variable suelta es el conjunto unitario de justamente esa variable suelta. El segundo caso, que las variables libres de una abstracción son las variables libres de su cuerpo, quitando la variable ligada por la abstracción. El último caso, que las variables libres en una aplicación entre dos términos consiste en la unión entre los conjuntos de variables libres de ambos términos.

2.3.1 Sistemas de reescritura

Los *sistemas de reescritura de términos* (TRS por sus siglas en inglés) son relaciones matemáticas compuestas por:

- Un conjunto de términos $Ter(\Sigma)$ definido de forma inductiva a partir de un alfabeto Σ.
- Un conjunto R de reglas de reescritura o reducción que definen una relación binaria entre términos. Si para dos términos $r, s \in Ter(\Sigma)$ existe $(r, s) \in R$ dado por la regla R_i, escribiremos $r \to_{R_i} s$ y diremos que r reduce en un paso a s.

A continuación presentaremos algunas definiciones. Sea $\mathcal{T} = (T, \{\to\})$ un TRS:

- $\to^+$ es la *clausura transitiva* de $\to$, es decir que si $r \to s$ y $s \to t$, entonces $r \to^+ t$.
- $\to^*$ es la *clausura reflexiva transitiva* de $\to$, es decir que si $r \to s$ y $s \to t$, entonces $r \to^+ t$, y para todo r, $r \to^* r$.
- Un término $r \in T$ es una *forma normal* (f.n.) si no existe $s \in T$ tal que $r \to s$.
- Un término $r \in T$ tiene forma normal si existe $s \in T$ tal que s es una formal normal y $r \to^* s$.

- Un término $r \in T$ es *fuertemente normalizante* si toda secuencia de reducción comenzada en r es finita y termina en una forma normal. Llamamos secuencia de reducción a una secuencia de términos $r_0, r_1, \ldots, r_n$ tal que $r_i \to r_{i+1}$. Decimos que el tamaño de dicha secuencia es n, ya que la misma se forma a partir de n pasos de reducción.
- $\to$ es *débilmente normalizante* (WN por sus siglas en inglés) si todo $r \in T$ tiene una forma normal. En este ejemplo, $\mathcal{T}$ sería también WN.
- $\to$ es *fuertemente normalizante* (SN por sus siglas en inglés) si toda posible secuencia de reducciones en el sistema eventualmente termina. En este ejemplo, $\mathcal{T}$ sería también SN.

2.3.2 Reescritura en el cálculo lambda

El cálculo lambda es un TRS, lo que permite que se lo pueda utilizar para computar. En su forma más básica se compone del conjunto de los λ-términos y un conjunto de reglas de reducción con una regla principal, la regla β, y reglas de congruencia.

La β-reducción establece que un término que consiste en la aplicación de una función a un argumento, reduce al cuerpo de la función donde la variable ligada por la función se substituyó por el argumento. También la llamaremos β-conversión y β-substitución. Formalmente se escribe:

$$(\lambda x.r)s \to_\beta [x := s]r$$

Dado cualquier λ-término, podemos identificar en su interior un conjunto de términos reducibles que llamaremos *redexes* por su nombre en inglés (*reducible expressions*). Un *redex* es un término al que se le puede aplicar una regla de reducción. Por ejemplo, bajo la β-conversión, en el término $\lambda z.(\lambda x.(\lambda y.y)x)z$ tenemos dos redexes, $(\lambda y.y)x$ y $(\lambda x.(\lambda y.y)x)z$, pero dado que β está definida sobre aplicaciones de funciones y este término es una abstracción, no es posible reducirlo. A su vez, tomando como ejemplo el término $true\ w\ z = (\lambda x.\lambda y.x)wz$, que representa la aplicación de la función $true\ w$ al argumento z, vemos que no podemos aplicar la β-reducción ya que $true\ w$, si bien reduce a una función, no tiene forma sintáctica de función. Si observamos los redexes, encontramos solo uno: $(\lambda x.\lambda y.x)w$. Estamos entonces ante el inconveniente de que un término como $true\ w\ z$ no puede reescribirse porque no tiene la forma sintáctica esperada por la regla β. Las reglas de congruencia solucionan este problema, permitiendo reducir términos que no consistan sintácticamente en una función aplicada a un argumento.

En λ-cálculo las reglas de congruencia son:

$$\frac{r \to s}{\lambda x.r \to \lambda x.s}(C_1) \qquad \frac{r \to s}{rt \to st}(C_2) \qquad \frac{r \to s}{tr \to ts}(C_3)$$

Obtenemos entonces las siguientes reducciones para los ejemplos anteriores.

- x no reduce a ningún término, está en forma normal.
- $\lambda x.x$ está en forma normal.
- $(\lambda x.x)y \to_\beta [x := y]x$
- $\lambda x.\lambda y.x$ está en forma normal.
- $(\lambda x.\lambda y.x)wz \to_{\beta,(C_2)} ([x := w]\lambda y.x)z$
- $\lambda x.\lambda y.xy$ está en formal normal.
- $(\lambda x.\lambda y.xy)(\lambda x.x)x \to_{\beta,(C_2)} ([x := \lambda x.x]\lambda y.xy)x$

Para lograr resultados reales en términos computacionales, resta definir la función de substitución $[x := s]r$.

$$\begin{aligned}
&[x := r]x = r \\
&[x := r]y = y \\
&[x := r]\lambda x.s = s \\
&[x := r]\lambda y.s = \lambda y.[x := r]s && y \notin fv(r) \\
&[x := r]\lambda y.s = \lambda z.[x := r][y := z]s && y \in fv(r) \\
&[x := r]st = ([x := r]s)([x := r]t)
\end{aligned}$$

Dada la substitución de la variable x por el término r, esta definición establece que:

- Cuando el término sobre el que se quiere efectuar la substitución es una variable, se la analiza en comparación con x: si es también x, se substituye dando como resultado r; si es una variable distinta de x, por ejemplo y, se descarta la substitución dando como resultado y.

- Cuando el término es una abstracción, se analiza la variable que liga: si es x, se descarta la substitución dando como resultado el mismo término. Si es distinta a x, por ejemplo y, se analiza la pertenencia de y al conjunto de variables libres de r: si no pertenece, es seguro reemplazar x por r en el cuerpo de la función; si pertenece, el reemplazo produciría que la y libre en r pase a estar ligada en el cuerpo de la función y esto cambiaría el significado del término (este fenómeno se denomina *captura de variable*), por lo que se efectúa un cambio de nombre de la variable ligada por la función, y luego se prosigue con la substitución. Cabe destacar que la variable z simboliza una variable *nueva*, es decir no utilizada en otra parte del término.
- Cuando el término es una aplicación, se realiza la substitución en ambos subtérminos de la misma.

Llamaremos *evaluación* a la secuencia de reducción desde un término hasta alcanzar una forma normal. Habiendo dado una definición para la substitución, podemos dar una evaluación para los ejemplos previos:

- $(\lambda x.x)y$

  $\rightarrow_\beta$
  $[ax := y](x)$
  $=$ def. *subst*, caso 1
  y

- $(\lambda x.\lambda y.x)wz$

  $\rightarrow_\beta,(C_2)$
  $([x := w]\lambda y.x)z$
  $=$ def. *subst*, caso 4
  $(\lambda y.[x := w]x)z$
  $=$ def. *subst*, caso 1
  $(\lambda y.w)z$
  $\rightarrow_\beta$
  $[y := z]w$
  $=$ def. *subst*, caso 2
  w

- $(\lambda x.\lambda y.xy)(\lambda x.x)x$

  $\rightarrow_\beta$,(C_2)
  $([x := \lambda x.x]\lambda y.xy)x$
  $=$ def. *subst*, caso 4
  $(\lambda y.[x := \lambda x.x]xy)x$
  $=$ def. *subst*, caso 6
  $(\lambda y.([x := \lambda x.x]x)([x := \lambda x.x]y))x$
  $=$ def. *subst*, caso 1
  $(\lambda y.(\lambda x.x)([x := \lambda x.x]y))x$
  $=$ def. *subst*, caso 2
  $(\lambda y.(\lambda x.x)y)x$
  $\rightarrow_\beta$
  $[y := x]((\lambda x.x)y)$
  $=$ def. *subst*, caso 6
  $([y := x]\lambda x.x)([y := x]y)$
  $=$ def. *subst*, caso 2
  $(\lambda x.x)([y := x]y)$
  $=$ def. *subst*, caso 1
  $(\lambda x.x)x$
  $\rightarrow_\beta$
  $[x := x]x$
  $=$ def. *subst*, caso 1
  x

2.3.3 Estrategias de reducción

Debido a la potencial existencia de múltiples *redexes* en un término, resulta de interés la definición de estrategias para la elección de un *redex* particular al momento de efectuar un paso de reducción. Estas estrategias constituirán funciones de términos a términos que vinculan un término con el *redex* que corresponde que sea reducido.

Veremos que la existencia de términos con secuencias de reducción infinitas juega un rol importante al momento de evaluarlas. Para introducir la noción utilizaremos el término $\Omega = (\lambda x.xx)(\lambda x.xx)$, para el cual un paso

de reducción basta para visualizar el caracter infinito de su única secuencia de reducción posible.

$$\begin{aligned}
& (\lambda x.xx)(\lambda x.xx) \\
\rightarrow_\beta & \\
& [x := \lambda x.xx]xx \\
= & \text{ def. } \textit{subst}, \text{ caso } 6 \\
& ([x := \lambda x.xx]x)([x := \lambda x.xx]x) \\
= & \text{ def. } \textit{subst}, \text{ caso } 1 \\
& (\lambda x.xx)(\lambda x.xx)
\end{aligned}$$

Existen algunas estrategias clásicas, que son las que describiremos. Cada una tiene una serie de consecuencias que las hacen más o menos adecuadas para diferentes contextos. Nos valdremos de la modificación de las reglas de reducción para definir las estrategias presentadas.

Call-by-value En esta estrategia se reducen primero los argumentos y luego las aplicaciones. Por ejemplo:

$$\begin{aligned}
& (\lambda y.y)((\lambda x.x)x) \\
\rightarrow_{\text{call-by-value}} & \\
& (\lambda y.y)([x := x]x) \\
= & \text{ def. } \textit{subst}, \text{ caso } 1 \\
& (\lambda y.y)x \\
\rightarrow_{\text{call-by-value}} & \\
& [y := x]y \\
= & \text{ def. } \textit{subst}, \text{ caso } 1 \\
& x
\end{aligned}$$

Esto se logra mediante la modificación de la regla β agregando la restricción de que el argumento debe estar en forma normal, con lo que las reglas de reducción quedan conformadas por:

$$\text{si } s \text{ está en f.n.} \quad (\lambda x.r)s \rightarrow_\beta [x := s]r$$

$$\frac{r \rightarrow s}{\lambda x.r \rightarrow \lambda x.s}(C_1) \qquad \frac{r \rightarrow s}{rt \rightarrow st}(C_2) \qquad \frac{r \rightarrow s}{tr \rightarrow ts}(C_3)$$

La principal ventaja de la estrategia call-by-value es que los argumentos a funciones se evalúan por única vez. En los casos en donde la variable ligada por una función aparece n veces en el cuerpo, haber evaluado el argumento previamente ahorra las $n - 1$ evaluaciones restantes, que son idénticas.

Call-by-name En esta estrategia se reducen primero las aplicaciones y luego los argumentos. Por ejemplo:

$$
\begin{aligned}
& (\lambda y.y)((\lambda x.x)x) \\
\rightarrow_{\text{call-by-name}} \\
& [y := (\lambda x.x)x]y \\
= \ & \text{def. } \textit{subst}\text{, caso 1} \\
& (\lambda x.x)x \\
\rightarrow_{\text{call-by-name}} \\
& [x := x]x \\
= \ & \text{def. } \textit{subst}\text{, caso 1} \\
& x
\end{aligned}
$$

Esto se logra mediante la eliminación de la regla de congruencia (C_3), que permite reducir el argumento en una aplicación, con lo que las reglas de reducción quedan conformadas por:

$$(\lambda x.r)s \rightarrow_\beta [x := s]r$$

$$\frac{r \rightarrow s}{\lambda x.r \rightarrow \lambda x.s}(C_1) \qquad \frac{r \rightarrow s}{rt \rightarrow st}(C_2)$$

La principal ventaja de esta estrategia es la posibilidad de computar utilizando términos con secuencias de reducción infinitas. Por ejemplo, *true* $x\ \Omega \rightarrow_\beta^* x$. El teorema de estandarización establece que si un término tiene forma normal, call-by-name la encuentra.

Reducción débil (weak reduction) Esta es una estrategia adicional que resulta transversal a las anteriores, ya que solo establece la no reducción bajo lambda, es decir la no reducción de un término consistente en una abstracción. La reducción de abstracciones se corresponde con la idea de optimización de programa, en donde partes de un programa son reescritas previamente a su ejecución. Hay una regla en particular que determina si una estrategia es débil o no: (C_1). Cuando no está presente, la estrategia es

débil, y cuando está no lo es. Las reglas de las estrategias anteriores en su formato débil son, entonces:

- Weak call-by-value:

$$\text{si } s \text{ está en f.n.} \quad (\lambda x.r)s \to_\beta [x := s]r$$

$$\frac{r \to s}{rt \to st}(C_2) \qquad \frac{r \to s}{tr \to ts}(C_3)$$

- Weak call-by-name:

$$(\lambda x.r)s \to_\beta [x := s]r \qquad \frac{r \to s}{rt \to st}(C_2)$$

Capítulo 3

Cálculo lambda y sistemas de tipos

3.1 Sistemas de tipos

El tipado en los sistemas formales consiste en la asignación de elementos de un conjunto de tipos a cada término válido, lo que permite efectuar una separación entre diferentes conjuntos de valores. Desde la perspectiva de los lenguajes de programación, la idea de tipado consiste en la incorporación de un sistema formal –el sistema de tipos– cuya principal característica es que permite distinguir entre dos clases de programas: los *bien tipados* que son aceptados por el sistema, y los que no tienen un tipo y por lo tanto son rechazados por el sistema. Los beneficios que se desprenden de la existencia de un sistema de tipos en un lenguaje son:

- Detección de errores en compilación: errores sencillos como aplicación de funciones con argumentos de tipos equivocados pueden ser detectados antes de la ejecución.
- Mejora del rendimiento: la afirmación por parte del sistema de tipos de que los términos cumplen con las reglas de tipado constituyen una demostración formal, por lo que permiten prescindir de las verificaciones de tipos en tiempo de ejecución, que de otra forma deberían realizarse.
- Documentación: los tipos de los términos constituyen una especificación formal parcial de su naturaleza y comportamiento, por lo que su propia presencia añade información importante.

La definición de un sistema de tipos quedará marcada por elecciones en las restricciones impuestas sobre los términos tipables, las cuales determinarán el nivel de expresividad y seguridad del lenguaje de programación. Jones describe [20, página 2] los cuatro factores más importantes en la la definición de un sistema de tipos: la flexibilidad de aceptar una cantidad razonablemente grande de programas y no imponer demasiadas restricciones, la existencia de un procedimiento de chequeo de tipos que funciones de manera aceptable, que el mismo exhiba un comportamiento intuitivo y quien programa pueda predecir cuáles programas serán aceptados y cuáles no, y la precisión con la que los tipos reflejan las propiedades de sus elementos asociados.

3.2 Cálculo lambda simplemente tipado

La definición formal de un sistema de tipos involucra especificar: el conjunto de los tipos; las reglas de producción de juicios de tipado válidos; y la demostración formal de ciertas propiedades.

3.2.1 El conjunto de los tipos

En este cálculo, el conjunto de los tipos está definido de manera inductiva como sigue:

$$A \quad := \quad \tau \mid A \to A$$

Utilizaremos las primeras letras del abecedario en mayúscula para referirnos a tipos cualesquiera.

3.2.2 El conjunto de los términos

Los términos serán los del cálculo lambda simplemente tipado, con la modificación de que se le indicará a las variables el tipo que le corresponde:

$$t ::= x^A \mid \lambda x^A.t \mid tt$$

La notación que agrega el tipo explícitamente a las variables se denomina *à la Church*. Omitimos el tipo en variables cuando es fácilmente deducible del contexto o cuando no resulte relevante, y por ejemplo escribimos $\lambda x^A.x$ en lugar de $\lambda x^A.x^A$.

3.2.3 Reglas de tipado

Un *juicio* en un sistema de tipos es una relación entre un contexto, un término y un tipo, y se escribe

$$\Gamma \vdash r : A$$

donde Γ es un *contexto* conformado por pares $x : B$ de variables y tipos. Este juicio establece que un término r tiene tipo A dadas las asunciones establecidas en Γ. A su vez, al escribir $s : B$ estaremos implicando que existe un contexto Γ tal que $\Gamma \vdash s : B$. Utilizaremos las letras griegas mayúsculas Γ, Δ, Φ para referirnos a contextos.

Cada sistema de tipos establece un conjunto de aquellos juicios que son válidos en el mismo. Este conjunto queda determinado por las reglas de tipado o de producción de juicios. En el cálculo lambda, estas reglas son las siguientes:

- $\Gamma, x : A \vdash x : A$
- Si $\Gamma, x : A \vdash r : B$, entonces $\Gamma \vdash \lambda x^A.r : A \to B$
- Si $\Gamma \vdash r : A \to B$ y $\Gamma \vdash s : A$, entonces $\Gamma \vdash rs : B$

donde $\Gamma, x : A$ representa $\Gamma \cup \{x : A\}$. La primera regla establece que si el contexto indica que x es de tipo A, entonces podemos deducir que en ese contexto x es de tipo A. La segunda regla, que si dado el contexto x es de tipo A, se tiene que r es de tipo B, entonces la función $\lambda x^A.r$ es de tipo $A \to B$ y no se requiere $x : A$ en el contexto; esto significa que $\lambda x^A.r$ es una función que recibe como argumento un valor de tipo A y tiene como cuerpo un término de tipo B. La tercera regla, que si dado un contexto se tiene que r es de tipo $A \to B$ y s de tipo A, entonces, en el mismo contexto, la aplicación de r a s tiene tipo B; esto significa que aplicar una función con un argumento del tipo adecuado produce un valor del tipo del cuerpo de la función, lo cual se condice con las reglas de reducción del cálculo.

La forma usual de expresar derivaciones de juicios es mediante árboles de las reglas aplicadas, por lo cual escribiremos las reglas de producción como sigue:

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

$$\frac{\Gamma, x : A \vdash r : B}{\Gamma \vdash \lambda x^A.r : A \to B}(\to_i)$$

$$\frac{\Gamma \vdash r : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash rs : B}(\to_e)$$

3.2.4 Derivaciones de tipos

La derivación de un tipo consiste en una demostración formal de que un juicio dado es válido. Lo expresaremos en forma de árbol donde cada nodo es un juicio y la raiz es el juicio a probar, y donde cada hoja debe ser un juicio obtenido mediante la regla (ax) o bien una hipótesis. Un juicio es válido si y solo si existe un árbol de derivación válido para este. Para ilustrar el concepto retomaremos algunos de los ejemplos vistos en la sección 2:

- x:

$$\frac{}{x : \tau \vdash x : \tau}\ (ax)$$

- $\lambda x^\tau.x$:

$$\frac{\dfrac{}{x : \tau \vdash x : \tau}\ (ax)}{\vdash \lambda x^\tau.x : \tau \to \tau}\ (\to_i)$$

- $(\lambda x^\tau.x)y$:

$$\frac{\dfrac{\dfrac{}{x : \tau \vdash x : \tau}\ (ax)}{\vdash \lambda x^\tau.x : \tau \to \tau}\ (\to_i) \qquad \dfrac{}{y : \tau \vdash y : \tau}\ (ax)}{y : \tau \vdash (\lambda x^\tau.x)y : \tau}\ (\to_e)$$

- $\lambda x^\tau.\lambda y^{\tau\to\tau}.x$:

$$\frac{\dfrac{\dfrac{}{x : \tau, y : \tau \to \tau \vdash x : \tau}\ (ax)}{x : \tau \vdash \lambda y^{\tau\to\tau}.x : (\tau \to \tau) \to \tau}\ (\to_i)}{\vdash \lambda x^\tau.\lambda y^{\tau\to\tau}.x : \tau \to (\tau \to \tau) \to \tau}\ (\to_i)$$

- $(\lambda x^{\tau}.\lambda y^{\tau\to\tau}.x)wz$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x:\tau, y:\tau\to\tau\vdash x:\tau}\,(ax)}{x:\tau\vdash \lambda y^{\tau\to\tau}.x:\tau\to\tau\to\tau}\,(\to_i)}{\vdash \lambda x^{\tau}.\lambda y^{\tau\to\tau}.x:\tau\to\tau\to\tau\to\tau}\,(\to_i) \qquad \dfrac{}{w:\tau\vdash w:\tau}\,(ax)}{w:\tau\vdash (\lambda x^{\tau}.\lambda y^{\tau\to\tau}.x)w:\tau\to\tau\to\tau}\,(\to_e)}{(1)}$$

$$\dfrac{\dfrac{(1)}{w:\tau\vdash (\lambda x^{\tau}.\lambda y^{\tau\to\tau}.x)w:\tau\to\tau\to\tau} \qquad \dfrac{}{z:\tau\to\tau\vdash z:\tau\to\tau}\,(ax)}{w:\tau, z:\tau\to\tau\vdash (\lambda x^{\tau}.\lambda y^{\tau\to\tau}.x)wz:\tau}\,(\to_e)$$

- $\lambda x^{\tau\to\tau\to\tau}.\lambda y^{\tau}.xy$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x:\tau\to\tau\to\tau, y:\tau\vdash x:\tau\to\tau\to\tau}\,(ax) \qquad \dfrac{}{y:\tau\vdash y:\tau}\,(ax)}{x:\tau\to\tau\to\tau, y:\tau\vdash xy:\tau\to\tau}\,(\to_e)}{x:\tau\to\tau\to\tau\vdash \lambda y^{\tau}.xy:\tau\to\tau\to\tau}\,(\to_i)}{\vdash \lambda x^{\tau\to\tau\to\tau}.\lambda y^{\tau}.xy:(\tau\to\tau\to\tau)\to\tau\to\tau\to\tau}\,(\to_i)$$

- $(\lambda x^{\tau\to\tau}.\lambda y^{\tau}.xy)(\lambda x^{\tau}.x)x$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x:\tau\to\tau\vdash x:\tau\to\tau}\,(ax) \qquad \dfrac{}{y:\tau\vdash y:\tau}\,(ax)}{x:\tau\to\tau, y:\tau\vdash xy:\tau}\,(\to_e)}{x:\tau\to\tau\vdash \lambda y^{\tau}.xy:\tau\to\tau}\,(\to_i)}{\vdash \lambda x^{\tau\to\tau}.\lambda y^{\tau}.xy:(\tau\to\tau)\to\tau\to\tau}\,(\to_i) \qquad \dfrac{\dfrac{}{x:\tau\vdash x:\tau}\,(ax)}{\vdash \lambda x^{\tau}.x:\tau\to\tau}\,(\to_i)}{\vdash (\lambda x^{\tau\to\tau}.\lambda y^{\tau}.xy)(\lambda x^{\tau}.x):\tau\to\tau}\,(\to_e)$$

$$(1)$$

$$\dfrac{\dfrac{(1)}{\vdash (\lambda x^{\tau\to\tau}.\lambda y^{\tau}.xy)(\lambda x^{\tau}.x):\tau\to\tau} \qquad \dfrac{}{x:\tau\vdash x:\tau}\,(ax)}{x:\tau\vdash (\lambda x^{\tau\to\tau}.\lambda y^{\tau}.xy)(\lambda x^{\tau}.x)x:\tau}\,(\to_e)$$

Aplicadas estas restricciones, se puede observar que ahora un término sin sentido como Ω no puede tiparse y por lo tanto no es válido.

3.2.5 Propiedades de interés

Enunciamos a continuación las propiedades que resultarán de interés para el desarrollo de esta tesina.

Generación Establece las conclusiones que pueden obtenerse a partir de la observación de un juicio, acerca de cómo el mismo fue generado.

Lema 3.2.1 (Generación).

1. *Si* $\Gamma \vdash x : A$, *entonces* $x : A \in \Gamma$
2. *Si* $\Gamma \vdash \lambda x^A.r : C$, *entonces existe* B *tal que* $\Gamma, x : A \vdash r : B$ *y* $C = A \to B$
3. *Si* $\Gamma \vdash rs : B$, *entonces existe* A *tal que* $\Gamma \vdash r : A \to B$ *y* $\Gamma \vdash s : A$

Prueba. Por inducción en la estructura del juicio. □

Substitución Establece que substituir en un término una variable por un término del mismo tipo que la variable, no modifica el tipo del primero.

Lema 3.2.2 (Substitución).
Si $\Gamma, x : A \vdash r : B$ *y* $\Gamma \vdash s : B$, *entonces* $\Gamma \vdash [x := s]r : B$

Prueba. Por inducción en la estructura de r. □

Preservación de tipos Establece la preservación del tipo a través de la reducción, es decir, todo término tiene el mismo tipo que cualquier término al que pueda reducir.

Teorema 3.2.3 (Preservación de tipos).
Si $\Gamma \vdash r : A$ *y* $r \to s$, *entonces* $\Gamma \vdash s : A$.

Prueba. Por inducción en la relación $\to$. □

3.3 Sistema F

Sistema F [18] es una extensión del cálculo lambda simplemente tipado que permite expresar *polimorfismo*. Se lo llama también cálculo lambda tipado de segundo orden o cálculo lambda polimórfico. La idea del polimorfismo reside en la abstracción de tipos, que consiste en agregar la noción de variable de tipo, y permitir que se pase como parámetro el tipo efectivo.

3.3.1 Tipos y términos

El conjunto de los tipos del cálculo lambda simplemente tipado se ve modificado, primeramente, por la incorporación de las variables de tipo. En lugar de tener un tipo básico τ como antes, se utilizarán variables de tipo que serán notadas con las últimas letras del abecedario en mayúscula. Además se le agrega la construcción que indica abstracción de tipos, representado con el símbolo del cuantificador universal $\forall$. El conjunto queda definido como sigue:

$$A \quad := \quad X \mid A \to A \mid \forall X.A$$

donde X pertenece a TVar, el conjunto infinito de variables de tipo.

Por otro lado, el conjunto de los términos sufre también modificaciones. Se agrega un constructor para la abstracción y un constructor para la aplicación. La abstracción se notará con un Λ y la aplicación con un término seguido del argumento de tipo entre corchetes. Queda definido entonces de la siguiente manera:

$$t ::= x^A \mid \lambda x^A.t \mid tt \mid \Lambda X.t \mid t[A]$$

La α-equivalencia vista en el Capítulo 2 se hace extensiva a este sistema, donde, por ejemplo, la abstracción $\Lambda Y.\lambda x^Y.x$ se considera equivalente a $\Lambda Z.\lambda x^Z.x$.

3.3.2 Funciones relevantes

Definimos a continuación un conjunto de funciones que resultan de importancia para las propias definiciones del sistema.

Variables de término libres en términos La función FV_t toma un término y denota el conjunto de todas aquellas variables de término que se encuentran libres en dicho término. Se abrevia FV_t por la traducción al inglés de "variables libres en términos": *"free variables in terms"*. Se define como sigue:

$$\begin{aligned}
&FV_t(x^A) = \{x^A\} \\
&FV_t(\lambda x^A.r) = FV_t(r) - \{x^A\} \\
&FV_t(rs) = FV_t(r) \cup FV_t(s) \\
&FV_t(\Lambda X.r) = FV_t(r) \\
&FV_t(r[A]) = FV_t(r)
\end{aligned}$$

Variables de tipo libres en términos La función FTV_t toma un término y denota el conjunto de las variables de tipo que son el tipo de variables libres en dicho término. Para ello utiliza la función auxiliar FTV'_t que toma un conjunto de variables de término cuyos tipos son variables de tipo, y denota el conjunto de las variables de tipo de dichas variables de término. El nombre de la función está dado por *"free type variables in terms"*, "variables de tipo libres en términos" en inglés. FTV'_t se define como sigue:

$$FTV'_t(\{x\}) = \{typeof\, x\}$$
$$FTV'_t(\{x, x_0, x_1, ..., x_n\}) = \{typeof\, x\} \cup FTV'_t(\{x_0, x_1, ..., x_n\})$$
$$\text{donde } typeof(x^A) = A$$

Luego FTV_t se define como la composición entre FTV'_t y FV_t :

$$FTV_t(t) = FTV'_t(FV_t(t))$$

Variables de tipo libres en tipos La función FTV_A toma un tipo y denota el conjunto de las variables de tipo (es decir $X, Y, Z, \dots$) libres en dicho tipo. Su nombre está dado por *"free type variables in types"*, inglés de "variables de tipo libres en tipos". Se define como sigue:

$$FTV_A(X) = \{X\}$$
$$FTV_A(A \to B) = FTV_A(A) \cup FTV_A(B)$$
$$FTV_A(\forall X.A) = FTV_A(A) - \{X\}$$

Variables de tipo libres en contextos La función FTV_c toma un contexto y denota el conjunto de las variables de tipo que se encuentran libres en dicho contexto. El nombre de la función está dado por *"free type variables in contexts"*, "variables de tipo libres en contextos" en inglés. FTV_c se define como sigue:

$$FTV_c(x : A) = FTV_A(A)$$
$$FTV_c(\Gamma, x : A) = FTV_c(\Gamma) \cup FTV_A(A)$$

Substitución de variables de término La función $[x := t]r$ toma una variable de término x, un término t y un término r y denota el término r en el que se substituyó cada ocurrencia libre de x por t. Se define por inducción sobre el término r.

$$[x := t](x) = t$$

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

$$\frac{\Gamma, x : A \vdash r : B}{\Gamma \vdash \lambda x^A.r : A \to B}(\to_i) \qquad \frac{\Gamma \vdash r : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash rs : B}(\to_e)$$

$$\frac{\Gamma \vdash r : A \qquad X \notin FTV_c(\Gamma)}{\Gamma \vdash \Lambda X.r : \forall X.A}(\forall_i) \qquad \frac{\Gamma \vdash r : \forall X.A}{\Gamma \vdash r[B] : [X := B]A}(\forall_e)$$

Figura 3.1: Reglas de tipado de Sistema F

$$\begin{aligned}
&[x := t](y) = y \\
&[x := t](\lambda x^A.r) = \lambda x^A.r \\
&[x := t](\lambda y^A.r) = \lambda y^A.[x := t]r && \text{si } y \notin FV_t(t) \\
&[x := t](\lambda y^A.r) = [x := t](\lambda z^A.[y := z]r) && \text{si } y \in FV_t(t) \\
&[x := t](\Lambda X.r) = \Lambda X.[x := t]r \\
&[x := t](r[A]) = ([x := t]r)[A]
\end{aligned}$$

Substitución de variables de tipo La función $[X := A]B$ toma una variable de tipo X, un tipo A y un tipo B y denota el tipo B en el que se substituyó cada ocurrencia libre de X por A. Se define por inducción sobre el tipo B.

$$\begin{aligned}
&[X := A](X) = A \\
&[X := A](Y) = Y \\
&[X := A](B \to C) = [X := A]B \to [X := A]C \\
&[X := A](\forall X.B) = \forall X.B \\
&[X := A](\forall Y.B) = \forall Y.[X := A]B && \text{si } Y \notin FTV_A(A) \\
&[X := A](\forall Y.B) = [X := A](\forall Z.[Y := Z]B) && \text{si } Y \in FTV_A(A)
\end{aligned}$$

3.3.3 Reglas de tipado

A las reglas de tipado del cálculo lambda simplemente tipado se le agregan las dos correspondientes al nuevo constructor de tipos $\forall$, una para introducirlo y otra para eliminarlo, de forma que las reglas de Sistema F son las de la Figura 6.5.

La regla $(\forall_i)$ indica que si un término r tiene tipo A y la variable X no se encuentra libre en Γ, entonces se puede abstraer r produciendo el término $\Lambda X.r$ con tipo $\forall X.A$. La regla $(\forall_e)$ dice que si un término r tiene tipo $\forall X.A$, entonces se puede aplicar r pasándole como argumento un tipo B, produciendo el término $r[B]$ cuyo tipo es A modificado cambiando todas las apariciones de X por B, lo que se expresa como $[X := B]A$.

3.3.4 Reglas de reducción

Dado que agregamos constructores de términos y una regla de eliminación de un constructor de tipos, debemos indicar cuál es la semántica operacional de los términos formados a partir de una derivación que aplica la regla de eliminación del constructor; es decir, cómo reducen dichos términos. En este caso la reducción asociada a la eliminación de $\forall$ es similar a la β-conversión, con la diferencia de que aplicada con tipos. La relación de reducción se define entonces como sigue:

$$(\lambda x.r)s \to_{\beta_\lambda} [x := s]r \qquad (\Lambda X.r)[A] \to_{\beta_\Lambda} [X := A]r$$

3.3.5 Ejemplos

Podremos entonces definir, por ejemplo, la función *id* que funciona para cualquier tipo que se le indique

$$\Lambda X.\lambda y^X.y$$

cuya derivación de tipo es

$$\dfrac{\dfrac{\dfrac{}{y : X \vdash y : X}\,(ax)}{\vdash \lambda y^X.y : X \to X}\,(\to_i)}{\vdash \Lambda X.\lambda y^X.y : \forall X.(X \to X)}\,(\forall_i) \tag{3.1}$$

Y luego aplicarla por ejemplo a $\tau \to \tau$

$$(\Lambda X.\lambda y^X.y)[\tau \to \tau](\lambda x^\tau.x)$$

obteniendo la siguiente secuencia de reducción

$$(\Lambda X.\lambda y^X.y)[\tau \to \tau](\lambda x^\tau.x) \to_{\beta_\Lambda} (\lambda y^{\tau\to\tau}.y)(\lambda x^\tau.x) \to_{\beta_\lambda} \lambda x^\tau.x$$

También puede utilizarse en algún término más complejo como

$$(\lambda z^{\forall X.(X\to X)}.\lambda x^{\tau}.\lambda w^{\tau\to\tau}.\langle z[\tau]x, z[\tau\to\tau]w\rangle)(\Lambda X.\lambda y^{X}.y)$$

cuya derivación de tipo, donde abreviaremos $r = \langle z[\tau]x, z[\tau\to\tau]w\rangle$ y $I_\forall = \forall X.(X\to X)$, es

$$\dfrac{\dfrac{\dfrac{}{z : I_\forall \vdash z : I_\forall}\,(ax)}{z : I_\forall \vdash z[\tau] : \tau\to\tau}\,(\forall_e) \qquad \dfrac{}{x : \tau \vdash x : \tau}\,(ax)}{z : I_\forall, x : \tau \vdash z[\tau]x : \tau}\,(\to_e) \tag{3.2}$$

$$\dfrac{\dfrac{\dfrac{}{z : I_\forall \vdash z : I_\forall}\,(ax)}{z : I_\forall \vdash z[\tau\to\tau] : (\tau\to\tau)\to(\tau\to\tau)}\,(\forall_e) \qquad \dfrac{}{w : \tau\to\tau \vdash w : \tau\to\tau}\,(ax)}{z : I_\forall, w : \tau\to\tau \vdash z[\tau\to\tau]w : \tau\to\tau}\,(\to_e) \tag{3.3}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(3.2)}{z : I_\forall, x : \tau \vdash z[\tau]x : \tau} \qquad \dfrac{(3.3)}{z : I_\forall, w : \tau\to\tau \vdash z[\tau\to\tau]w : \tau\to\tau}}{z : I_\forall, x : \tau, w : \tau\to\tau \vdash r : \tau\times(\tau\to\tau)}\,(\times_i)}{z : I_\forall, x : \tau \vdash \lambda w^{\tau\to\tau}.r : (\tau\to\tau)\to(\tau\times(\tau\to\tau))}\,(\to_i)}{z : I_\forall \vdash \lambda x^{\tau}.\lambda w^{\tau\to\tau}.r : \tau\to(\tau\to\tau)\to(\tau\times(\tau\to\tau))}\,(\to_i)}{\vdash \lambda z^{I_\forall}.\lambda x^{\tau}.\lambda w^{\tau\to\tau}.r : (I_\forall)\to(\tau\to\tau)\to(\tau\times(\tau\to\tau))}\,(\to_i) \tag{3.4}$$

$$\dfrac{\dfrac{(3.4)}{\vdash \lambda z^{I_\forall}.\lambda x^{\tau}.\lambda w^{\tau\to\tau}.r : I_\forall\to(\tau\to\tau)\to(\tau\times(\tau\to\tau))} \qquad \dfrac{(3.1)}{\vdash \Lambda X.\lambda y^{X}.y : I_\forall}}{\vdash \lambda z^{I_\forall}.\lambda x^{\tau}.\lambda w^{\tau\to\tau}.r : (\tau\to\tau)\to(\tau\times(\tau\to\tau))}\,(\to_e)$$

Es de interés destacar que en el cálculo lambda simplemente tipado no sería posible construir un término que se comporte como este, debido a que z debería tener un tipo monomórfico, es decir podría ser o bien de tipo $A\to A$ o bien de tipo $B\to B$, no de ambos. De manera que se requerirían dos funciones *id* distintas para lograr el mismo resultado.

3.4 Extensión con pares

Las extensiones con diferentes constructores resultan transversales, ya que podemos modificar el cálculo lambda simplemente tipado extendiéndolo con constructores de a uno, e incluso mezclar cuáles vamos agregando. Por

ejemplo, un constructor que vamos a agregar de manera transversal a los dos cálculos tipados vistos, es el de los pares o producto. Este constructor, que notaremos con el símbolo $\times$, se corresponde con los pares ordenados. Para ello se requiere agregar:

- El constructor al conjunto de los tipos:

$$A := \ldots \mid A \times A$$

- Las construcciones correspondientes a formar y proyectar un par, que notaremos con $\langle , \rangle$ y π respectivamente:

$$t ::= \ldots \mid \langle t, t \rangle \mid \pi_1 t \mid \pi_2 t$$

- Las reglas de tipado correspondientes a la introducción y eliminación del constructor de tipos:

$$\frac{\Gamma \vdash r : A \qquad \Gamma \vdash s : B}{\Gamma \vdash \langle r, s \rangle : A \wedge B}(\wedge_i)$$

$$\frac{\Gamma \vdash r : A \wedge B}{\Gamma \vdash \pi_1 r : A}(\wedge_{e_1}) \qquad \frac{\Gamma \vdash r : A \wedge B}{\Gamma \vdash \pi_2 r : B}(\wedge_{e_2})$$

- Las reglas de reducción correspondientes a las reglas de eliminación del constructor de tipos:

$$\pi_1 \langle r, s \rangle \rightarrow_{\pi_1} r \qquad \pi_2 \langle r, s \rangle \rightarrow_{\pi_2} s$$

Capítulo 4

Computación y Lógica: la correspondencia Curry-Howard

4.1 Deducción Natural

4.1.1 Definición

Uno de los interrogantes de principios del siglo XX era establecer la consistencia de ciertas lógicas. En 1935, Gentzen introdujo una formulación de la lógica que llamó *Deducción Natural* [16, 17], cuya propuesta se centra en expresar reglas de inferencia que se acerquen a la forma en que efectivamente razonamos.

En Deducción Natural la implicación se indica con $\Rightarrow$, la conjunción con $\wedge$, la disyunción con $\vee$, el cuantificador universal con $\forall$ y el existencial con $\exists$. La principal novedad de Gentzen fue la definición de reglas de derivación en pares introducción-eliminación para las conectivas. Las reglas de introducción sirven para derivar una fórmula incluyendo la conectiva, estableciendo así el significado de la misma, mientras que las de eliminación sirven para derivar una fórmula quitando la conectiva, determinando qué puedo hacer a partir de la conectiva. Tomaremos las reglas correspondientes al fragmento de la lógica compuesta por las conectivas $\Rightarrow$ y $\wedge$.

Las reglas para la conectiva $\Rightarrow$ son las siguientes:

- La introducción establece que si ante la veracidad de A se puede probar

B, entonces se puede concluir $A \Rightarrow B$ sin dicha presunción.

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}(\Rightarrow_i)$$

- La eliminación establece que dada una prueba de $A \Rightarrow B$ y una prueba de A, se puede concluir B.

$$\frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}(\Rightarrow_e)$$

Las reglas para la conectiva $\wedge$ son las siguientes:

- La introducción establece que a partir de una prueba de A y de una prueba de B, se puede concluir $A \wedge B$.

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}(\wedge_i)$$

- La eliminación establece que dada una prueba de $A \wedge B$, se pueden concluir A por un lado y B por el otro.

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}(\wedge_{e1}) \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}(\wedge_{e2})$$

Además de las reglas de introducción y eliminación de conectivas, la regla axiomática permite concluir una fórmula cuando se asume verdadera.

$$\frac{}{\Gamma, A \vdash A}(ax)$$

4.1.2 Construcción de pruebas

Con las definiciones previas podemos entonces probar, por ejemplo, las siguientes fórmulas mediante la construcción de sus árboles de derivación:

$A \Rightarrow B \Rightarrow A$:

$$\dfrac{\dfrac{\dfrac{}{A, B \vdash A}\,(ax)}{A \vdash B \Rightarrow A}\,(\Rightarrow_i)}{\vdash A \Rightarrow B \Rightarrow A}\,(\Rightarrow_i)$$

$(A \wedge B) \Rightarrow (B \wedge A)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\,(ax)}{A \wedge B \vdash B}\,(\wedge_{i2}) \qquad \dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}\,(ax)}{A \wedge B \vdash A}\,(\wedge_{i1})}{A \wedge B \vdash B \wedge A}\,(\wedge_i)}{\vdash (A \wedge B) \Rightarrow (B \wedge A)}\,(\Rightarrow_i)$$

4.1.3 Simplificación de pruebas

Uno de los resultados clave de este sistema consiste en la definición de un metateorema conocido como *eliminación de corte* (*cut-elimination* en inglés) que establece que siempre que en una derivación se encuentra una introducción inmediatamente seguida de una eliminación de la misma conectiva, la derivación puede transformarse en otra equivalente sin dicha secuencia, y por lo tanto más sencilla.

Simplificación de $\Rightarrow$ La introducción de $\Rightarrow$ para $A \Rightarrow B$ requiere poseer una prueba de B a partir de la suposición de A. La eliminación de $\Rightarrow$ requiere poseer una prueba de $A \Rightarrow B$ y de A, y da como resultado una prueba de B. Introducir un $\Rightarrow$ e inmediatamente eliminarlo parte de una prueba de B a partir de A, y nos deja con una prueba de B. Dado que para obtener esta última prueba de B se requirió una prueba de A (que puede obtenerse del mismo contexto), la secuencia introducción-eliminación es redundante. La regla que formaliza esta simplificación es la siguiente:

$$\dfrac{\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\,(\Rightarrow_i) \qquad \Gamma \vdash A}{\Gamma \vdash B}\,(\Rightarrow_e) \qquad \Longrightarrow \qquad \Gamma \vdash B$$

Simplificación de $\wedge$ Introducir un $\wedge$ entre dos proposiciones A y B requiere poseer pruebas para ambos A y B. Eliminar un $\wedge$ inmediatamente luego de haberlo introducido arroja como resultado una prueba de alguno de los dos componentes de la conjunción. Es decir, introducir un $\wedge$ entre A y B y luego eliminarlo nos deja con una prueba de A o de B, según la regla aplicada. Dado que para introducir el $\wedge$ se requirieron ambas pruebas, la secuencia introducción-eliminación de $\wedge$ es redundante. Las reglas que formalizan esta simplificación son las siguientes:

$$\dfrac{\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,(\wedge_i)}{\Gamma \vdash A}\,(\wedge_{e1}) \qquad \Longrightarrow \qquad \Gamma \vdash A$$

$$\dfrac{\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,(\wedge_i)}{\Gamma \vdash B}\,(\wedge_{e2}) \qquad \Longrightarrow \qquad \Gamma \vdash B$$

Aplicando las reglas definidas podremos, por ejemplo, simplificar una prueba como se muestra en la Figura 4.1.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \Rightarrow A \vdash A \Rightarrow A}\,(ax)}{\vdash (A \Rightarrow A) \Rightarrow (A \Rightarrow A)}\,(\Rightarrow_i) \quad \dfrac{\dfrac{}{A \vdash A}\,(ax)}{\vdash A \Rightarrow A}\,(\Rightarrow_i)}{\vdash A \Rightarrow A}\,(\Rightarrow_e) \quad \dfrac{\dfrac{}{B \vdash B}\,(ax)}{\vdash B \Rightarrow B}\,(\Rightarrow_i)}{\vdash (A \Rightarrow A) \wedge (B \Rightarrow B)}\,(\wedge_i)}{\vdash A \Rightarrow A}\,(\wedge_{e1})$$

$$\Longrightarrow \quad \dfrac{\dfrac{\dfrac{}{A \vdash A}\,(ax)}{\vdash A \Rightarrow A}\,(\Rightarrow_i) \quad \dfrac{\dfrac{}{B \vdash B}\,(ax)}{\vdash B \Rightarrow B}\,(\Rightarrow_i)}{\vdash (A \Rightarrow A) \wedge (B \Rightarrow B)}\,(\wedge_i)}{\vdash A \Rightarrow A}\,(\wedge_{e1})$$

$$\Longrightarrow \quad \dfrac{\dfrac{}{A \vdash A}\,(ax)}{\vdash A \Rightarrow A}\,(\Rightarrow_i)$$

Figura 4.1: Ejemplo de simplificación de una prueba

$$\dfrac{\dfrac{\dfrac{}{A, B \vdash A}\,(ax)}{A \vdash B \Rightarrow A}\,(\Rightarrow_i)}{\vdash A \Rightarrow B \Rightarrow A}\,(\Rightarrow_i) \qquad \dfrac{\dfrac{\dfrac{}{x : A, y : B \vdash x : A}\,(ax)}{x : A \vdash \lambda y^B.x : B \to A}\,(\to_i)}{\vdash \lambda x^A.\lambda y^B.x : A \to B \to A}\,(\to_i)$$

Figura 4.2: Derivaciones de $\vdash A \Rightarrow B \Rightarrow A$ y de $\vdash \lambda x^A.\lambda y^B.x : A \to B \to A$

4.2 Correspondencia Curry-Howard

4.2.1 Definición

Vimos que en Deducción Natural podemos producir pruebas de proposiciones aplicando las reglas de inferencia. Así, por ejemplo, podemos obtener una prueba de la proposición $A \Rightarrow B \Rightarrow A$ denotada por el juicio $\vdash A \Rightarrow B \Rightarrow A$. De manera similar, en el cálculo lambda simplemente tipado podemos producir una derivación de que un término tiene cierto tipo aplicando las reglas de tipado, y así obtener, por ejemplo, el juicio $\vdash \lambda x^A.\lambda y^B.x : A \to B \to A$.

La proposición $A \Rightarrow B \Rightarrow A$ expresa el razonamiento de que si sabe-

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

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}(\Rightarrow_i) \qquad \frac{\Gamma, x : A \vdash r : B}{\Gamma \vdash \lambda x^A.r : A \to B}(\to_i)$$

$$\frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}(\Rightarrow_e) \qquad \frac{\Gamma \vdash \lambda x^A.r : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash (\lambda x^A.r)s : B}(\to_e)$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}(\wedge_i) \qquad \frac{\Gamma \vdash r : A \qquad \Gamma \vdash s : B}{\Gamma \vdash \langle r, s\rangle : A \times B}(\times_i)$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}(\wedge_{e1}) \qquad \frac{\Gamma \vdash r : A \times B}{\Gamma \vdash \pi_1 r : A}(\times_{e1})$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}(\wedge_{e2}) \qquad \frac{\Gamma \vdash r : A \times B}{\Gamma \vdash \pi_2 r : B}(\times_{e2})$$

(a) Deducción Natural (b) Cálculo lambda

Figura 4.3: Comparación de reglas

mos que vale A, sabemos que vale $B \Rightarrow A$; a su vez, $B \Rightarrow A$ expresa el razonamiento de que si sabemos que vale B, entonces sabemos que vale A. El tipo $A \to B \to A$ denota el conjunto de los programas que toman un valor de tipo A como parámetro y dan como resultado otro programa de tipo $B \to A$; a su vez, el tipo $B \to A$ denota el conjunto de los programas que toman un valor de tipo B como parámetro y dan como resultado un valor de tipo A. Se puede observar la similitud de ambas derivaciones en la Figura 4.2, las cuales siguen la misma estructura, difiriendo únicamente en los símbolos de conectivas y en la presencia de términos. En la Figura 4.3 se pueden apreciar las similitudes estructurales de las reglas de derivación de pruebas en Deducción Natural y de tipos en cálculo lambda. A continuación analizaremos de forma conceptual dichas similitudes:

- (ax): en Deducción Natural, la regla axiomática expresa que podemos probar A ante la suposición de A. En el cálculo lambda, expresa que podemos afirmar que x tiene tipo A, ante la suposición de que x tiene

tipo A.

- $(\Rightarrow_i)/(\rightarrow_i)$: en Deducción Natural, la regla de introducción de la implicación expresa la incorporación de la dependencia a la proposición: si a partir de la suposición de que vale A puedo probar B, entonces sin necesidad de suponer A puedo probar $A \Rightarrow B$. En el cálculo lambda, la regla de la introducción de la abstracción expresa que si un valor r tiene tipo B ante la suposición de que la variable x tiene tipo A, entonces el programa que al recibir como parámetro una variable x de tipo A da como resultado r, tiene tipo $A \rightarrow B$.

- $(\Rightarrow_e)/(\rightarrow_e)$: en Deducción Natural, la regla de eliminación de la implicación expresa una de las clásicas formas de argumentación descubiertas en la antigüedad, el *modus ponendo ponens*: cuando se tiene una prueba de $A \Rightarrow B$ y otra de A, se concluye B. En el cálculo lambda, la regla de eliminación de la abstracción (o simplemente *aplicación*) expresa que si al tener un programa que produce un valor de tipo B al recibir un argumento de tipo A, que es lo que especifica el tipo $A \rightarrow B$, entonces pasarle un argumento de tipo A efectivamente produce un valor de tipo B.

- $(\wedge_i)/(\times_i)$: en Deducción Natural, la regla de introducción de la conjunción expresa la agrupación de dos pruebas: a partir de dos pruebas, produce una nueva prueba que las contiene a ambas. En el cálculo lambda, la regla de la introducción del producto expresa también una agrupación, en este caso de valores: ante la existencia de dos valores, produce un nuevo valor que los contiene a ambos.

- $(\wedge_e)/(\times_e)$: en Deducción Natural, la regla de eliminación de la conjunción expresa la proyección de una prueba a partir de una prueba que contiene a otras dos. En el cálculo lambda, la regla de la eliminación del producto expresa la proyección de un valor, a partir de un valor que contiene a otros dos.

Esta similitud tanto estructural como semántica dio origen a la llamada correspondencia Curry-Howard [26], que establece que las proposiciones y los tipos por un lado, y las pruebas y los programas por el otro, se comportan de la misma manera, y por lo tanto podemos entenderlos como que son lo mismo. De esta forma, se sabe que toda proposición asociada a un tipo que tiene algún término que lo habita, tiene una prueba, y dicha prueba tiene la misma estructura que la derivación del tipo de cualquier término con dicho

tipo.Esto trae consecuencias por demás interesantes, como poder estudiar la computación desde la lógica y la lógica desde la computación, así como recurrir al formalismo que resulte más útil en distintas circunstancias. El ejemplo de la Figura 4.2 ilustra esta conclusión de manera sencilla.

En las Figuras 4.4 y 4.5 vemos ejemplos de mayor complejidad, en donde la utilidad de pensar a la lógica en términos de la computación se torna más evidente. El ejemplo de la Figura 4.4 se corresponde, por un lado, con el axioma $(A2)$ de la teoría L de Mendelson para la lógica proposicional [23], y por otro, con el combinador S del cálculo de combinadores SKI [6,25]. El axioma $(A2)$ expresa que a partir de una prueba de $A \Rightarrow (B \Rightarrow C)$, se puede construir una prueba de que $A \Rightarrow B$ implica $A \Rightarrow C$. El combinador S (un operador primitivo del cálculo de combinadores que, si se tradujera al cálculo lambda, no tendría variables libres) representa una substitución: aplica el primer parámetro con el tercero, y el resultado de ello con la aplicación del segundo parámetro con el tercero. En esta comparación puede visualizarse la dualidad: (1) el primer parámetro de S necesariamente tendrá el tipo del antecedente de $(A2)$, es decir $A \Rightarrow (B \Rightarrow C)$; (2) el segundo parámetro tendrá el tipo del antecedente del consecuente, es decir $A \Rightarrow B$; (3) el tercer parámetro tendrá el tipo del antecedente del consecuente del consecuente, es decir A; finalmente, de las pruebas de los términos en (1) y (3) obtenemos la prueba de $B \Rightarrow C$, de las pruebas de los términos en (2) y (3) obtenemos la prueba de B, y de estas dos pruebas obtenidas obtenemos la prueba de C. El ejemplo de la Figura 4.5 sigue el mismo formato: por un lado constituye una proposición tautológica, y por el otro una función programable en el cálculo lambda tipado. Expresa la irrelevancia del orden de las hipótesis en pruebas y de los parámetros en programas: si es posible suponer A, luego suponer B y a partir de ello probar C, entonces es posible suponer B, luego A y a partir de eso probar C; de igual modo, si tomando un parámetro de tipo A y luego uno de tipo B se puede producir un valor de tipo C, también tomando un parámetro de tipo A y luego uno de tipo B se puede producir un valor de tipo C.

4.2.2 Simplificación de pruebas y evaluación de programas

La correspondencia no termina en la equiparación de proposiciones con tipos y de pruebas con programas. Además establece una equivalencia entre la simplificación de pruebas y la evaluación de programas. Vimos en la Sección 4.1.3 que la simplificación de pruebas consiste en descartar partes redundantes, obteniendo pruebas más sencillas. La evaluación de programas, por su parte, está dada por las reglas de reescritura, que en el caso del cálculo

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma' \vdash A \Rightarrow (B \Rightarrow C)}\,(ax) \quad \dfrac{}{\Gamma' \vdash A}\,(ax)}{\Gamma' \vdash B \Rightarrow C}\,(\Rightarrow_e) \quad \dfrac{}{\Gamma' \vdash A \Rightarrow B}\,(ax) \quad \dfrac{}{\Gamma' \vdash A}\,(ax)}{\Gamma' \vdash B}\,(\Rightarrow_e)}{\Gamma' = \Gamma, A \Rightarrow (B \Rightarrow C), A \Rightarrow B, A \vdash C}\,(\Rightarrow_e)}{\Gamma, A \Rightarrow (B \Rightarrow C), A \Rightarrow B \vdash A \Rightarrow C}\,(\Rightarrow_i)}{\Gamma, A \Rightarrow (B \Rightarrow C) \vdash (A \Rightarrow B) \Rightarrow (A \Rightarrow C)}\,(\Rightarrow_i)}{\Gamma \vdash (A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))}\,(\Rightarrow_i)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma' \vdash x : A \to (B \to C)}\,(ax) \quad \dfrac{}{\Gamma' \vdash z : A}\,(ax)}{\Gamma' \vdash xz : B \to C}\,(\to_e) \quad \dfrac{}{\Gamma' \vdash y : A \to B}\,(ax) \quad \dfrac{}{\Gamma' \vdash z : A}\,(ax)}{\Gamma' \vdash yz : B}\,(\to_e)}{\Gamma' = \Gamma, x : A \to (B \to C), y : A \to B, z : A \vdash xz(yz) : C}\,(\to_e)}{\Gamma, x : A \to (B \to C), y : A \to B \vdash \lambda z^A.xz(yz) : A \to C}\,(\to_i)}{\Gamma, x : A \to (B \to C) \vdash \lambda y^{A \to B}.\lambda z^A.xz(yz) : (A \to B) \to (A \to C)}\,(\to_i)}{\Gamma \vdash \lambda x^{A \to (B \to C)}.\lambda y^{A \to B}.\lambda z^A.xz(yz) : (A \to (B \to C)) \to ((A \to B) \to (A \to C))}\,(\to_i)$$

Figura 4.4: Derivaciones de $\Gamma \vdash (A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$ y de $\Gamma \vdash \lambda x^{A \to (B \to C)}.\lambda y^{A \to B}.\lambda z^A.xz(yz) : (A \to (B \to C)) \to ((A \to B) \to (A \to C))$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma' \vdash A \Rightarrow (B \Rightarrow C)}\,(ax) \quad \dfrac{}{\Gamma' \vdash A}\,(ax)}{\Gamma' \vdash B \Rightarrow C}\,(\Rightarrow_e) \quad \dfrac{}{\Gamma' \vdash B}\,(ax)}{\Gamma' = \Gamma, A \Rightarrow (B \Rightarrow C), B, A \vdash C}\,(\Rightarrow_e)}{\Gamma, A \Rightarrow (B \Rightarrow C), B \vdash A \Rightarrow C}\,(\Rightarrow_i)}{\Gamma, A \Rightarrow (B \Rightarrow C) \vdash B \Rightarrow (A \Rightarrow C)}\,(\Rightarrow_i)}{\Gamma \vdash (A \Rightarrow (B \Rightarrow C)) \Rightarrow (B \Rightarrow (A \Rightarrow C))}\,(\Rightarrow_i)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma' \vdash x : A \to (B \to C)}\,(ax) \quad \dfrac{}{\Gamma' \vdash z : A}\,(ax)}{\Gamma' \vdash xz : B \to C}\,(\to_e) \quad \dfrac{}{\Gamma' \vdash y : B}\,(ax)}{\Gamma' = \Gamma, x : A \to (B \to C), y : B, z : A \vdash xzy : C}\,(\to_e)}{\Gamma, x : A \to (B \to C), y : B \vdash \lambda z^A.xzy : A \to C}\,(\to_i)}{\Gamma, x : A \to (B \to C) \vdash \lambda y^B.\lambda z^A.xzy : B \to (A \to C)}\,(\to_i)}{\Gamma \vdash \lambda x^{A \to (B \to C)}.\lambda y^B.\lambda z^A.xzy : (A \to (B \to C)) \to (B \to (A \to C))}\,(\to_i)$$

Figura 4.5: Derivaciones de $\Gamma \vdash (A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$ y de $\Gamma \vdash \lambda x^{A \to (B \to C)}.\lambda y^B.\lambda z^A.xzy : (A \to (B \to C)) \to (B \to (A \to C))$

lamdba con pares son la beta conversión y la proyección.

Analizaremos primero el caso de la simplificación de $\Rightarrow$ y de la beta conversión. Sea una derivación que comienza con una prueba de B a partir de suponer A, produce una prueba intermedia de $A \Rightarrow B$ y, a partir de esta y de una prueba de A, produce una prueba de B. La simplificación de $\Rightarrow$ entonces arroja una prueba de B sin la suposición de A. De igual manera, cualquier término sobre el cual se pueda aplicar la beta conversión conlleva una derivación de tipo en la cual se asumió el tipo A de una variable x para concluir que un término r tiene tipo B, introduciendo luego una abstracción que da como resultado el término $\lambda x^A.r$ con tipo $A \to B$, para finalmente aplicar este último término con otro s de tipo A, obteniendo $(\lambda x^A.r)s$ con tipo B. Debido a la propiedad de la preservación de tipos, sabemos que las reglas de reducción no modifican el tipo de los términos sobre los que se aplican. Al aplicar la beta conversión sobre $(\lambda x^A.r)s$ se obtiene $[x := s]r$, y sabemos por el lema de substitución que el tipo de ambos términos es el mismo, B en este caso, pero la derivación del segundo es más simple: ya no se requiere suponer que x es de tipo A, simplemente del propio contexto se puede derivar que s es de tipo A, y luego r es de tipo B.

En el caso de la simplificación de $\wedge$ y de la proyección (π), dada una derivación que a partir de una prueba de A y una prueba de B produce una prueba de $A \wedge B$, y luego produce una prueba de A (respectivamente B), la simplificación arroja únicamente una prueba de A (respetivamente B). Asimismo, todo término $\pi_1\langle r, s\rangle$ de tipo A (respectivamente $\pi_2\langle r, s\rangle$ de tipo B) conlleva la existencia de una derivación de tipo para $\langle r, s\rangle$, y la aplicación de la proyección conllevará el descarte de la introducción y eliminación del par, dejando solamente r de tipo A (respectivamente s de tipo B).

En las Figuras 4.6 y 4.7 se puede apreciar la similitud estructural entre las aplicaciones de la simplificación y la evaluación. La simplificación de $\Rightarrow$ y $\wedge$ y la β-conversión y la proyección, respectivamente, son equivalentes, lo que permite la simplificación de una prueba a partir de la evaluación de su programa asociado, así como la evaluación de un programa a partir de la simplificación de su prueba asociada.

4.3 Conclusiones

En este capítulo describimos Deducción Natural, un formalismo para la lógica que permite expresar pruebas de proposiciones, y mostramos cómo, a partir de la sintaxis y comportamiento de sus reglas de derivación, es equivalente al cálculo lambda. Establecimos de esta manera al cálculo lambda

$$\dfrac{\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\,(\Rightarrow_i) \qquad \Gamma \vdash A}{\Gamma \vdash B}\,(\Rightarrow_e) \qquad \Longrightarrow \qquad \Gamma \vdash B$$

$$\dfrac{\dfrac{\Gamma, x : A \vdash r : B}{\Gamma \vdash \lambda x^A.r : A \to B}\,(\to_i) \qquad \Gamma \vdash s : A}{\Gamma \vdash (\lambda x^A.r)s : B}\,(\to_e) \qquad \Longrightarrow \qquad \Gamma \vdash [x := s]r : B$$

Figura 4.6: Simplificación de $\Rightarrow$ y β-conversión

$$\dfrac{\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,(\wedge_i)}{\Gamma \vdash A}\,(\wedge_{e1}) \qquad \Longrightarrow \qquad \Gamma \vdash A$$

$$\dfrac{\dfrac{\Gamma \vdash r : A \quad \Gamma \vdash s : B}{\Gamma \vdash \langle r, s\rangle : A \wedge B}\,(\wedge_i)}{\Gamma \vdash \pi_1\langle r, s\rangle : A}\,(\wedge_{e1}) \qquad \Longrightarrow \qquad \Gamma \vdash r : A$$

$$\dfrac{\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\,(\wedge_i)}{\Gamma \vdash B}\,(\wedge_{e2}) \qquad \Longrightarrow \qquad \Gamma \vdash A$$

$$\dfrac{\dfrac{\Gamma \vdash r : A \quad \Gamma \vdash s : B}{\Gamma \vdash \langle r, s\rangle : A \wedge B}\,(\wedge_i)}{\Gamma \vdash \pi_2\langle r, s\rangle : B}\,(\wedge_{e2}) \qquad \Longrightarrow \qquad \Gamma \vdash s : B$$

Figura 4.7: Simplificación de $\wedge$ y proyección (π)

como un lenguaje de la lógica, trazando una correspondencia directa entre proposiciones y tipos, pruebas y programas, y simplificación y evaluación. Esto nos permitirá entender a Sistema I y a Sistema I Polimórfico como sistemas de pruebas y al mismo tiempo como lenguajes de programación, pudiendo movernos entre uno y otro según sea conveniente. En los capítulos que siguen, utilizaremos simbología lógica y encararemos definiciones y objetivos con el foco puesto en la lógica, valiéndonos del cálculo lambda y de la programación en general como una herramienta para describir el comportamiento de la lógica, y viceversa cuando resulte de utilidad.

Parte II

Estado del arte

Capítulo 5

Isomorfismos en la programación y en la lógica

5.1 Definición

Dos proposiciones A y B son isomorfas si existen dos pruebas p de $A \Rightarrow B$ y q de $B \Rightarrow A$ tales que la composición de las pruebas q y p constituya una prueba de $A \Rightarrow A$, y la de las pruebas p y q una prueba de $B \Rightarrow B$. La existencia de un par de pruebas p y q que cumplan con estas propiedades tiene como consecuencia la posibilidad de transformar pruebas de una de las proposiciones en pruebas de la otra manteniendo toda la información relevante, permitiendo así volver a obtener la prueba original mediante las propias p y q.

De la misma manera, dos tipos A y B son isomorfos si existen dos funciones $\Gamma \vdash f : A \to B$ y $\Gamma \vdash g : B \to A$ tales que $g \circ f$ sea extensionalmente igual a la identidad en A, y $f \circ g$ sea extensionalmente igual a la identidad en B. Este par de funciones f y g permiten transformar términos de uno de los tipos en términos del otro sin pérdida de información, ya que aplicando la composición de ambas se puede obtener el término original.

Notaremos a los isomorfismos como equivalencias y escribiremos $A \equiv B$ cuando A y B sean isomorfos.

Un ejemplo de isomorfismo es la currificación $((A \wedge B) \Rightarrow C) \equiv (A \Rightarrow (B \Rightarrow C))$. En términos lógicos, establece que toda prueba de una implicación que tenga como antecedente una conjunción, puede transformarse en una prueba de una implicación anidada donde el primer antecedente es la primera proposición de la conjunción, y el segundo antecedente es la segunda proposición de la conjunción. Este razonamiento resulta natural: si a partir

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma' \vdash x : (A \wedge B) \Rightarrow C}\,(ax) \quad \dfrac{\dfrac{}{\Gamma' \vdash y : A}\,(ax) \quad \dfrac{}{\Gamma' \vdash z : B}\,(ax)}{\Gamma' \vdash \langle y, z\rangle : A \wedge B}\,(\wedge_i)}{\Gamma' = \Gamma, x : (A \wedge B) \Rightarrow C, y : A, z : B \vdash x\langle y, z\rangle : C}\,(\Rightarrow_i)}{\Gamma, x : (A \wedge B) \Rightarrow C, y : A \vdash \lambda z^B.x\langle y, z\rangle : B \Rightarrow C}\,(\Rightarrow_i)}{\Gamma, x : (A \wedge B) \Rightarrow C \vdash \lambda y^A.\lambda z^B.x\langle y, z\rangle : A \Rightarrow (B \Rightarrow C)}\,(\Rightarrow_i)}{\Gamma \vdash \lambda x^{(A\wedge B)\Rightarrow C}.\lambda y^A.\lambda z^B.x\langle y, z\rangle : ((A \wedge B) \Rightarrow C) \Rightarrow (A \Rightarrow (B \Rightarrow C))}\,(\Rightarrow_i)$$

Figura 5.1

de la introducción de una hipótesis compuesta puedo probar algo, también podré probarlo a partir de la introducción de las hipótesis por separado.

Desde la programación, el isomorfismo de Curry nos dice que una función que tome como parámetro un par de elementos puede tomar dicho par de elementos como dos parámetros por separado. La función que transforma una función no currificada en una currificada es de intuitiva construcción

$$curry = \lambda x^{(A\wedge B)\Rightarrow C}.\lambda y^A.\lambda z^B.x\langle y, z\rangle$$

y debido a la correspondencia Curry-Howard, es en sí misma la prueba de $((A \wedge B) \Rightarrow C) \Rightarrow (A \Rightarrow (B \Rightarrow C))$. La derivación puede verse en la Figura 5.1, donde además puede notarse que "mezclamos" lógica y programación, utilizando términos lambda junto a las conectivas lógicas $\Rightarrow$ y $\wedge$ en lugar de $\rightarrow$ y $\times$. De aquí en adelante escribiremos utilizando este formato.

Continuando desde el lado de la programación, definiremos la función $uncurry : (A \Rightarrow (B \Rightarrow C)) \Rightarrow (A \wedge B) \Rightarrow C$

$$uncurry = \lambda x^{A\Rightarrow(B\Rightarrow C)}.\lambda y^{A\wedge B}.x(\pi_1 y)(\pi_2 y)$$

y probaremos que $uncurry \circ curry = id_{(A\wedge B)\Rightarrow C}$ aplicando el principio de extensionalidad[1] para una función $f : (A\wedge B) \Rightarrow C$ y un par $p = \langle u, v\rangle : A\wedge B$ cualesquiera:

$$\begin{aligned} & (uncurry \circ curry) f \langle u, v\rangle \\ = \ & \text{def. } \circ \\ & uncurry(curry f)\langle u, v\rangle \end{aligned}$$

[1]El principio de extensionalidad establece que dos funciones f y g son equivalentes si y solo si para todo x, $fx = gx$

$$
\begin{aligned}
&= \text{def. } curry \\
&\quad uncurry((\lambda x^{(A\wedge B)\Rightarrow C}.\lambda y^{A}.\lambda z^{B}.x\langle y,z\rangle)f)\langle u,v\rangle \\
&\rightarrow_{\beta} \\
&\quad uncurry(\lambda y^{A}.\lambda z^{B}.f\langle y,z\rangle)\langle u,v\rangle \\
&= \text{def. } uncurry \\
&\quad (\lambda x^{A\Rightarrow(B\Rightarrow C)}.\lambda y^{A\wedge B}.x(\pi_1 y)(\pi_2 y))(\lambda y^{A}.\lambda z^{B}.f\langle y,z\rangle)\langle u,v\rangle \\
&\rightarrow_{\beta} \\
&\quad (\lambda y^{A\wedge B}.(\lambda w^{A}.\lambda z^{B}.f\langle w,z\rangle)(\pi_1 y)(\pi_2 y))\langle u,v\rangle \\
&\rightarrow_{\beta} \\
&\quad (\lambda w^{A}.\lambda z^{B}.f\langle w,z\rangle)(\pi_1\langle u,v\rangle)(\pi_2\langle u,v\rangle) \\
&\rightarrow_{\pi} \\
&\quad (\lambda w^{A}.\lambda z^{B}.f\langle w,z\rangle)u(\pi_2\langle u,v\rangle) \\
&\rightarrow_{\pi} \\
&\quad (\lambda w^{A}.\lambda z^{B}.f\langle w,z\rangle)uv \\
&\rightarrow_{\beta} \\
&\quad (\lambda z^{B}.f\langle u,z\rangle)v \\
&\rightarrow_{\beta} \\
&\quad f\langle u,v\rangle
\end{aligned}
$$

La prueba de $curry \circ uncurry = id_{A\Rightarrow(B\Rightarrow C)}$ es análoga.

Mediante el isomorfismo de la currificación podremos entonces transformar de manera explícita funciones que toman un argumento de tipo par en funciones que toman dos argumentos separados, y viceversa. Por ejemplo, la función $const : A \Rightarrow (B \Rightarrow A)$ que toma dos argumentos y da como resultado el primero, definida como sigue

$$const = \lambda x^{A}.\lambda y^{B}.x$$

podrá transformarse en otra función que toma un argumento de tipo par y da como resultado el primero de ellos:

$$uncurry\ const\ \langle r,s\rangle \rightarrow^{*} r$$

$$A \wedge B \equiv B \wedge A \qquad (\text{COMM})$$
$$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C \qquad (\text{ASSO})$$
$$A \Rightarrow (B \wedge C) \equiv (A \Rightarrow B) \wedge (A \Rightarrow C) \qquad (\text{DIST})$$
$$(A \wedge B) \Rightarrow C \equiv A \Rightarrow B \Rightarrow C \qquad (\text{CURRY})$$

Figura 5.2: Conjunto adecuado de isomorfismos para $\Rightarrow$ y $\wedge$

5.2 Caracterización

Di Cosmo [9] caracterizó los conjuntos mínimos de isomorfismos que permiten construir todos los demás, a los que llamaremos conjuntos adecuados, en distintos sistemas. En particular, lo hizo para los dos sistemas que competen a este trabajo: el fragmento de la lógica con $\Rightarrow$ y $\wedge$, correspondiente al cálculo lambda simplemente tipado con pares, y el fragmento de la lógica con $\Rightarrow$, $\wedge$ y $\forall$, correspondiente a Sistema F con pares.

5.2.1 Cálculo lambda simplemente tipado con pares

El conjunto adecuado de isomorfismos para este sistema puede observarse en la Figura 5.2. Analizamos el significado de los tres primeros a continuación, dado que el último (la currificación) fue analizado al comienzo del capítulo.

Conmutatividad de $\wedge$ (COMM) No es relevante el orden de las pruebas en una conjunción, dado que ambas pruebas están efectivamente presentes. Una función testigo de este isomorfismo, que sirve para efectuar la transformación hacia ambos lados de la relación, es $swap_\wedge : A \wedge B \Rightarrow B \wedge A$:

$$swap_\wedge = \lambda x^{A \wedge B}.\langle \pi_2 x, \pi_1 x \rangle$$

Asociatividad de $\wedge$ (ASSO) No es relevante la forma en que se organice una anidación de conjunciones, dado que la totalidad de las pruebas se encuentran presentes. Dos funciones testigo son $assocr : ((A \wedge B) \wedge C) \Rightarrow (A \wedge (B \wedge C))$ y $assocl : (A \wedge (B \wedge C)) \Rightarrow ((A \wedge B) \wedge C)$, definidas como sigue:

$$assocr = \lambda x^{(A \wedge B) \wedge C}.\langle \pi_1 \pi_1 x, \langle \pi_2 \pi_1 x, \pi_2 x \rangle \rangle$$

$$assocl = \lambda x^{A \wedge (B \wedge C)}.\langle \langle \pi_1 x, \pi_1 \pi_1 x \rangle, \pi_2 \pi_1 x \rangle$$

Distributividad de $\Rightarrow$ con respecto a $\wedge$ (DIST) Toda prueba de una implicación donde el consecuente sea una conjunción puede expresarse como una conjunción de implicaciones donde el antecedente de ambas es el antecedente original, y el consecuente es cada prueba de la conjunción original. Dos funciones testigo son $dist_{\Rightarrow} : (A \Rightarrow (B \wedge C)) \Rightarrow ((A \Rightarrow B) \wedge (A \Rightarrow C))$ y $comb_{\Rightarrow} : ((A \Rightarrow B) \wedge (A \Rightarrow C)) \Rightarrow (A \Rightarrow (B \wedge C))$, definidas como sigue:

$$dist_{\Rightarrow} = \lambda x^{A \Rightarrow (B \wedge C)}.\langle \lambda y^{A}.\pi_1 xy, \lambda y^{A}.\pi_2 xy \rangle$$

$$comb_{\Rightarrow} = \lambda x^{(A \Rightarrow B) \wedge (A \Rightarrow C)}.\lambda y^{A}.\langle \pi_1 xy, \pi_2 xy \rangle$$

5.2.2 Sistema F con pares

El conjunto adecuado de isomorfismos para este sistema puede observarse en la Figura 5.3. Analizamos a continuación el significado de los cuatro que se agregan a los correspondientes al cálculo lambda simplemente tipado.

Conmutatividad de $\forall$ y $\Rightarrow$ (P-COMM) Si una implicación es válida para todo tipo parametrizado por una variable X, pero X no aparece en el antecedente, entonces la parametrización solo tiene incidencia en el consecuente y puede "postergarse", obteniendo una implicación en la que el consecuente es válido para todo tipo X. En términos prácticos, dado que A no tiene relación con la variable X, disponer de la prueba de A antes o después de introducir el $\forall$ es irrelevante. Dos funciones testigo de este isomorfismo son $post : \forall X.(A \Rightarrow B) \Rightarrow A \Rightarrow \forall X.B$ y $pre : (A \Rightarrow \forall X.B) \Rightarrow \forall X.(A \Rightarrow B)$:

$$post = \lambda y^{\forall X.(A \Rightarrow B)}.\lambda z^{A}.\Lambda X.y[X]z$$

$$pre = \lambda y^{A \Rightarrow \forall X.B}.\Lambda X.\lambda z^{A}.yz[X]$$

Distributividad de $\forall$ con respecto a $\wedge$ (P-DIST) Si una conjunción es válida para todo tipo parametrizado por X, entonces las dos componentes de la conjunción son también válidas para todo tipo X, y esta validez ignora el momento en que se parametriza el tipo. Dos funciones testigo son $dist_{\forall} : \forall X.(A \wedge B) \Rightarrow ((\forall X.A) \wedge (\forall X.B))$ y $comb_{\forall} : ((\forall X.A) \wedge (\forall X.B)) \Rightarrow \forall X.(A \wedge B)$, definidas como sigue:

$$dist_{\forall} = \lambda y^{\forall X.(A \wedge B)}.\langle \Lambda X.\pi_1 y[X], \Lambda X.\pi_2 y[X] \rangle$$

$$
\begin{array}{lrl}
 & A \wedge B \equiv B \wedge A & (\textsc{comm}) \\
 & A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C & (\textsc{asso}) \\
 & A \Rightarrow (B \wedge C) \equiv (A \Rightarrow B) \wedge (A \Rightarrow C) & (\textsc{dist}) \\
 & (A \wedge B) \Rightarrow C \equiv A \Rightarrow B \Rightarrow C & (\textsc{curry}) \\
\text{if } X \notin FTV(A) & \forall X.(A \Rightarrow B) \equiv A \Rightarrow \forall X.B & (\textsc{p-comm}) \\
 & \forall X.(A \wedge B) \equiv \forall X.A \wedge \forall X.B & (\textsc{p-dist}) \\
 & \forall X.A \equiv \forall Y.[X := Y]A & (\textsc{alpha}) \\
 & \forall X.\forall Y.A \equiv \forall Y.\forall X.A & (\textsc{swap})
\end{array}
$$

Figura 5.3: Conjunto adecuado de isomorfismos para $\Rightarrow$, $\wedge$ y $\forall$

$$comb_\forall = \lambda y^{(\forall X.A)\wedge(\forall X.B)}.\Lambda X.\langle \pi_1 x[X], \pi_2 x[X]\rangle$$

Renombre de variables de tipo (ALPHA) El nombre de la variable con la que se parametriza un tipo en una proposición polimórfica no es relevante. Dado que el mismo constituye solo una forma de nombrar y referenciar a dicha variable, puede cambiarse en cualquier momento, siempre que no haya conflictos de captura con el nombre nuevo. Como se explicó en los Capítulos 2 y 3, esta noción se aplica en general al cálculo lambda tanto en el nivel de los tipos como en el de los términos, bajo el nombre de α-equivalencia.

Conmutatividad de $\forall$ (SWAP) El orden en el que se parametriza una proposición polimórfica no es relevante. Si una proposición r es válida para todo par de tipos parametrizados por X e Y, entonces también será válida para todo par de tipos parametrizados por Y y X. Una función testigo, que sirve para efectuar la transformación hacia ambos lados de la relación, es $swap_\forall : \forall X.\forall Y.A \Rightarrow \forall Y.\forall X.A$:

$$swap_\forall = \lambda z^{\forall X.\forall Y.A}.\Lambda Y.\Lambda X.z[X][Y]$$

Capítulo 6

Conjunto $\mathcal{I}$: sistemas módulo isomorfismos

6.1 Motivación

6.1.1 La perspectiva de la programación

En el Capítulo 1 se planteó una problemática presentada por la rigidez provista por los sistemas de tipos: el tipo de un programa establece una *forma* concreta que puede dejar de lado aspectos interesantes del *significado* del mismo. El ejemplo utilizado fue el de dos variantes de la función *suma*, una currificada y otra sin currificar, mostrando cómo en términos conceptuales resulta irrelevante la forma en que recibimos los dos operandos, sean dos argumentos por separado o un argumento par. En el Capítulo 5 se profundizó la idea de irrelevancia de las formas, analizando los conjuntos adecuados de isomorfismos para el cálculo lambda tipado con pares, y exhibiendo funciones que efectivamente adaptan la forma de un tipo a otro.

La familia de sistemas módulo isomorfismos [1, 11–13, 27], que llamaremos Conjunto $\mathcal{I}$ y referenciaremos simplemente como $\mathcal{I}$, es un conjunto de cálculos que identifican isomorfismos, es decir, consideran que los tipos (y por lo tanto sus términos asociados) que son isomorfos son equivalentes. Esto tiene como consecuencia que los sistemas funcionan módulo significado, admitiendo combinaciones de construcciones tradicionalmente incompatibles, y permitiendo a quien programa ignorar formas y pensar directamente en los significados de los valores que manipula.

6.1.2 La perspectiva de la lógica

Habiendo introducido la correspondencia Curry-Howard, es claro que los sistemas de $\mathcal{I}$ no solo constituyen lenguajes de programación, sino también sistemas de pruebas. Como se explicó también en los Capítulo 1 y 5, que dos proposiciones A y B sean isomorfas significa que toda prueba de A puede transformarse en una prueba de B y viceversa, además de que las composiciones de las transformaciones tienen que dar como resultado pruebas de $A \Rightarrow A$ y $B \Rightarrow B$.

Siguiendo la línea del conjunto adecuado de isomorfismos definido en el capítulo anterior, tomaremos la conmutatividad de la conjunción, y veremos por qué resulta tan útil la identificación. Este isomorfismo relaciona a las proposiciones $A \wedge B$ y $B \wedge A$. Intuitivamente, resulta evidente que una prueba de $A \wedge B$ es también una prueba de $B \wedge A$. Formalmente, sin embargo, esto no es evidente y si se busca obtener una prueba de $B \wedge A$, se debe arribar a la misma mediante deducciones[1]. Pragmáticamente, resultaría más cómodo y eficiente que toda prueba de $A \wedge B$ constituya también una prueba de $B \wedge A$. Lo que propone $\mathcal{I}$ es aprovechar las consecuencias de la existencia del isomorfismo de la conmutatividad de la conjunción, y que efectivamente toda prueba de $A \wedge B$ sirva como prueba de $B \wedge A$, y viceversa. Esta idea no queda restringida a este isomorfismo, sino que cada sistema determina cuál es el conjunto de isomorfismos de interés a internalizar. De esta manera, si se busca en cualquier sistema de $\mathcal{I}$ obtener una prueba de A y se dispone de una prueba de B, y es el caso que vía los isomorfismos internalizados puede transformarse a la prueba de B en una prueba de A, entonces B sirve como prueba de A, sin necesidad de transformaciones explícitas.

6.2 Internalización de isomorfismos

La esencia de este conjunto de sistemas reside en la internalización de isomorfismos, que es lo que permite que una prueba de una proposición constituya también una prueba de otra proposición isomorfa. Esta internalización puede divirse en cuatro instancias, que detallamos a continuación.

6.2.1 Equivalencia entre tipos

El primer paso hacia la identificación de isomorfismos es tomar un conjunto axiomático de isomorfismos de base y, mediante transitividad y congruencia, definir una relación de equivalencia entre todas las proposiciones isomorfas

[1] Como se puede ver en el Capítulo 4.

que se puedan construir a partir de la base, constituyendo así el conjunto de isomorfismos de interés. A nivel de proposiciones, entonces, todas las proposiciones isomorfas según dicho conjunto serán equivalentes.

6.2.2 Reglas de tipado

Definida la relación de equivalencia entre tipos, se le debe dar un significado, ya que, hasta el momento, la misma no tiene injerencia sobre el sistema porque no especificamos qué haremos con ella.

La segunda instancia de la internalización consiste en introducir una nueva regla de tipado que notaremos ($\equiv$). Esta regla es la que cocienta al sistema de tipos módulo isomorfismos, estableciendo que si r tiene tipo A y $A \equiv B$, entonces r tiene tipo B:

$$\frac{A \equiv B \qquad \Gamma \vdash r : A}{\Gamma \vdash r : B}(\equiv)$$

La consecuencia principal de esta nueva regla es una suerte de multiplicidad de tipos de los términos: todo término r de tipo A tiene no solo el tipo A, sino además todos los tipos isomorfos a A. Es así que, por ejemplo:

- mediante el isomorfismo de la conmutatividad de la conjunción, si $\langle r, s\rangle$ tiene tipo $A \wedge B$, también tiene tipo $B \wedge A$;
- mediante el isomorfismo de la asociatividad de la conjunción, si se cumple que $\langle r, \langle s, t\rangle\rangle$ tiene tipo $A \wedge (B \wedge C)$, también tiene tipo $(A \wedge B) \wedge C$;
- mediante el isomorfismo de la conmutatividad entre la implicación y la conjunción, si $\Lambda X.r$ tiene tipo $\forall X.(A \Rightarrow B)$ y $X \notin FTV_A(A)$, entonces también tiene tipo $A \Rightarrow (\forall X.B)$.

6.2.3 Equivalencia entre términos

La introducción de una relación de equivalencia en el nivel de los tipos necesariamente deberá tener consecuencias en el nivel de los términos, ya que al cumplirse $A \equiv B$, cualquier término de tipo A puede combinarse con todos los términos que son combinables con aquellos de tipo B, dando como resultado términos que no parecen bien formados.

Por ejemplo, debido al isomorfismo (CURRY), si el término $\lambda x^{A \wedge B}.r$ tiene tipo $(A \wedge B) \Rightarrow C$, lo que permite que sea aplicado con otro término de tipo $A \wedge B$, entonces también tiene tipo $A \Rightarrow (B \Rightarrow C)$, lo que permite que sea

$$\langle r, s\rangle \rightleftarrows \langle s, r\rangle \qquad (\text{comm})$$
$$r\langle s, t\rangle \rightleftarrows rst \qquad (\text{curry})$$

Figura 6.1: Equivalencia entre términos para los isomorfismos (comm) y (curry)

aplicado con un término de tipo A. Esta última construcción $(\lambda x^{A\wedge B}.r)s$, con s de tipo A, resulta intuitivamente incorrecta y por eso llamamos a estos términos *mal formados*. Pero según lo que hemos visto, desde el punto de vista semántico tiene sentido. El problema de los términos mal formados es que son eliminaciones y entonces no tiene sentido que no reduzcan, pero en algunos casos no se les puede aplicar ninguna regla de reducción porque su sintaxis no coincide con la de ninguna regla, y en los otros casos la aplicación de las reglas de reducción darían resultados incorrectos. Para que el sistema se comporte acorde a la semántica que planteamos, es necesario que se habilite un mecanismo de transformación de términos mal formados en bien formados, y para ello deberemos introducir cambios en el lenguaje en pos de que las equivalencias a nivel de tipos se vean efectivamente reflejadas a nivel de términos.

El cambio a introducir será precisamente una relación de equivalencia entre términos que permita que aquellos resultantes de las nuevas combinaciones introducidas por las etapas previas no se queden "trabados" como mal formados ni produzcan reducciones incorrectas. Para esto, lo primero que se debe determinar es cuál o cuáles son las equivalencias entre términos que son inducidas por la relación de equivalencia entre tipos del sistema. La técnica para determinar cuáles términos son candidatos a ser equivalentes se abordará en la Sección 7.2.4.

En esta sección mostraremos dos equivalencias entre términos que pueden verse en la Figura 6.1: una que refleja la equivalencia entre tipos dada por el isomorfismo de la conmutatividad de la conjunción, y otra que hace lo propio con el isomorfismo de Curry [11]. De esta manera transformamos en bien formados términos como, por ejemplo, $(\lambda x^A.\lambda y^B.r)ts$, cuya derivación de tipo puede verse en la Figura 6.2 y cuya equivalencia con un término bien formado puede verse en la Figura 6.3. Estas dos nos servirán como base para la próxima instancia, en la que se aplicarán los cambios necesarios para que estas equivalencias no produzcan inconvenientes.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, x : A, y : B \vdash r : C}{\Gamma, x : A \vdash \lambda y^B.r : B \Rightarrow C}\ (\rightarrow_i)}{\Gamma \vdash \lambda x^A.\lambda y^B.r : A \Rightarrow (B \Rightarrow C)}\ (\rightarrow_i)}{\Gamma \vdash \lambda x^A.\lambda y^B.r : (A \times B) \Rightarrow C}\ (\equiv)}{\Gamma \vdash \lambda x^A.\lambda y^B.r : (B \times A) \Rightarrow C}\ (\equiv)}{\Gamma \vdash \lambda x^A.\lambda y^B.r : B \Rightarrow A \Rightarrow C}\ (\equiv) \qquad \Gamma \vdash t : B}{\Gamma \vdash (\lambda x^A.\lambda y^B.r)t : (B \Rightarrow C)}\ (\rightarrow_e) \qquad \Gamma \vdash s : A}{\Gamma \vdash (\lambda x^A.\lambda y^B.r)ts : C}\ (\rightarrow_e)$$

Figura 6.2: Derivación del tipo de $(\lambda x^A.\lambda y^B.r)ts$

$$\begin{aligned}
&(\lambda x^A.\lambda y^B.r)ts \\
\rightleftarrows_{(\text{CURRY})}\ &(\lambda x^A.\lambda y^B.r)\langle t, s\rangle \\
\rightleftarrows_{(\text{COMM})}\ &(\lambda x^A.\lambda y^B.r)\langle s, t\rangle \\
\rightleftarrows_{(\text{CURRY})}\ &(\lambda x^A.\lambda y^B.r)st
\end{aligned}$$

Figura 6.3: Equivalencia de $(\lambda x^A.\lambda y^B.r)ts$ con un término bien formado

6.2.4 Relación de reducción

En los sistemas basados en cálculo lambda con pares, las reglas de reducción clásicas son las siguientes:

$$(\lambda x.r)s \to_\beta [x := s]r \qquad \pi_1\langle r, s\rangle \to_{\pi_1} r \qquad \pi_2\langle r, s\rangle \to_{\pi_2} s$$

Debido a las modificaciones introducidas por las relaciones de equivalencia entre tipos y términos, estas reglas tienen los siguientes problemas.

Proyección Debido al isomorfismo de la conmutatividad de la conjunción (COMM), todo par $\langle r, s\rangle$ de tipo $A \wedge B$ tiene a su vez tipo $B \wedge A$ y es equivalente al par $\langle s, r\rangle$. La relación de equivalencia entre términos combinada con las reglas de proyección tradicionales generan problemas sobre la preservación de tipos, ya que las siguientes secuencias de reducción son válidas pero los términos resultantes pueden tener distinto tipo:

$$\langle r, s\rangle \to_{\pi_1} r$$

$$\langle r, s\rangle \rightleftarrows_{(\text{COMM})} \langle s, r\rangle \to_{\pi_1} s$$

Dado que el orden en un par ya no es relevante, sino que lo relevante es el tipo de cada término, se modifica la proyección para plasmar esta característica. En vez de indicar la posición a proyectar en un par, se indicará el tipo, quedando una única construcción sintáctica para la misma

$$\pi_A r$$

y una única regla definida como sigue:

$$\text{si } r : A \qquad \pi_A\langle r, s\rangle \hookrightarrow_\pi r$$

Esta regla es no determinista ya que si r y s tienen tipo A, entonces $\pi_A\langle r, s\rangle$ podrá reducir indistintamente a r o a s. Si pensamos a los cálculos de $\mathcal{I}$ como sistemas de pruebas, gracias a la propiedad de preservación de tipos, el no determinismo implica que algunas pruebas diferentes son identificadas, como una forma de irrelevancia de pruebas (*proof-irrelevance* en inglés). Así, como r y s son pruebas de A, no es importante cuál de las dos se toma. Por otro lado, si pensamos a los cálculos de $\mathcal{I}$ como lenguajes de programación, recuperar el determinismo se vuelve más apremiante, y esto puede lograrse mediante una codificación sencilla: si r y s tienen el mismo tipo, la proyección determinista de $\langle r, s\rangle$ en r se puede codificar como

$\pi_{B \Rightarrow A}\langle \lambda x^B.t, \lambda x^C.r\rangle s$ con $B \not\equiv C$ y s de tipo B. De esta manera el no determinismo de $\mathcal{I}$ se considera una característica deseable y no un problema (para una discusión más profunda ver [11]).

Los sistemas de $\mathcal{I}$ que incluyen dicha regla son entonces algunos de los muchos cálculos no deterministas que pueden encontrarse en la literatura [4, 5, 7, 8, 24], y el constructor de pares se puede considerar como el operador de composición paralela en un cálculo de tales características.

β-conversión El término que utilizamos de ejemplo en la instancia anterior, $(\lambda x^A.\lambda y^B.r)ts$, cuyo tipo es efectivamente C, está mal formado pero es reducible. El inconveniente es que la regla β clásica no distingue entre términos bien formados y mal formados, por lo que así como puede reducirse el término equivalente $(\lambda x^A.\lambda y^B.r)st$, también puede reducirse $(\lambda x^A.\lambda y^B.r)ts$, obteniendo resultados posiblemente de distinto tipo, y perdiendo así la propiedad de preservación de tipos.

Para evitar que se reduzcan términos mal formados, se modificará la β-conversión agregando una restricción: solo podrá aplicarse la regla entre una abstracción y un término cualquiera cuando el término tenga el mismo tipo . De esta manera la nueva regla queda definida como

$$\text{si } s : A \qquad (\lambda x^A.r)s \hookrightarrow_\beta [x := s]r$$

y la relación de reducción se considera módulo isomorfismos

$$\rightarrow \; := \rightleftarrows^* \circ \hookrightarrow \circ \rightleftarrows^*$$

de forma tal que un término mal formado siempre se verá transformado a otro bien formado antes de que se aplique la β-reducción o la proyección.

6.3 Sistema I

Establecimos entonces una forma de construir un cálculo módulo isomorfismos a partir de la definición de un esquema de cambios a introducir sobre sistemas tradicionales. A continuación veremos cómo fue la aplicación de este esquema a Sistema I [11], el primero de los sistemas de $\mathcal{I}$.

6.3.1 Definición

Sistema I está basado en el cálculo lambda simplemente tipado con pares. Su gramática para tipos es la misma, mientras que su gramática para términos

$$A \wedge B \equiv B \wedge A \qquad (\text{COMM})$$
$$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C \qquad (\text{ASSO})$$
$$A \Rightarrow (B \wedge C) \equiv (A \Rightarrow B) \wedge (A \Rightarrow C) \qquad (\text{DIST})$$
$$(A \wedge B) \Rightarrow C \equiv A \Rightarrow (B \Rightarrow C) \qquad (\text{CURRY})$$

Figura 6.4: Relación de equivalencia entre tipos de Sistema I

$$\frac{}{\Gamma, x : A \vdash x : A}(ax) \qquad \frac{A \equiv B \qquad \Gamma \vdash r : A}{\Gamma \vdash r : B}(\equiv)$$

$$\frac{\Gamma, x : A \vdash r : B}{\Gamma \vdash \lambda x^A.r : A \Rightarrow B}(\Rightarrow_i) \qquad \frac{\Gamma \vdash r : A \Rightarrow B \qquad \Gamma \vdash s : A}{\Gamma \vdash rs : B}(\Rightarrow_e)$$

$$\frac{\Gamma \vdash r : A \qquad \Gamma \vdash s : B}{\Gamma \vdash \langle r, s \rangle : A \wedge B}(\wedge_i) \qquad \frac{\Gamma \vdash r : A \wedge B}{\Gamma \vdash \pi_A r : A}(\wedge_e)$$

Figura 6.5: Reglas de tipado de Sistema I

difiere únicamente en la proyección, que se da con respecto al tipo y no a la posición, como se discutió en la sección anterior.

$$A \;:=\; \tau \mid A \Rightarrow A \mid A \wedge A$$

$$t \;:=\; x \mid \lambda x^A.t \mid tt \mid \langle t, t \rangle \mid \pi_A t$$

Equivalencia entre tipos El conjunto de interés será el de todos los isomorfismos del fragmento de la lógica correspondiente ($\Rightarrow$ y $\wedge$), por lo que se agregarán los cuatro que constituyen el conjunto adecuado para la misma, y que fueron descriptos en el Capítulo 5. Las reglas de equivalencias introducidas pueden observarse en la Figura 6.4.

Reglas de tipado Las reglas de tipado estarán dadas por las del cálculo lambda simplemente tipado con pares, con la modificación de que la eliminación de la conjunción se produce respecto al tipo y no a la posición, y la adición la regla ($\equiv$) que produce la identificación de los tipos isomorfos. El conjunto completo de reglas de tipado puede observarse en la Figura 6.5.

$$\langle r, s\rangle \rightleftarrows \langle s, r\rangle \qquad (\text{COMM})$$
$$\langle r, \langle s, t\rangle\rangle \rightleftarrows \langle\langle r, s\rangle, t\rangle \qquad (\text{ASSO})$$
$$\lambda x^A.\langle r, s\rangle \rightleftarrows \langle \lambda x^A.r, \lambda x^A.s\rangle \qquad (\text{DIST}_\lambda)$$
$$\langle r, s\rangle t \rightleftarrows \langle rt, st\rangle \qquad (\text{DIST}_{\text{app}})$$
$$r\langle s, t\rangle \rightleftarrows rst \qquad (\text{CURRY})$$

$$\frac{\lambda x^A.t \rightleftarrows \lambda x^A.r}{t \rightleftarrows r} \quad \frac{ts \rightleftarrows rs}{t \rightleftarrows r} \quad \frac{st \rightleftarrows sr}{t \rightleftarrows r}$$

$$\frac{\langle t, s\rangle \rightleftarrows \langle r, s\rangle}{t \rightleftarrows r} \quad \frac{\langle s, t\rangle \rightleftarrows \langle s, r\rangle}{t \rightleftarrows r} \quad \frac{\pi_A t \rightleftarrows \pi_A r}{t \rightleftarrows r}$$

Figura 6.6: Relación de equivalencia entre términos de Sistema I

$$\text{si } s : A \qquad (\lambda x^A.r)s \hookrightarrow_\beta [x := s]r$$

$$\text{si } r : A \qquad \pi_A\langle r, s\rangle \hookrightarrow_\pi r$$

$$\frac{\lambda x^A.t \hookrightarrow \lambda x^A.r}{t \hookrightarrow r} \quad \frac{ts \hookrightarrow rs}{t \hookrightarrow r} \quad \frac{st \hookrightarrow sr}{t \hookrightarrow r}$$

$$\frac{\langle t, s\rangle \hookrightarrow \langle r, s\rangle}{t \hookrightarrow r} \quad \frac{\langle s, t\rangle \hookrightarrow \langle s, r\rangle}{t \hookrightarrow r} \quad \frac{\pi_A t \hookrightarrow \pi_A r}{t \hookrightarrow r}$$

Figura 6.7: Relación de reducción de Sistema I

Equivalencia entre términos La relación de equivalencia en el nivel de los términos está sujeta a decisiones de diseño, ya que cada isomorfismo entre tipos puede generar varias equivalencias entre términos. En la primera versión de Sistema I [10], esta relación incluía nueve equivalencias, mientras que en su versión más reciente [11] incorpora solo cinco. Un motivo para quitar algunas de ellas fue el hallazgo de que incluirlas a todas hacía perder la propiedad de normalización. Además de las reglas mencionadas, se agregan las de congruencia. En la Figura 6.6 puede observarse esta relación en su forma corregida [11].

Relación de reducción La reducción en Sistema I está dada por las reglas descriptas anteriormente, más las reglas de congruencia correspondientes. Puede visualizarse en la Figura 6.7

donde

$$\to \quad := \quad \rightleftarrows^* \circ \hookrightarrow \circ \rightleftarrows^*$$

6.3.2 Ejemplos

Como primer ejemplo retomaremos la equivalencia de la Figura 6.3, que involucra a los isomorfismos (COMM) y (CURRY), a partir de la cual queremos remarcar la posibilidad de aplicar una función de varias maneras. Dada la existencia de dicha equivalencia, que involucra a un término bien formado, las cuatro instancias son aplicaciones válidas para funciones de dos argumentos:

- Aplicación clásica: $(\lambda x^A.\lambda y^B.r)st$
- Pasaje de argumentos en forma de par: $(\lambda x^A.\lambda y^B.r)\langle s,t\rangle$
- Pasaje de argumentos en orden invertido: $(\lambda x^A.\lambda y^B.r)ts$
- Pasaje de argumentos en forma de par, en orden invertido: $(\lambda x^A.\lambda y^B.r)\langle t,s\rangle$

El mismo razonamiento podría aplicarse si la función en cuestión fuera $\lambda x^{A\wedge B}.r$ en vez de $\lambda x^A.\lambda y^B.r$, con la salvedad de que la aplicación clásica consistiría en el pasaje de argumentos en forma de par.

Como segundo ejemplo utilizaremos el siguiente término, que involucra al isomorfismo (DIST) y en el que se proyecta una función que devuelve pares:

$$\pi_{(\tau\Rightarrow\tau)\Rightarrow\tau}(\lambda x^{\tau\Rightarrow\tau}.\langle t,r\rangle)$$

Su derivación de tipo es

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma\vdash t:\tau \qquad \Gamma\vdash r:\tau\Rightarrow\tau}{\Gamma\vdash\langle t,r\rangle:\tau\wedge(\tau\Rightarrow\tau))}\,(\wedge_i)}{\Gamma\vdash\lambda x^{\tau\Rightarrow\tau}.\langle t,r\rangle:(\tau\Rightarrow\tau)\Rightarrow(\tau\wedge(\tau\Rightarrow\tau))}\,(\Rightarrow_i)}{\Gamma\vdash\lambda x^{\tau\Rightarrow\tau}.\langle t,r\rangle:((\tau\Rightarrow\tau)\Rightarrow\tau)\wedge((\tau\Rightarrow\tau)\Rightarrow(\tau\Rightarrow\tau))}\,(\equiv)}{\Gamma\vdash\pi_{(\tau\Rightarrow\tau)\Rightarrow\tau}(\lambda x^{\tau\Rightarrow\tau}.\langle t,r\rangle):(\tau\Rightarrow\tau)\Rightarrow\tau}\,(\wedge_e)$$

La reducción se produce como sigue:

$$\begin{array}{l}\pi_{(\tau\Rightarrow\tau)\Rightarrow\tau}(\lambda x^{\tau\Rightarrow\tau}.\langle t,r\rangle)\\ \rightleftarrows_{(\mathrm{DIST}_\lambda)}\\ \pi_{(\tau\Rightarrow\tau)\Rightarrow\tau}\langle\lambda x^{\tau\Rightarrow\tau}.t,\lambda x^{\tau\Rightarrow\tau}.r\rangle\end{array}$$

$$\hookrightarrow_{\pi}$$
$$\lambda x^{\tau \Rightarrow \tau}.t$$

Observamos que la función es proyectada a pesar de no estar aplicada, devolviendo una función y permitiendo la optimización del programa al descartar un subtérmino antes de que el mismo sea evaluado.

Parte III

Nuestro aporte: Sistema I Polimórfico

Capítulo 7

Sistema I Polimórfico

7.1 Introducción

Sistema I Polimórfico es el segundo sistema de $\mathcal{I}$. Consiste en una extensión de Sistema I que agrega polimorfismo, es decir la conectiva $\forall$ al fragmento de la lógica de base. Este es el principal aporte de esta tesina.

7.2 Definición

Sistema I Polimórfico, que abreviaremos está basado en dos sistemas. A la gramática de tipos y términos definida en Sistema I, le agrega la gramática de tipos y términos correspondientes al polimorfismo propia de Sistema F. Sus tipos y términos son entonces los siguientes:

$$A \quad := \quad X \mid A \Rightarrow A \mid A \wedge A \mid \forall X.A$$

$$t \quad := \quad x \mid \lambda x^A.t \mid tt \mid \langle t,t \rangle \mid \pi_A t \mid \Lambda X.t \mid t[A]$$

Siendo parte de $\mathcal{I}$, sigue el esquema definido en el Capítulo 6.

7.2.1 Funciones relevantes

Modificamos a continuación algunas de las funciones definidas en la Sección 3.3.2 para adaptarlas a SIP, y agregaremos otras nuevas.

Variables de término libres en términos La función FV_t toma un término y denota el conjunto de todas aquellas variables de término que se encuentran libres en dicho término. Se abrevia FV_t por la traducción al inglés de "variables libres en términos": *"free variables in terms"*. Se define como sigue:

$$
\begin{aligned}
&FV_t(x^A) = \{x^A\} \\
&FV_t(\lambda x^A.r) = FV_t(r) - \{x^A\} \\
&FV_t(rs) = FV_t(r) \cup FV_t(s) \\
&FV_t(\langle r, s\rangle) = FV_t(r) \cup FV_t(s) \\
&FV_t(\pi_A r) = FV_t(r) \\
&FV_t(\Lambda X.r) = FV_t(r) \\
&FV_t(r[A]) = FV_t(r)
\end{aligned}
$$

Variables de tipo libres en tipos La función FTV_A toma un tipo y denota el conjunto de las variables de tipo (es decir $X, Y, Z, \dots$) libres en dicho tipo. Su nombre está dado por *"free type variables in types"*, inglés de "variables de tipo libres en tipos". Se define como sigue:

$$
\begin{aligned}
&FTV_A(X) = \{X\} \\
&FTV_A(A \Rightarrow B) = FTV_A(A) \cup FTV_A(B) \\
&FTV_A(A \wedge B) = FTV_A(A) \cup FTV_A(B) \\
&FTV_A(\forall X.A) = FTV_A(A) - \{X\}
\end{aligned}
$$

Substitución de variables de término La función $[x := t]r$ toma una variable de término x, un término t y un término r y denota el término r en el que se substituyó cada ocurrencia libre de x por t. Se define por inducción sobre el término r.

$$
\begin{aligned}
&[x := t](x) = t \\
&[x := t](y) = y \\
&[x := t](\lambda x^A.r) = \lambda x^A.r \\
&[x := t](\lambda y^A.r) = \lambda y^A.[x := t]r && \text{si } y \notin FV_t(t) \\
&[x := t](\lambda y^A.r) = [x := t](\lambda z.[y := z]r) && \text{si } y \in FV_t(t) \\
&[x := t](\langle r, s\rangle) = \langle [x := t]r, [x := t]s\rangle \\
&[x := t](\pi_A r) = \pi_A([x := t]r)
\end{aligned}
$$

$$[x := t](\Lambda X.r) = \Lambda X.[x := t]r$$
$$[x := t](r[A]) = ([x := t]r)[A]$$

Substitución de variables de tipo en tipos La función $[X := A]B$ toma una variable de tipo X, un tipo A y un tipo B y denota el tipo B en el que se substituyó cada ocurrencia libre de X por A. Se define por inducción sobre el tipo B.

$$[X := A](X) = A$$
$$[X := A](Y) = Y$$
$$[X := A](B \Rightarrow C) = [X := A]B \Rightarrow [X := A]C$$
$$[X := A](B \wedge C) = ([X := A]B) \wedge ([X := A]C)$$
$$[X := A](\forall X.B) = \forall X.B$$
$$[X := A](\forall Y.B) = \forall Y.[X := A]B \qquad \text{si } Y \notin FTV_A(A)$$
$$[X := A](\forall Y.B) = [X := A](\forall Z.[Y := Z]B) \qquad \text{si } Y \in FTV_A(A)$$

Substitución de variables de tipo en términos La función $[X := A]r$ toma una variable de tipo X, un tipo A y un término r y denota el término r en el que se substituyó cada ocurrencia libre de X por A. Se define por inducción sobre el término r.

$$[X := A](x^B) = x^{[X:=A]B}$$
$$[X := A](\lambda x^B.r) = \lambda x^{[X:=A]B}.[X := A]r$$
$$[X := A](\langle r, s\rangle) = \langle [X := A]r, [X := A]s\rangle$$
$$[X := A](\pi_B r) = \pi_{[X:=A]B}([X := A]r)$$
$$[X := A](\Lambda X.r) = \Lambda X.r$$
$$[X := A](\Lambda Y.r) = \Lambda Y.[X := A]r$$
$$[X := A](r[B]) = ([X := A]r)[[X := A]B]$$

Substitución de variables de tipo en contextos La función $[X := A]\Gamma$ toma una variable de tipo X, un tipo A y un contexto Γ y denota el contexto Γ en el que se substituyó cada ocurrencia libre de X por A. Se define por inducción sobre el contexto Γ.

$$[X := A](x : B) = \{x : [X := A]B\}$$
$$[X := A](\Gamma, x : B) = [X := A]\Gamma \cup \{x : [X := A]B\}$$

$$A \wedge B \equiv B \wedge A \qquad (\text{COMM})$$
$$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C \qquad (\text{ASSO})$$
$$A \Rightarrow (B \wedge C) \equiv (A \Rightarrow B) \wedge (A \Rightarrow C) \qquad (\text{DIST})$$
$$(A \wedge B) \Rightarrow C \equiv A \Rightarrow (B \Rightarrow C) \qquad (\text{CURRY})$$

(a) Heredadas de Sistema I

$$\text{si } X \notin FTV_A(A) \qquad \forall X.(A \Rightarrow B) \equiv A \Rightarrow \forall X.B \qquad (\text{P-COMM})$$
$$\forall X.(A \wedge B) \equiv \forall X.A \wedge \forall X.B \qquad (\text{P-DIST})$$

(b) Incorporadas por SIP

Figura 7.1: Relación de equivalencia entre tipos de SIP

7.2.2 Equivalencia entre tipos

Los isomorfismos de caracter axiomático que se considerarán en este sistema serán los de la Figura 7.1. Se puede apreciar que no consisten en la totalidad de los isomorfismos presentes en la caracterización de Di Cosmo sobre Sistema F con pares, la cual analizamos en el Capítulo 5. Se descartan los isomorfismos correspondientes a la conmutatividad del cuantificador universal y a la α-equivalencia con respecto a las variables de tipo. Revisamos a continuación los motivos de la no incorporación de todos los isomorfismos del fragmento, así como el conjunto de isomorfismos de interés finalmente constituido.

Isomorfismos del fragmento no incluidos Los dos isomorfismos que no se tuvieron en cuenta son los siguientes:

$$\forall X.A \equiv \forall Y.[X := Y]A \qquad (\text{ALPHA})$$
$$\forall X.\forall Y.A \equiv \forall Y.\forall X.A \qquad (\text{SWAP})$$

Por un lado, la incorporación de (ALPHA) no aporta nada nuevo al sistema, ya que el mismo funciona módulo α-equivalencia. De esta manera, la equivalencia es de hecho parte del sistema, simplemente no en el formato de isomorfismo.

Por el otro, (SWAP) sí agranda el conjunto de interés, pero no queda claro que sean suficientes los beneficios que aporta por sobre el costo de introducirlo. Este isomorfismo es el análogo a $A \Rightarrow (B \Rightarrow C) \equiv B \Rightarrow$

$(A \Rightarrow C)$, consecuencia de los isomorfismos (CURRY) y (COMM), en primer orden. En dicho nivel, el isomorfismo induce la equivalencia entre términos $(\lambda x^A.\lambda y^B.r)st \rightleftarrows^* (\lambda x^A.\lambda y^B.r)ts$. Un enfoque alternativo podría ser considerar la equivalencia $\lambda x^A.\lambda y^B.r \rightleftarrows \lambda y^B.\lambda x^A.r$, donde, como se explicó en el Capítulo 6, la β-conversión solo podrá aplicarse cuando los tipos de la variable en la abstracción y del argumento coincidan. Esta idea no es fácilmente transferible a segundo orden, ya que en la β_Λ-conversión el argumento no es un término tipado sino un tipo en sí mismo. Dado que Sistema F, a diferencia de la teoría de tipos de Martin-Löf [21,22], admite un único *kind*[1] para los tipos, no es posible restringir la regla con ese método y todas las combinaciones de abstracciones con argumentos serían aceptables, haciendo que la modificación de la sintaxis y la adición de la regla no modifiquen en absoluto la semántica del lenguaje.

Otra solución posible es inspirarse en la implementada por el cálculo lambda selectivo [15], que solo incluye $\Rightarrow$ y no $\wedge$. Allí el único isomorfismo involucrado es $A \Rightarrow B \Rightarrow C \equiv B \Rightarrow A \Rightarrow C$, y utiliza etiquetas en todos los argumentos de las aplicaciones de tipo para indicar a cuál variable corresponde cada uno. Siguiendo esta propuesta, podríamos agregar las siguientes reglas:

$$r[A_X][B_Y] \rightleftarrows r[B_Y][A_X]$$

$$(\Lambda X.r)[A_X] \hookrightarrow_{\beta_\Lambda} [X := A]r$$

donde los tipos son etiquetados con la variable que deben substituir.

Este enfoque fue el elegido para la primera versión del sistema [27]. Si bien la solución es correcta, no tiene consecuencias interesantes en el lenguaje, a la vez que agrega complejidad a la sintaxis y lo hace menos legible. Por este motivo decidimos no incorporar el isomorfismo (ALPHA).

Conjunto de isomorfismos de interés El conjunto entonces está dado por los isomorfismos del framento de la lógica $\Rightarrow$, $\wedge$ y $\forall$, a excepción de aquellos para los cuales no se divisaba una relación costo-beneficio aceptable, en particular (SWAP).

7.2.3 Reglas de tipado

Las reglas de tipado de este cálculo surgen de tomar las reglas de Sistema I y agregar las correspondientes a polimorfismo de Sistema F ($\forall_i$ y $\forall_e$). Se pueden observar en la Figura 7.2.

[1] Los *kinds* son clasificaciones de tipos.

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

$$\frac{\Gamma, x : A \vdash r : B}{\Gamma \vdash \lambda x^A.r : A \Rightarrow B}(\Rightarrow_i) \qquad \frac{\Gamma \vdash r : A \Rightarrow B \qquad \Gamma \vdash s : A}{\Gamma \vdash rs : B}(\Rightarrow_e)$$

$$\frac{\Gamma \vdash r : A \qquad \Gamma \vdash s : B}{\Gamma \vdash \langle r, s \rangle : A \wedge B}(\wedge_i) \qquad \frac{\Gamma \vdash r : A \wedge B}{\Gamma \vdash \pi_A(r) : A}(\wedge_e)$$

$$\frac{\Gamma \vdash r : A \qquad X \notin FTV_c(\Gamma)}{\Gamma \vdash \Lambda X.r : \forall X.A}(\forall_i) \qquad \frac{\Gamma \vdash r : \forall X.A}{\Gamma \vdash r[B] : [X := B]A}(\forall_e)$$

Figura 7.2: Reglas de tipado de SIP

7.2.4 Equivalencia entre términos

Las equivalencias a nivel de términos inducidas por las equivalencias a nivel de tipos, junto con las reglas de congruencia correspondientes, pueden visualizarse en la Figura 7.3. Dado un isomorfismo entre tipos con dos constructores involucrados, la manera en que se determinan las posibles equivalencias inducidas por el mismo es mediante la construcción de términos aplicando de diferentes maneras las reglas de tipado relacionadas con los constructores. Para cualquier par de conectivas, tenemos cuatro posibles formas de combinar sus reglas de tipado: introducción-introducción, introducción-eliminación, eliminación-introducción, y eliminación-eliminación. Cada una de ellas dará origen a una equivalencia, y cada lado de la equivalencia estará dado por el orden en que se apliquen las reglas de tipado: primero un constructor y luego el otro, y viceversa. Un detalle para notar es que las propias derivaciones de tipos involucradas establecerán ciertas restricciones sobre los términos. Si bien estas restricciones son condiciones necesarias para que las equivalencias tengan sentido, no se las incluye explícitamente en cada relación porque, de no cumplirse, los términos no podrían tiparse y por lo tanto no serían válidos. Analizaremos a continuación las equivalencias heredadas de Sistema I y las equivalencias derivadas de los dos isomorfismos relacionados al constructor $\forall$ internalizados en SIP. En el caso de las heredadas solo veremos las combinaciones que fueron consideradas, mientras que en el caso de las nuevas revisaremos todas las combinaciones posibles.

Heredadas de Sistema I

$A \wedge B \equiv B \wedge A$ (COMM) En este isomorfismo hay solo una conectiva involucrada, $\wedge$, y se incluye la equivalencia correspondiente a su introducción: $\langle r, s\rangle \rightleftarrows \langle s, r\rangle$ (COMM)

$$\dfrac{\Gamma \vdash r : A \qquad \Gamma \vdash s : B}{\Gamma \vdash \langle r, s\rangle : A \wedge B}\ (\wedge_i)$$

$$\dfrac{\dfrac{\Gamma \vdash s : B \qquad \Gamma \vdash r : A}{\Gamma \vdash \langle s, r\rangle : B \wedge A}\ (\wedge_i)}{\Gamma \vdash \langle s, r\rangle : A \wedge B}\ (\equiv)$$

$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$ (ASSO) En este isomorfismo hay dos conectivas involucradas, $\wedge$ y $\wedge$, y se incluye la equivalencia correspondiente a la introducción de ambas: $\langle r, \langle s, t\rangle\rangle \rightleftarrows \langle\langle r, s\rangle, t\rangle$ (ASSO)

$$\dfrac{\dfrac{\Gamma \vdash s : B \qquad \Gamma \vdash t : C}{\Gamma \vdash \langle s, t\rangle : B \wedge C}\ (\wedge_i) \qquad \Gamma \vdash r : A}{\Gamma \vdash \langle r, \langle s, t\rangle\rangle : A \wedge (B \wedge C)}\ (\wedge_i)$$

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash r : A \qquad \Gamma \vdash s : B}{\Gamma \vdash \langle r, s\rangle : A \wedge B}\ (\wedge_i) \qquad \Gamma \vdash t : C}{\Gamma \vdash \langle\langle r, s\rangle, t\rangle : (A \wedge B) \wedge C}\ (\wedge_i)}{\Gamma \vdash \langle\langle r, s\rangle, t\rangle : A \wedge (B \wedge C)}\ (\equiv)$$

$A \Rightarrow (B \wedge C) \equiv (A \Rightarrow B) \wedge (A \Rightarrow C)$ (DIST) En este isomorfismo hay dos conectivas involucradas, $\Rightarrow$ y $\wedge$, y se incluyen dos equivalencias:

- la correspondiente a la introducción de $\Rightarrow$ e introducción de $\wedge$, $\lambda x^A.\langle r, s\rangle \rightleftarrows \langle \lambda x^A.r, \lambda x^A.s\rangle$ (DIST_λ)

$$\dfrac{\dfrac{\Gamma, x : A \vdash r : B \qquad \Gamma, x : A \vdash s : C}{\Gamma, x : A \vdash \langle r, s\rangle : (B \wedge C)}\ (\wedge_i)}{\Gamma \vdash \lambda x^A.\langle r, s\rangle : A \Rightarrow (B \wedge C)}\ (\Rightarrow_i)$$

$$\dfrac{\dfrac{\dfrac{\Gamma, x : A \vdash r : B}{\Gamma \vdash \lambda x^A.r : A \Rightarrow B}\ (\Rightarrow_i) \qquad \dfrac{\Gamma, x : A \vdash s : C}{\Gamma \vdash \lambda x^A.s : A \Rightarrow C}\ (\Rightarrow_i)}{\Gamma \vdash \langle \lambda x^A.r, \lambda x^A.s\rangle : (A \Rightarrow B) \wedge (A \Rightarrow C)}\ (\wedge_i)}{\Gamma \vdash \langle \lambda x^A.r, \lambda x^A.s\rangle : A \Rightarrow (B \wedge C)}\ (\equiv)$$

- la correspondiente a la eliminación de $\Rightarrow$ e introducción de $\wedge$, $\langle r, s\rangle t \rightleftarrows \langle rt, st\rangle$ (DIST_{app})

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash r : A \Rightarrow B \quad \Gamma \vdash s : A \Rightarrow C}{\Gamma \vdash \langle r, s\rangle : (A \Rightarrow B) \wedge (A \Rightarrow C)}\ (\wedge_i)}{\Gamma \vdash \langle r, s\rangle : A \Rightarrow (B \wedge C)}\ (\equiv) \quad \Gamma \vdash t : A}{\Gamma \vdash \langle r, s\rangle t : B \wedge C}\ (\Rightarrow_e)$$

$$\dfrac{\dfrac{\Gamma \vdash r : A \Rightarrow B \quad \Gamma \vdash t : A}{\Gamma \vdash st : B}\ (\Rightarrow_e) \quad \dfrac{\Gamma \vdash s : A \Rightarrow C \quad \Gamma \vdash t : A}{\Gamma \vdash rt : C}\ (\Rightarrow_e)}{\Gamma \vdash \langle rt, st\rangle : B \wedge C}\ (\wedge_i)$$

$(A \wedge B) \Rightarrow C \equiv A \Rightarrow (B \Rightarrow C)$ (CURRY) Este isomorfismo difiere de los demás en cuanto a que no aparecen las mismas conectivas de ambos lados de la equivalencia. Se tomará como equivalencia la correspondiente a la de introducción de $\wedge$ y eliminación de $\Rightarrow$ a la izquierda, y eliminación de ambas $\Rightarrow$ a la derecha: $r\langle s, t\rangle \rightleftarrows rst$ (CURRY)

$$\dfrac{\Gamma \vdash r : (A \wedge B) \Rightarrow C \qquad \dfrac{\Gamma \vdash s : A \quad \Gamma \vdash t : B}{\Gamma \vdash \langle s, t\rangle : A \Rightarrow B}\ (\wedge_i)}{\Gamma \vdash r\langle s, t\rangle : C}\ (\Rightarrow_e)$$

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash r : (A \wedge B) \Rightarrow C}{\Gamma \vdash r : A \Rightarrow (B \Rightarrow C)}\ (\equiv) \quad \Gamma \vdash s : A}{\Gamma \vdash rs : B \Rightarrow C}\ (\Rightarrow_e) \quad \Gamma \vdash t : B}{\Gamma \vdash rst : C}\ (\Rightarrow_e)$$

Incorporadas por SIP

$\forall X.(A \Rightarrow B) \equiv A \Rightarrow \forall X.B$ (P-COMM) Este isomorfismo involucra a las conectivas $\forall$ y $\Rightarrow$. Las cuatro combinaciones de reglas son las siguientes:

- la introducción de $\forall$ y $\Rightarrow$ da lugar a la equivalencia $\Lambda X.\lambda x^A.r \rightleftarrows \lambda x^A.\Lambda X.r$ ($\text{P-COMM}_{\forall_i \Rightarrow_i}$)

$$\dfrac{\dfrac{\Gamma, x : A \vdash r : B}{\Gamma \vdash \lambda x^A.r : A \Rightarrow B}\ (\Rightarrow_i)}{\Gamma \vdash \Lambda X.\lambda x^A.r : \forall X.(A \Rightarrow B)}\ (\forall_i)$$

$$\dfrac{\dfrac{\Gamma, x : A \vdash r : B}{\Gamma, x : A \vdash \Lambda X.r : \forall X.B}\ (\forall_i)}{\Gamma \vdash \lambda x^A.\Lambda X.r : A \Rightarrow \forall X.B}\ (\Rightarrow_i)$$

- la introducción de $\forall$ y eliminación de $\Rightarrow$ da lugar a la equivalencia $(\Lambda X.r)s \rightleftarrows \Lambda X.rs$ que no ha sido considerada por ser redudante con la anterior

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash r : A \Rightarrow B}{\Gamma \vdash \Lambda X.r : \forall X.(A \Rightarrow B)}\ (\forall_i)}{\Gamma \vdash \Lambda X.r : A \Rightarrow \forall X.B}\ (\equiv) \qquad \Gamma \vdash s : A}{\Gamma \vdash (\Lambda X.r)s : \forall X.B}\ (\Rightarrow_e)$$

$$\dfrac{\Gamma \vdash r : A \Rightarrow B \qquad \Gamma \vdash s : A}{\dfrac{\Gamma \vdash rs : B}{\Gamma \vdash \Lambda X.rs : \forall X.B}\ (\forall_i)}\ (\Rightarrow_e)$$

- la eliminación de $\forall$ e introducción de $\Rightarrow$ da lugar a la equivalencia $(\lambda x^A.r)[B] \rightleftarrows \lambda x^A.r[B]$ ($\text{P-COMM}_{\forall_e \Rightarrow_i}$)

$$\dfrac{\dfrac{\dfrac{\Gamma, x : A \vdash r : \forall X.C}{\Gamma \vdash \lambda x^A.r : A \Rightarrow \forall X.C}\ (\Rightarrow_i)}{\Gamma \vdash \lambda x^A.r : \forall X.(A \Rightarrow C)}\ (\equiv)}{\Gamma \vdash (\lambda x^A.r)[B] : [X := B](A \Rightarrow C)}\ (\forall_e)$$

$$\dfrac{\dfrac{\Gamma, x : A \vdash r : \forall X.C}{\Gamma, x : A \vdash r[B] : [X := B]C}\ (\forall_e)}{\Gamma \vdash \lambda x^A.r[B] : A \Rightarrow [X := B]C}\ (\Rightarrow_i)$$

- la eliminación de $\forall$ y $\Rightarrow$ da lugar a la equivalencia $r[A]s \rightleftarrows rs[A]$ que no ha sido considerada por ser redundante con la anterior

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash r : \forall X.(A \Rightarrow B)}{\Gamma \vdash r[C] : [X := C](A \Rightarrow B)}\ (\forall_e)}{\Gamma \vdash r[C] : A \Rightarrow [X := C]B}\ (\equiv) \qquad \Gamma \vdash s : A}{\Gamma \vdash r[C]s : [X := C]B}\ (\Rightarrow_e)$$

$$\dfrac{\dfrac{\Gamma \vdash r : \forall X.(A \Rightarrow B)}{\Gamma \vdash r : A \Rightarrow \forall X.B}\ (\equiv) \qquad \Gamma \vdash s : A}{\dfrac{\Gamma \vdash rs : \forall X.B}{\Gamma \vdash rs[C] : [X := C]B}\ (\forall_e)}\ (\Rightarrow_e)$$

$\forall X.(A \wedge B) \equiv \forall X.A \wedge \forall X.B$ (P-DIST) Este isomorfismo involucra a las conectivas $\forall$ y $\wedge$. Las cuatro combinaciones de reglas son las siguientes:

- la introducción de $\forall$ y $\wedge$ da lugar a la equivalencia $\Lambda X.\langle r, s\rangle \rightleftarrows \langle \Lambda X.r, \Lambda X.s\rangle$ ($\text{P-DIST}_{\forall_i \wedge_i}$)

$$\dfrac{\dfrac{\Gamma \vdash r : A \quad \Gamma \vdash s : B}{\Gamma \vdash \langle r, s\rangle : A \wedge B}\ (\wedge_i)}{\Gamma \vdash \Lambda X.\langle r, s\rangle : \forall X.A \wedge B}\ (\forall_i)$$

$$\dfrac{\dfrac{\Gamma \vdash r : A}{\Gamma \vdash \Lambda X.r : \forall X.A}\ (\forall_i) \quad \dfrac{\Gamma \vdash s : B}{\Gamma \vdash \Lambda X.s : \forall X.B}\ (\forall_i)}{\Gamma \vdash \langle \Lambda X.r, \Lambda X.s\rangle : \forall X.A \wedge \forall X.B}\ (\wedge_i)$$

- la introducción de $\forall$ y eliminación de $\wedge$ da lugar a la equivalencia $\pi_{\forall X.A}(\Lambda X.r) \rightleftarrows \Lambda X.\pi_A r$ ($\text{P-DIST}_{\forall_i \wedge_e}$)

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash r : A \wedge B}{\Gamma \vdash \Lambda X.r : \forall X.(A \wedge B)}\ (\forall_i)}{\Gamma \vdash \Lambda X.r : \forall X.A \wedge \forall X.B}\ (\equiv)}{\Gamma \vdash \pi_{\forall X.A}(\Lambda X.r) : \forall X.A}\ (\wedge_e)$$

$$\dfrac{\dfrac{\Gamma \vdash r : A \wedge B}{\Gamma \vdash \pi_A r : A}\ (\wedge_e)}{\Gamma \vdash \Lambda X.\pi_A r : \forall X.A}\ (\forall_i)$$

- la eliminación de $\forall$ e introducción de $\wedge$ da lugar a la equivalencia $\langle r, s\rangle[\mathrm{A}] \rightleftarrows \langle r[A], s[A]\rangle$ ($\text{P-DIST}_{\forall_e \wedge_i}$)

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash r : \forall X.B \quad \Gamma \vdash s : \forall X.C}{\Gamma \vdash \langle r, s\rangle : \forall X.B \wedge \forall X.C}\ (\wedge_i)}{\Gamma \vdash \langle r, s\rangle : \forall X.(B \wedge C)}\ (\equiv)}{\Gamma \vdash \langle r, s\rangle[A] : [X := A](B \wedge C)}\ (\forall_e)$$

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash r : \forall X.B}{\Gamma \vdash r[A] : [X := A]B}\ (\forall_e) \quad \dfrac{\Gamma \vdash s : \forall X.C}{\Gamma \vdash s[A] : [X := A]C}\ (\forall_e)}{\Gamma \vdash \langle r[A], s[A]\rangle : [X := A]B \wedge [X := A]C}\ (\wedge_i)}{\Gamma \vdash \langle r[A], s[A]\rangle : [X := A](B \wedge C)}\ (\equiv)$$

- la eliminación de $\forall$ y $\wedge$ da lugar a la equivalencia que establece que si $r : \forall X.(B \wedge C)$, entonces $(\pi_{\forall X.B} r)[A] \rightleftarrows \pi_{[X:=A]B}(r[A])$ ($\text{P-DIST}_{\forall_e \wedge_e}$). En este caso es necesario asumir $\Gamma \vdash r : \forall X.(B \wedge C)$, ya que el término del lado izquierdo no fuerza dicho tipo para r, mientras que el lado derecho lo necesita.

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash r : \forall X.(B \wedge C)}{\Gamma \vdash r : \forall X.B \wedge \forall X.C}\,(\equiv)}{\Gamma \vdash \pi_{\forall X.B} r : \forall X.B}\,(\wedge_e)}{\Gamma \vdash (\pi_{\forall X.B} r)[A] : [X := A]B}\,(\forall_e)$$

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash r : \forall X.(B \wedge C)}{\Gamma \vdash r[A] : [X := A](B \wedge C)}\,(\forall_e)}{\Gamma \vdash r[A] : [X := A]B \wedge [X := A]C}\,(\equiv)}{\Gamma \vdash \pi_{[X:=A]B}(r[A]) : [X := A]B}\,(\wedge_e)$$

7.2.5 Relación de reducción

Las reglas de reducción de este sistema consisten en las de Sistema I con la adición de la beta reducción de tipos, y las reglas de congruencia correspondientes. Quedan conformadas como se indica en la Figura 7.4. Luego la relación de reducción está definida como $\rightarrow \;:=\; \rightleftarrows^* \circ \hookrightarrow \circ \rightleftarrows^*$.

7.3 Ejemplos

Veremos algunos ejemplos de términos que no tienen tipo en cálculos tradicionales y que es factible construir, y utilizar sin inconvenientes, en este sistema.

Ejemplo 1 Debido al isomorfismo (P-COMM) podremos construir el siguiente término, en el que efectuamos una aplicación de tipos sobre un término que no tiene forma de abstracción de tipos, sino de términos:

$$(\lambda x^{\forall X.(X \Rightarrow X)}.x)[\tau](\Lambda X.\lambda x^X.x)$$

Su derivación de tipo, donde consideraremos $r = (\lambda x^{\forall X.(X \Rightarrow X)}.x)[\tau]$, es

$$\langle r, s\rangle \rightleftarrows \langle s, r\rangle \qquad (\text{COMM})$$
$$\langle r, \langle s, t\rangle\rangle \rightleftarrows \langle\langle r, s\rangle, t\rangle \qquad (\text{ASSO})$$
$$\lambda x^A.\langle r, s\rangle \rightleftarrows \langle \lambda x^A.r, \lambda x^A.s\rangle \qquad (\text{DIST}_\lambda)$$
$$\langle r, s\rangle t \rightleftarrows \langle rt, st\rangle \qquad (\text{DIST}_{\text{app}})$$
$$r\langle s, t\rangle \rightleftarrows rst \qquad (\text{CURRY})$$

$$\frac{\lambda x^A.t \rightleftarrows \lambda x^A.r}{t \rightleftarrows r} \quad \frac{ts \rightleftarrows rs}{t \rightleftarrows r} \quad \frac{st \rightleftarrows sr}{t \rightleftarrows r}$$
$$\frac{\langle t, s\rangle \rightleftarrows \langle r, s\rangle}{t \rightleftarrows r} \quad \frac{\langle s, t\rangle \rightleftarrows \langle s, r\rangle}{t \rightleftarrows r} \quad \frac{\pi_A t \rightleftarrows \pi_A r}{t \rightleftarrows r}$$

(a) Heredadas de Sistema I

$$\text{si } X \notin FTV_A(A) \qquad \Lambda X.\lambda x^A.r \rightleftarrows \lambda x^A.\Lambda X.r \qquad (\text{P-COMM}_{\forall_i \Rightarrow_i})$$
$$\text{si } X \notin FTV_A(A) \qquad (\lambda x^A.r)[B] \rightleftarrows \lambda x^A.r[B] \qquad (\text{P-COMM}_{\forall_e \Rightarrow_i})$$
$$\Lambda X.\langle r, s\rangle \rightleftarrows \langle \Lambda X.r, \Lambda X.s\rangle \qquad (\text{P-DIST}_{\forall_i \wedge_i})$$
$$\pi_{\forall X.A}(\Lambda X.r) \rightleftarrows \Lambda X.\pi_A r \qquad (\text{P-DIST}_{\forall_i \wedge_e})$$
$$\langle r, s\rangle[A] \rightleftarrows \langle r[A], s[A]\rangle \qquad (\text{P-DIST}_{\forall_e \wedge_i})$$
$$\text{si } r : \forall X.(B \wedge C) \qquad (\pi_{\forall X.B} r)[A] \rightleftarrows \pi_{[X:=A]B}(r[A]) \qquad (\text{P-DIST}_{\forall_e \wedge_e})$$

$$\frac{\Lambda X.t \rightleftarrows \Lambda X.r}{t \rightleftarrows r} \qquad \frac{t[A] \rightleftarrows r[A]}{t \rightleftarrows r}$$

(b) Incorporadas por SIP

Figura 7.3: Relación de equivalencia entre términos de SIP

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : \forall X.(X \Rightarrow X) \vdash x : \forall X.(X \Rightarrow X)}\,(ax)}{\vdash \lambda x^{\forall X.(X \Rightarrow X)}.x : \forall X.(X \Rightarrow X) \Rightarrow \forall X.(X \Rightarrow X)}\,(\Rightarrow_i)}{\vdash \lambda x^{\forall X.(X \Rightarrow X)}.x : \forall X.(\forall X.(X \Rightarrow X)) \Rightarrow (X \Rightarrow X)}\,(\equiv)}{\vdash (\lambda x^{\forall X.(X \Rightarrow X)}.x)[\tau] : \forall X.(X \Rightarrow X) \Rightarrow (\tau \Rightarrow \tau)}\,(\forall_e) \qquad (7.1)$$

$$\dfrac{\dfrac{(7.1)}{\vdash r : \forall X.(X \Rightarrow X) \Rightarrow (\tau \Rightarrow \tau)}\,(\forall_e) \qquad \vdash \Lambda X.\lambda x^X.x : \forall X.(X \Rightarrow X)}{\vdash r(\Lambda X.\lambda x^X.x) : \tau \Rightarrow \tau}\,(\Rightarrow_e)$$

y su reducción

$$\text{si } s : A \qquad (\lambda x^A.r)s \hookrightarrow_{\beta_\lambda} [x := s]r$$

$$\text{si } r : A \qquad \pi_A\langle r, s\rangle \hookrightarrow_{\pi} r$$

$$(\Lambda X.r)A \hookrightarrow_{\beta_\Lambda} [X := A]r$$

$$\frac{\lambda x^A.t \hookrightarrow \lambda x^A.r}{t \hookrightarrow r} \qquad \frac{ts \hookrightarrow rs}{t \hookrightarrow r} \qquad \frac{st \hookrightarrow sr}{t \hookrightarrow r}$$

$$\frac{\langle t, s\rangle \hookrightarrow \langle r, s\rangle}{t \hookrightarrow r} \qquad \frac{\langle s, t\rangle \hookrightarrow \langle s, r\rangle}{t \hookrightarrow r} \qquad \frac{\pi_A t \hookrightarrow \pi_A r}{t \hookrightarrow r}$$

$$\frac{\Lambda X.t \hookrightarrow \Lambda X.r}{t \hookrightarrow r} \qquad \frac{t[A] \hookrightarrow r[A]}{t \hookrightarrow r}$$

Figura 7.4: Relación de reducción de SIP

$$\begin{array}{l} \quad (\lambda x^{\forall X.(X \Rightarrow X)}.x)[\tau](\Lambda X.\lambda y^X.y) \\ \rightleftarrows_{(\text{P-COMM}_{\forall_e \Rightarrow_i})} \\ \quad (\lambda x^{\forall X.(X \Rightarrow X)}.x[\tau])(\Lambda X.\lambda y^X.y) \\ \hookrightarrow_{\beta_\lambda} \\ \quad (\Lambda X.\lambda y^X.y)[\tau] \\ \hookrightarrow_{\beta_\Lambda} \\ \quad \lambda y^A.y \end{array}$$

Ejemplo 2 En la misma línea del ejemplo anterior, podremos construir el siguiente término, que es una aplicación de términos sobre una abstracción de tipos:

$$(\Lambda X.\lambda x^\tau.\lambda y^{\tau \Rightarrow X}.yx)t$$

Su derivación de tipos es

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : \tau, y : \tau \Rightarrow X \vdash y : \tau \Rightarrow X}\,(ax) \quad \dfrac{}{x : \tau, y : \tau \Rightarrow X \vdash x : \tau}\,(ax)}{x : \tau, y : \tau \Rightarrow X \vdash yx : X}\,(\Rightarrow_e)}{x : \tau \vdash \lambda y^{\tau \Rightarrow X}.yx : (\tau \Rightarrow X) \Rightarrow X}\,(\Rightarrow_i)}{\vdash \lambda x^\tau.\lambda y^{\tau \Rightarrow X}.yx : \tau \Rightarrow (\tau \Rightarrow X) \Rightarrow X}\,(\Rightarrow_i)}{\vdash \Lambda X.\lambda x^\tau.\lambda y^{\tau \Rightarrow X}.yx) : \forall X.(\tau \Rightarrow (\tau \Rightarrow X) \Rightarrow X)}\,(\forall_i) \tag{7.2}$$

$$\dfrac{\dfrac{(7.2)}{\vdash \Lambda X.\lambda x^{\tau}.\lambda y^{\tau \Rightarrow X}.yx) : \forall X.(\tau \Rightarrow (\tau \Rightarrow X) \Rightarrow X)}\ (\forall_i) \qquad \Gamma \vdash t : \tau}{\Gamma \vdash (\Lambda X.\lambda x^{\tau}.\lambda y^{\tau \Rightarrow X}.yx)t : \forall X.((\tau \Rightarrow X) \Rightarrow X)}\ (\Rightarrow_e)$$

y su reducción

$$\begin{array}{l} \quad (\Lambda X.\lambda x^{\tau}.\lambda y^{\tau \Rightarrow X}.yx)t \\ \rightleftarrows_{(\text{P-COMM}_{\forall_i \Rightarrow_i})} \\ \quad (\lambda x^{\tau}.(\Lambda X.\lambda y^{\tau \Rightarrow X}.yx))t \\ \hookrightarrow_{\beta_\lambda} \\ \quad (\Lambda X.\lambda y^{\tau \Rightarrow X}.yt) \end{array}$$

Ejemplo 3 Debido al isomorfismo (P-DIST), podremos construir el siguiente término, en el que proyectamos una abstracción de tipos:

$$\pi_{\forall X.(X \Rightarrow X)}(\Lambda X.\langle \lambda x^X.x, t\rangle)$$

Su derivación de tipo es

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : X \vdash x : X}\ (ax)}{\vdash \lambda x^X.x : X \Rightarrow X}\ (\Rightarrow_i) \qquad \Gamma \vdash t : \tau}{\Gamma \vdash \langle \lambda x^X.x, t\rangle : (X \Rightarrow X) \wedge \tau}\ (\wedge_i)}{\Gamma \vdash \Lambda X.\langle \lambda x^X.x, t\rangle : \forall X.((X \Rightarrow X) \wedge \tau)}\ (\forall_i)}{\Gamma \vdash \Lambda X.\langle \lambda x^X.x, t\rangle : \forall X.(X \Rightarrow X) \wedge \forall X.\tau}\ (\equiv)}{\Gamma \vdash \pi_{\forall X.(X \Rightarrow X)}(\Lambda X.\langle \lambda x^X.x, t\rangle) : \forall X.(X \Rightarrow X)}\ (\wedge_e)$$

Una posible reducción es

$$\begin{array}{l} \quad \pi_{\forall X.(X \Rightarrow X)}(\Lambda X.\langle \lambda x^X.x, t\rangle) \\ \rightleftarrows_{(\text{P-DIST}_{\forall_i \wedge_i})} \\ \quad \pi_{\forall X.(X \Rightarrow X)}\langle \Lambda X.\lambda x^X.x, \Lambda X.t\rangle \\ \hookrightarrow_{\pi} \\ \quad \Lambda X.\lambda x^X.x \end{array}$$

y otra es

$$\pi_{\forall X.(X \Rightarrow X)}(\Lambda X.\langle \lambda x^X.x, t\rangle)$$

$$
\begin{array}{l}
\rightleftarrows_{(\text{P-DIST}_{\forall_i \wedge_e})} \\
\quad \Lambda X.\pi_{X \Rightarrow X}\langle \lambda x^X.x, t\rangle \\
\hookrightarrow_{\pi} \\
\quad \Lambda X.\lambda x^X.x
\end{array}
$$

Ejemplo 4 También debido al isomorfismo (P-DIST), podremos construir el siguiente término, en el que

$$\langle \Lambda X.\lambda x^X.\lambda y^A.t,\ \Lambda X.\lambda x^X.\lambda z^B.r\rangle[C]$$

Su derivación de tipos, donde asumimos $X \notin FTV_A(D) \cup FTV_A(E)$ y consideramos $u = \Lambda X.\lambda x^X.\lambda y^A.s$ y $v = \Lambda X.\lambda x^X.\lambda z^B.t$, es la siguiente:

$$
\dfrac{\dfrac{\dfrac{\Gamma, x : X, y : A \vdash s : D}{\Gamma, x : X \vdash \lambda y^A.s : A \Rightarrow D}\ (\Rightarrow_i)}{\Gamma \vdash \lambda x^X.\lambda y^A.s : X \Rightarrow A \Rightarrow D}\ (\Rightarrow_i)}{\Gamma \vdash \Lambda X.\lambda x^X.\lambda y^A.s : \forall X.(X \Rightarrow A \Rightarrow D)}\ (\forall_i) \tag{7.3}
$$

$$
\dfrac{\dfrac{\dfrac{\Gamma, x : X, z : B \vdash t : E}{\Gamma, x : X \vdash \lambda z^B.t : B \Rightarrow E}\ (\Rightarrow_i)}{\Gamma \vdash \lambda x^X.\lambda z^B.t : X \Rightarrow B \Rightarrow E}\ (\Rightarrow_i)}{\Gamma \vdash \Lambda X.\lambda x^X.\lambda z^B.t : \forall X.(X \Rightarrow B \Rightarrow E)}\ (\forall_i) \tag{7.4}
$$

$$
\dfrac{\dfrac{\dfrac{\dfrac{(7.3)}{\Gamma \vdash u : \forall X.(X \Rightarrow A \Rightarrow D)}\ (\forall_i) \quad \dfrac{(7.4)}{\Gamma \vdash v : \forall X.(X \Rightarrow B \Rightarrow E)}\ (\forall_i)}{\Gamma \vdash \langle u, v\rangle : \forall X.(X \Rightarrow A \Rightarrow D) \wedge \forall X.(X \Rightarrow B \Rightarrow E)}\ (\wedge_i)}{\Gamma \vdash \langle u, v\rangle : \forall X.(X \Rightarrow A \Rightarrow D \wedge X \Rightarrow B \Rightarrow E)}\ (\equiv)}{\Gamma \vdash \langle u, v\rangle[C] : C \Rightarrow A \Rightarrow D \wedge X \Rightarrow B \Rightarrow E}\ (\forall_e)
$$

La reducción ocurre como sigue:

$$
\begin{array}{l}
\quad \langle \Lambda X.\lambda x^X.\lambda y^A.t,\ \Lambda X.\lambda x^X.\lambda z^B.r\rangle[C] \\
\rightleftarrows_{(\text{P-DIST}_{\forall_e \wedge_i})} \\
\quad \langle (\lambda x^X.\lambda y^A.t)[C], (\lambda x^X.\lambda z^B.r)[C]\rangle \\
\hookrightarrow_{\beta_\Lambda} \\
\quad \langle \lambda x^C.\lambda y^A.t,\ \lambda x^C.\lambda z^B.r\rangle
\end{array}
$$

Ejemplo 5 Nuevamente, debido al isomorfismo (P-DIST), podremos construir el siguiente término, en el que se efectúa una aplicación de tipos sobre una proyección:

$$(\pi_{\forall X.(X \Rightarrow X)}(\Lambda X.\langle \lambda x^X.x, r\rangle))[\tau]$$

Su derivación de tipo es

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : X \vdash x : X}\,(ax)}{\vdash \lambda x^X.x : X \Rightarrow X}\,(\Rightarrow_i) \qquad \Gamma \vdash r : (\tau \Rightarrow \tau) \Rightarrow \tau}{\Gamma \vdash \langle \lambda x^X.x, r\rangle : (X \Rightarrow X) \wedge ((\tau \Rightarrow \tau) \Rightarrow \tau)}\,(\wedge_i)}{\Gamma \vdash \Lambda X.\langle \lambda x^X.x, r\rangle : \forall X.((X \Rightarrow X) \wedge ((\tau \Rightarrow \tau) \Rightarrow \tau))}\,(\forall_i)}{\Gamma \vdash \Lambda X.\langle \lambda x^X.x, r\rangle : (\forall X.(X \Rightarrow X)) \wedge ((\tau \Rightarrow \tau) \Rightarrow \tau)}\,(\equiv)}{\Gamma \vdash \pi_{\forall X.(X \Rightarrow X)}(\Lambda X.\langle \lambda x^X.x, r\rangle) : \forall X.(X \Rightarrow X)}\,(\wedge_e)}{\Gamma \vdash (\pi_{\forall X.(X \Rightarrow X)}(\Lambda X.\langle \lambda x^X.x, r\rangle))[\tau] : \tau \Rightarrow \tau}\,(\forall_e)$$

y su reducción

$$\begin{array}{l} \quad (\pi_{\forall X.(X \Rightarrow X)}(\Lambda X.\langle \lambda x^X.x, r\rangle))[\tau] \\ \rightleftarrows_{(\text{P-DIST}_{\forall_e \wedge_e})} \\ \quad \pi_{\tau \Rightarrow \tau}((\Lambda X.\langle \lambda x^X.x, r\rangle)[\tau]) \\ \hookrightarrow_{\beta_\Lambda} \\ \quad \pi_{\tau \Rightarrow \tau}\langle \lambda x^\tau.x, [X := \tau]r\rangle \\ \hookrightarrow_{\pi} \\ \quad \lambda x^\tau.x \end{array}$$

7.4 Propiedades

Probaremos la preservación de tipos en la reducción. Para ello deberemos caracterizar ciertas equivalencias entre tipos, como por ejemplo: si $\forall X.A \equiv B \wedge C$, entonces $B \equiv \forall X.B'$ y $C \equiv \forall X.C'$, con $A \equiv B' \wedge C'$ (Lema 7.4.8). Debido a la cantidad de isomorfismos, probar estos lemas no resulta trivial. Para ello introduciremos el concepto de *factores primos* de un tipo y una serie de propiedades relacionadas. Esta técnica fue utilizada en Sistema I, aunque solo con un tipo básico τ; en SIP, en cambio, la cantidad de variables que constituyen los tipos básicos son infinitas, y por lo tanto las pruebas resultan más complejas.

La noción de factores primos de un tipo se construye como una analogía a los factores primos de los números naturales, valiéndonos de la correspondencia entre conjunción y producto. La idea es definir tipos canónicos como representativos de la clase de equivalencia generada por ($\equiv$). Los factores primos de un tipo (Definición 7.4.1) son el multiconjunto de tipos que no son equivalentes a conjunciones de otros tipos (y por esto los llamamos primos) tal que la conjunción de todos los elementos del mismo son equivalentes al tipo original.

Escribiremos $\forall\vec{X}.A$ para $\forall X_1.\forall X_2.\ldots.\forall X_n.A$, para algún $n \geq 0$ (donde si $n = 0$, entonces $\forall\vec{X}.A = A$). Por simplicidad, escribiremos $\emptyset \Rightarrow X$ para X cuando resulte conveniente.

Definición 7.4.1 (Factores primos)**.**

$$\begin{aligned}
PF(X) &= [X] \\
PF(A \Rightarrow B) &= [\forall\vec{X}_i.((A \wedge B_i) \Rightarrow Y_i)]_{i=1}^n \\
&\qquad \textit{donde } PF(B) = [\forall\vec{X}_i.(B_i \Rightarrow Y_i)]_{i=1}^n \\
PF(A \wedge B) &= PF(A) \uplus PF(B) \\
PF(\forall X.A) &= [\forall X.\forall\vec{Y}_i.(A_i \Rightarrow Z_i)]_{i=1}^n \\
&\qquad \textit{donde } PF(A) = [\forall\vec{Y}_i.(A_i \Rightarrow Z_i)]_{i=1}^n
\end{aligned}$$

donde $\uplus$ es la unión de multiconjuntos, definida como es usual.

El Lema 7.4.2 y el Corolario 7.4.2.1 establecen la correctitud de la definición 7.4.1. Escribiremos $\bigwedge([A_i]_i)$ para $\bigwedge_i A_i$.

Lema 7.4.2. *Para todo A, $PF(A) = [\forall\vec{X}_i.(B_i \Rightarrow Y_i)]_{i=1}^n$.*

Prueba. Por inducción trivial en la estructura de A, teniendo en cuenta las notaciones definidas previamente. □

Corolario 7.4.2.1. *Para todo A, $A \equiv \bigwedge(PF(A))$.*

Prueba. Por inducción en la estructura de A.

- Sea $A = X$. Entonces $PF(X) = [X]$, y $\bigwedge([X]) = X$.
- Sea $A = B \Rightarrow C$.
  Por Lema 7.4.2, $PF(C) = [\forall\vec{X}_i.(C_i \Rightarrow Y_i)]_{i=1}^n$.
  Entonces, por definición, $PF(A) = [\forall\vec{X}_i.(B \wedge C_i \Rightarrow Y_i)]_{i=1}^n$.

Por hipótesis inductiva, $C \equiv \bigwedge(PF(C)) = \bigwedge_{i=1}^{n} \forall \vec{X}_i.(C_i \Rightarrow Y_i)$.
Por lo tanto,

$$\begin{aligned} A &= B \Rightarrow C \\ &\equiv B \Rightarrow \bigwedge_{i=1}^{n} \forall \vec{X}_i.(C_i \Rightarrow Y_i) \\ &\equiv \bigwedge_{i=1}^{n} \forall \vec{X}_i.((B \wedge C_i) \Rightarrow Y_i) \\ &= \bigwedge([\forall \vec{X}_i.(B \wedge C_i \Rightarrow Y_i)]_{i=1}^{n}) \\ &= \bigwedge(PF(A)) \end{aligned}$$

- Sea $A = B \wedge C$.
  Por hipótesis inductiva, $B \equiv \bigwedge(PF(B))$ y $C \equiv \bigwedge(PF(C))$.
  Entonces,

$$\begin{aligned} A &= B \wedge C \\ &\equiv \bigwedge(PF(B)) \wedge \bigwedge(PF(C)) \\ &\equiv \bigwedge(PF(B) \uplus PF(C)) \\ &= \bigwedge(PF(A)) \end{aligned}$$

- Sea $A = \forall X.B$.
  Por Lema 7.4.2, $PF(B) = [\forall \vec{Y}_i.(B_i \Rightarrow Z_i)]_{i=1}^{n}$.
  Entonces, por definición, $PF(A) = [\forall X.\forall \vec{Y}_i.(B_i \Rightarrow Z_i)]_{i=1}^{n}$.
  Por hipótesis inductiva, $B \equiv \bigwedge(PF(B)) = \bigwedge_{i=1}^{n} \forall \vec{Y}_i.(B_i \Rightarrow Z_i)$.
  Por lo tanto,

$$\begin{aligned} A &= \forall X.B \\ &\equiv \forall X. \bigwedge_{i=1}^{n} \forall \vec{Y}_i.(B_i \Rightarrow Z_i) \\ &\equiv \bigwedge_{i=1}^{n} \forall X.\forall \vec{Y}_i.(B_i \Rightarrow Z_i) \\ &= \bigwedge([\forall X.\forall \vec{Y}.(B_i \Rightarrow Z)]_{i=1}^{n}) \\ &= \bigwedge(PF(A)) \end{aligned}$$

□

La Definición 7.4.3 remite a la relación de equivalencia entre multiconjuntos (dos multiconjuntos son equivalentes si y solo si tienen los mismos elementos, módulo isomorfismos, en igual cantidad). El Lema 7.4.4 determina la estabilidad de los factores primos ante la equivalencia, es decir dos tipos equivalentes tienen los mismos factores primos, y el Lema 7.4.5 una suerte de propiedad recíproca.

Definición 7.4.3. $[A_1, \ldots, A_n] \sim [B_1, \ldots, B_m]$ *si* $n = m$ *y* $A_i \equiv B_{p(i)}$, *para* $i = 1, \ldots, n$ *y* p *una permutación de* $\{1, \ldots .n\}$.

Lema 7.4.4. *Si* $A \equiv B$, *entonces* $PF(A) \sim PF(B)$.

Prueba. Es sencillo comprobar que $PF(A \wedge B) \sim PF(B \wedge A)$ y similar para los otros cinco isomorfismos. Por lo tanto, se prueba con una inducción estructural trivial que si A y B son equivalentes en un paso, entonces PF(A) $\sim$ PF(B). Finalmente, se concluye con una inducción en la longitud de la derivación de la equivalencia $A \equiv B$. □

Lema 7.4.5. *Si* $R \sim S$, *entonces* $\bigwedge(R) \equiv \bigwedge(S)$.

Prueba. Directo de la definición de $\sim$. □

Los Lemas 7.4.7, 7.4.8 y 7.4.9 establecen consecuencias que se obtienen de ciertas equivalencias entre tipos. Las mismas serán de importancia para la prueba de preservación de tipos de SIP. El Lema 7.4.6 establece consecuencias de la equivalencia entre dos tipos primos, y será utilizado para probar dichos lemas.

Lema 7.4.6. *Si* $\forall \vec{X}.(A \Rightarrow Y) \equiv \forall \vec{Z}.(B \Rightarrow W)$, *entonces* $\vec{X} = \vec{Z}$, $A \equiv B$, *y* $Y = W$.

Prueba. Por inspección simple de los isomorfismos considerados. □

Lema 7.4.7. *Si* $A \Rightarrow B \equiv C_1 \wedge C_2$, *entonces* $C_1 \equiv A \Rightarrow B_1$, $C_2 \equiv A \Rightarrow B_2$ *y* $B \equiv B_1 \wedge B_2$.

Prueba. Por Lema 7.4.4, $PF(A \Rightarrow B) \sim PF(C_1 \wedge C_2) = PF(C_1) \uplus PF(C_2)$. Por Lema 7.4.2, sean

$$PF(B) = [\forall \vec{X}_i.(D_i \Rightarrow Z_i)]_{i=1}^{n}$$
$$PF(C_1) = [\forall \vec{Y}_j.(E_j \Rightarrow Z'_j)]_{j=1}^{k}$$
$$PF(C_2) = [\forall \vec{Y}_j.(E_j \Rightarrow Z'_j)]_{j=k+1}^{m}$$

Entonces $[\forall \vec{X}_i.((A \wedge D_i) \Rightarrow Z_i)]_{i=1}^{n} \sim [\forall \vec{Y}_j.(E_j \Rightarrow Z'_j)]_{j=1}^{m}$.
Por definición de $\sim$, $n = m$ y para $i = 1, \ldots, n$ y una permutación p,
tenemos $\forall \vec{X}_i.((A \wedge D_i) \Rightarrow Z_i) \equiv \forall \vec{Y}_{p(i)}.(E_{p(i)} \Rightarrow Z'_{p(i)})$.
Luego, por Lema 7.4.6, tenemos $\vec{X}_i = \vec{Y}_{p(i)}$, $A \wedge D_i \equiv E_{p(i)}$, y $Z_i = Z'_{p(i)}$.
Entonces existe I tal que

$$\begin{aligned} I \cup \bar{I} &= \{1, \ldots, n\} \\ PF(C_1) &= [\forall \vec{Y}_{p(i)}.(E_{p(i)} \Rightarrow Z'_{p(i)})]_{i \in I} \\ PF(C_2) &= [\forall \vec{Y}_{p(i)}.(E_{p(i)} \Rightarrow Z'_{p(i)})]_{i \in \bar{I}} \end{aligned}$$

Por lo tanto, por Corolario 7.4.2.1

$$C_1 \equiv \bigwedge_{i \in I} \forall \vec{Y}_{p(i)}.(E_{p(i)} \Rightarrow Z'_{p_i}) \equiv \bigwedge_{i \in I} \forall \vec{X}_i.((A \wedge D_i) \Rightarrow Z_i)$$

y de manera similar

$$C \equiv \bigwedge_{i \in \bar{I}} \forall \vec{X}_i.((A \wedge D_i) \Rightarrow Z_i)$$

Sean $B_1 = \bigwedge_{i \in I} \forall \vec{X}_i.(D_i \Rightarrow Z_i)$ y $B_2 = \bigwedge_{i \in \bar{I}} \forall \vec{X}_i.(D_i \Rightarrow Z_i)$.
Entonces $C_1 \equiv A \Rightarrow B_1$ y $C_2 \equiv A \Rightarrow B_2$.
Finalmente, también por Corolario 7.4.2.1, tenemos
$B \equiv \bigwedge_{i=1}^{n} \forall \vec{X}_i.(D_i \Rightarrow Z_i) \equiv B_1 \wedge B_2$. □

Lema 7.4.8. *Si* $\forall X.A \equiv B \wedge C$, *entonces* $B \equiv \forall X.B'$, $C \equiv \forall X.C'$ *y* $A \equiv B' \wedge C'$.

Prueba. Por Lema 7.4.4, $PF(\forall X.A) \sim PF(B \wedge C) = PF(B) \uplus PF(C)$.
Por Lema 7.4.2, sean

$$\begin{aligned} PF(A) &= [\forall \vec{Y}_i.(A_i \Rightarrow Z_i)]_{i=1}^{n} \\ PF(B) &= [\forall \vec{W}_j.(D_j \Rightarrow Z'_j)]_{j=1}^{k} \\ PF(C) &= [\forall \vec{W}_j.(D_j \Rightarrow Z'_j)]_{j=k+1}^{m} \end{aligned}$$

Entonces $[\forall X.\forall \vec{Y}_i.(A_i \Rightarrow Z_i)]_{i=1}^{n} \sim [\forall \vec{W}_j.(D_j \Rightarrow Z'_j)]_{j=1}^{m}$.
Luego, por definición de $\sim$, $n = m$ y para $i = 1, \ldots, n$ y una permutación p,
tenemos $\forall X.\forall \vec{Y}_i.(A_i \Rightarrow Z_i) \equiv \forall \vec{W}_{p(i)}.(D_{p(i)} \Rightarrow Z'_{p(i)})$.
Entonces, por Lema 7.4.6, tenemos $X, \vec{Y}_i = \vec{W}_{p(i)}$, $A_i \equiv D_{p(i)}$, y $Z_i = Z'_{p(i)}$.
Por lo tanto, existe I tal que

$$I \cup \bar{I} = \{1, \ldots, n\}$$

$$PF(B) = [\forall \vec{W}_{p(i)}.(D_{p(i)} \Rightarrow Z'_{p(i)})]_{i\in I}$$
$$PF(C) = [\forall \vec{W}_{p(i)}.(D_{p(i)} \Rightarrow Z'_{p(i)})]_{i\in \bar{I}}$$

Así, por Corolario 7.4.2.1, tenemos

$$B \equiv \bigwedge_{i\in I} \forall \vec{W}_{p(i)}.(D_{p(i)} \Rightarrow Z'_{p_i}) \equiv \bigwedge_{i\in I} \forall X.\forall \vec{Y}_i.(A_i \Rightarrow Z_i)$$

y de manera similar

$$C \equiv \bigwedge_{i\in \bar{I}} \forall X.\forall \vec{Y}_i.(A_i \Rightarrow Z_i)$$

Sean $B' = \bigwedge_{i\in I} \forall \vec{Y}_i.(A_i \Rightarrow Z_i)$ y $C' = \bigwedge_{i\in \bar{I}} \forall \vec{Y}_i.(A_i \Rightarrow Z_i)$.
Entonces $B \equiv \forall X.B'$ y $C \equiv \forall X.C'$.
Finalmente, también por Corolario 7.4.2.1, tenemos
$A \equiv \bigwedge_{i=1}^{n} \forall \vec{Y}_i.(A_i \Rightarrow Z_i) \equiv B' \wedge C'$. □

Lema 7.4.9. *Si $\forall X.A \equiv B \Rightarrow C$, entonces $C \equiv \forall X.C'$ y $A \equiv B \Rightarrow C'$.*

Prueba. Por Lema 7.4.4, $PF(\forall X.A) \sim PF(B \Rightarrow C)$.
Por Lema 7.4.2, sean

$$PF(A) = [\forall \vec{Y}_i.(A_i \Rightarrow Z_i)]_{i=1}^{n}$$
$$PF(C) = [\forall \vec{W}_j.(D_j \Rightarrow Z'_j)]_{j=1}^{m}$$

Entonces $[\forall X.\forall \vec{Y}_i.(A_i \Rightarrow Z_i)]_{i=1}^{n} \sim [\forall \vec{W}_j.((B \wedge D_j) \Rightarrow Z'_j)]_{j=1}^{m}$.
Luego, por definición de $\sim$, $n = m$ y para $i = 1, \ldots, n$ y una permutación p, tenemos $\forall X.\forall \vec{Y}_i.(A_i \Rightarrow Z_i) \equiv \forall \vec{W}_{p(i)}.((B \wedge D_{p(i)}) \Rightarrow Z'_{p(i)})$.
Por lo tanto, por Lema 7.4.6, tenemos

$$X, \vec{Y}_i = \vec{W}_{p(i)}$$
$$A_i \equiv B \wedge D_{p(i)}$$
$$Z_i = Z'_{p(i)}$$

Entonces, por Corolario 7.4.2.1

$$C \equiv \bigwedge_{j=1}^{n} \forall \vec{W}_j.(D_j \Rightarrow Z'_j)$$
$$\equiv \bigwedge_{i=1}^{n} \forall \vec{W}_{p(i)}.(D_{p(i)} \Rightarrow Z'_{p(i)})$$

$$\equiv \bigwedge_{i=1}^{n} \forall X.\forall \vec{Y_i}.(D_{p(i)} \Rightarrow Z_i)$$

Sea $C' = \bigwedge_{i=1}^{n} \forall \vec{Y_i}.(D_{p(i)} \Rightarrow Z_i)$.
Entonces $C \equiv \forall X.C'$.
Finalmente, también por Corolario 7.4.2.1, tenemos

$$\begin{aligned} A &\equiv \bigwedge_{i=1}^{n} \forall \vec{Y_i}.(A_i \Rightarrow Z_i) \\ &\equiv \bigwedge_{i=1}^{n} \forall \vec{Y_i}.((B \wedge D_{p(i)}) \Rightarrow Z_i) \\ &\equiv B \Rightarrow \bigwedge_{i=1}^{n} \forall \vec{Y_i}.(D_{p(i)} \Rightarrow Z_i) \\ &\equiv B \Rightarrow C' \end{aligned}$$

□

Dado que el cálculo se presenta *à la Church*, el mismo es dirigido por sintaxis, excluyendo la regla (≡). Por lo tanto el lema de generación es trivial (Lema 7.4.10) y tenemos un lema de unicidad (Lema 7.4.11).

Lema 7.4.10 (Generación)**.**

1. *Si* $\Gamma \vdash x : A$ *y* $\Gamma \vdash x : B$*, entonces* $A \equiv B$*.*
2. *Si* $\Gamma \vdash \lambda x^A.r : B$*, entonces* $\Gamma, x : A \vdash r : C$ *y* $B \equiv A \Rightarrow C$*.*
3. *Si* $\Gamma \vdash rs : B$*, entonces* $\Gamma \vdash r : A \Rightarrow B$ *y* $\Gamma \vdash s : A$*.*
4. *Si* $\Gamma \vdash \langle r, s\rangle : A$*, entonces* $A \equiv B \wedge C$*,* $\Gamma \vdash r : B$ *y* $\Gamma \vdash s : C$*.*
5. *Si* $\Gamma \vdash \pi_A r : B$*, entonces* $A \equiv B$ *y* $\Gamma \vdash r : B \wedge C$*.*
6. *Si* $\Gamma \vdash \Lambda X.r : B$*, entonces* $B \equiv \forall X.A$*,* $\Gamma \vdash r : A$ *y* $X \notin FTV_c(\Gamma)$*.*
7. *Si* $\Gamma \vdash r[A] : B$*, entonces* $[X := A]C \equiv B$ *y* $\Gamma \vdash r : \forall X.C$*.* □

Lema 7.4.11 (Unicidad módulo)**.**
Si $\Gamma \vdash r : A$ *y* $\Gamma \vdash r : B$*, entonces* $A \equiv B$*.*

Prueba. Si la última regla de la derivación de $\Gamma \vdash r : A$ es (≡), entonces existe una derivación más corta $\Gamma \vdash r : C$ con $C \equiv A$. Por hipótesis inductiva, $C \equiv B$ y por lo tanto $A \equiv B$. Si la última regla de la derivación de $\Gamma \vdash r : B$ es (≡), procedemos de la misma manera. Todos los demás casos son dirigidos por sintaxis. □

El siguiente lema de substitución es central para probar el teorema de preservación de tipos.

Lema 7.4.12 (Substitución).

1. *Si* $\Gamma, x : B \vdash r : A$ *y* $\Gamma \vdash s : B$ *entonces* $\Gamma \vdash [x := s]r : A$.
2. *Si* $\Gamma \vdash r : A$, *entonces* $[X := B]\Gamma \vdash [X := B]r : [X := B]A$.

Prueba.

1. Por inducción estructural en r.

   - Sea $r = x$.
     Por Lema 7.4.10, $A \equiv B$,
     por lo tanto $\Gamma \vdash s : A$.
     Como $[x := s]x = s$,
     entonces $\Gamma \vdash [x := s]x : A$.
   - Sea $r = y$, con $y \neq x$.
     Como $[x := s]y = y$,
     entonces $\Gamma \vdash [x := s]y : A$.
   - Sea $r = \lambda x^C.t$.
     Tenemos $[x := s](\lambda x^C.t) = \lambda x^C.t$,
     entonces $\Gamma \vdash [x := s](\lambda x^C.t) : A$.
   - Sea $r = \lambda y^C.t$, con $y \neq x$.
     Por Lema 7.4.10, $A \equiv C \Rightarrow D$ y $\Gamma, y : C \vdash t : D$.
     Por hipótesis inductiva, $\Gamma, y : C \vdash [x := s]t : D$.
     Por regla $(\Rightarrow_i)$, $\Gamma \vdash \lambda y^C.[x := s]t : C \Rightarrow D$.
     Como $\lambda y^C.[x := s]t = [x := s](\lambda y^C.t)$,
     usando la regla $(\equiv)$, tenemos $\Gamma \vdash [x := s](\lambda x^C.t) : A$.
   - Sea $r = tu$.
     Por Lema 7.4.10, $\Gamma \vdash t : C \Rightarrow A$ y $\Gamma \vdash u : C$.
     Por hipótesis inductiva, $\Gamma \vdash [x := s]t : C \Rightarrow A$ y $\Gamma \vdash [x := s]u : C$.
     Por regla $(\Rightarrow_e)$, $\Gamma \vdash ([x := s]t)([x := s]u) : A$.
     Como $([x := s]t)([x := s]u) = [x := s](tu)$,
     tenemos $\Gamma \vdash [x := s](tu) : A$.
   - Sea $r = \langle t, u \rangle$.
     Por Lema 7.4.10, $\Gamma \vdash t : C$ y $\Gamma \vdash u : D$, con $A \equiv C \wedge D$.
     Por hipótesis inductiva, $\Gamma \vdash [x := s]t : C$ y $\Gamma \vdash [x := s]u : D$
     Por regla $(\wedge_i)$, $\Gamma \vdash \langle [x := s]t, [x := s]u \rangle : C \wedge D$.
     Como $\langle [x := s]t, [x := s]u \rangle = [x := s]\langle t, u \rangle$,
     usando la regla $(\equiv)$, tenemos $\Gamma \vdash [x := s]\langle t, u \rangle : A$.

- Sea $r = \pi_A t$.
  Por Lema 7.4.10, $\Gamma \vdash t : A \wedge C$.
  Por hipótesis inductiva, $\Gamma \vdash [x := s]t : A \wedge C$.
  Por regla $(\wedge_e)$, $\Gamma \vdash \pi_A([x := s]t) : A$.
  Como $\pi_A([x := s]t) = [x := s](\pi_A t)$,
  tenemos $\Gamma \vdash [x := s](\pi_A t) : A$.
- Sea $r = \Lambda X.t$.
  Por Lema 7.4.10, $A \equiv \forall X.C$ y $\Gamma \vdash t : C$.
  Por hipótesis inductiva, $\Gamma \vdash [x := s]t : C$.
  Por regla $(\forall_i)$, $\Gamma \vdash \Lambda X.[x := s]t : \forall X.C$.
  Como $\Lambda X.[x := s]t = [x := s](\Lambda X.t)$,
  usando la regla $(\equiv)$, tenemos $\Gamma \vdash [x := s](\Lambda X.t) : A$.
- Sea $r = t[C]$.
  Por Lema 7.4.10, $A \equiv [X := C]D$ y $\Gamma \vdash t : \forall X.D$.
  or hipótesis inductiva, $\Gamma \vdash [x := s]t : \forall X.D$.
  Por regla $(\forall_e)$, $\Gamma \vdash ([x := s]t)[C] : [X := C]D$.
  Como $([x := s]t)[C] = [x := s](t[C])$,
  usando la regla $(\equiv)$, tenemos $\Gamma \vdash [x := s](t[C]) : A$.

2. Por inducción en las reglas de tipado.

- (ax): Sea $\Gamma, x : A \vdash x : A$.
  Usando la regla (ax), tenemos
  $[X := B]\Gamma, x : [X := B]A \vdash [X := B]x : [X := B]A$.
- $(\equiv)$: Sea $\Gamma \vdash r : A$, con $A \equiv C$.
  Por hipótesis inductiva, $[X := B]\Gamma \vdash [X := B]r : [X := B]C$.
  Como $A \equiv C$, entonces $[X := B]A \equiv [X := B]C$.
  Usando la regla $(\equiv)$, tenemos
  $[X := B]\Gamma \vdash [X := B]r : [X := B]A$.
- $(\Rightarrow_i)$: Sea $\Gamma \vdash \lambda x^C.t : C \Rightarrow D$.
  Por hipótesis inductiva,
  $[X := B]\Gamma, x : [X := B]C \vdash [X := B]t : [X := B]D$.
  Usando la regla $(\Rightarrow_i)$, tenemos
  $[X := B]\Gamma \vdash \lambda x^{[X:=B]C}.[X := B]t : [X := B]C \Rightarrow [X := B]D$.
  Como $\lambda x^{[X:=B]C}.[X := B]t = [X := B](\lambda x^C.t)$,
  se cumple $[X := B]\Gamma \vdash [X := B](\lambda x^C.t) : [X := B](C \Rightarrow D)$.
- $(\Rightarrow_e)$: Sea $\Gamma \vdash ts : D$.
  Por hipótesis inductiva,
  $[X := B]\Gamma \vdash [X := B]t : [X := B](C \Rightarrow D)$

y $[X := B]\Gamma \vdash [X := B]s : [X := B]C$.
Como $[X := B](C \Rightarrow D) = [X := B]C \Rightarrow [X := B]D$,
por regla $(\Rightarrow_e)$ tenemos
$[X := B]\Gamma \vdash ([X := B]t)([X := B]s) : [X := B]D$.
Dado que se cumple $([X := B]t)([X := B]s) = [X := B](ts)$,
también se cumple $[X := B]\Gamma \vdash [X := B](ts) : [X := B]D$.

- $(\wedge_i)$: Sea $\Gamma \vdash \langle t, s\rangle : C \wedge D$.
Por hipótesis inductiva,
$[X := B]\Gamma \vdash [X := B]t : [X := B]C$
y $[X := B]\Gamma \vdash [X := B]s : [X := B]D$.
Usando la regla $(\wedge_i)$, tenemos
$[X := B]\Gamma \vdash \langle [X := B]t, [X := B]s\rangle : [X := B]C \wedge [X := B]D$.
Como $\langle [X := B]t, [X := B]s\rangle = [X := B]\langle t, s\rangle$
y $[X := B]C \wedge [X := B]D = [X := B](C \wedge D)$,
se cumple $[X := B]\Gamma \vdash [X := B]\langle t, s\rangle : [X := B](C \wedge D)$.

- $(\wedge_e)$: Sea $\Gamma \vdash t : C \wedge D$.
Por hipótesis inductiva,
$[X := B]\Gamma \vdash [X := B]t : [X := B](C \wedge D)$.
Como $[X := B](C \wedge D) = [X := B]C \wedge [X := B]D$,
por regla $(\wedge_e)$ tenemos
$[X := B]\Gamma \vdash \pi_{[X:=B]C}([X := B]t) : [X := B]C$.
Dado que $\pi_{[X:=B]C}[X := B]t = [X := B]\pi_C t$,
se cumple $[X := B]\Gamma \vdash [X := B]\pi_C t : [X := B]C$.

- $(\forall_i)$: Sea $\Gamma \vdash \Lambda Y.t : \forall Y.C$, con $X \notin FTV_c(\Gamma)$.
Por hipótesis inductiva, $[X := B]\Gamma \vdash [X := B]t : [X := B]C$.
Como $X \notin FTV_c(\Gamma)$, $X \notin FTV_c([X := B]\Gamma)$,
usando la regla $(\forall_i)$, tenemos
$[X := B]\Gamma \vdash \Lambda Y.[X := B]t : \Lambda Y.[X := B]C$.
Dado que se cumple $\Lambda Y.[X := B]t = [X := B]\Lambda Y.t$
y $\forall Y.[X := B]C = [X := B]\forall Y.C$,
también se cumple $[X := B]\Gamma \vdash [X := B]\Lambda Y.t : [X := B]\forall Y.C$.

- $(\forall_e)$: Sea $\Gamma \vdash t[D] : [Y := D]C$.
Por hipótesis inductiva, $[X := B]\Gamma \vdash [X := B]t : [X := B]\forall Y.C$.
Como $[X := B]\forall Y.C = \forall Y.[X := B]C$,
usando la regla $(\forall_e)$ tenemos $[X := B]\Gamma \vdash ([X := B]t)[[X := B]D] : [Y := [X := B]D][X := B]C$.
Dado que $([X := B]t)[[X := B]D] = [X := B](t[D])$
y $[Y := [X := B]D][X := B]C = [X := B][Y := D]C$,
se cumple $[X := B]\Gamma \vdash [X := B](t[D]) : [X := B][Y := D]C$. $\square$

Teorema 7.4.13 (Preservación de tipos).
Si $\Gamma \vdash t : A$ *y* $t \hookrightarrow r$ *o* $t \rightleftarrows r$, *entonces* $\Gamma \vdash r : A$.

Prueba. Por inducción en las relaciones $\rightleftarrows$ y $\hookrightarrow$.

- (comm): $\langle t, r\rangle \rightleftarrows \langle r, t\rangle$

  ($\rightarrow$) 1. $\Gamma \vdash \langle t, r\rangle : A$ (Hipótesis)
  2. $A \equiv B \wedge C$
     $\Gamma \vdash t : B$
     $\Gamma \vdash r : C$ (1, Lema 7.4.10)
  3. $B \wedge C \equiv C \wedge B$ (Iso. (comm))
  4.
  $$\dfrac{\dfrac{\dfrac{\Gamma \vdash r : C \quad \Gamma \vdash t : B}{\Gamma \vdash \langle r, t\rangle : C \wedge B}\,(\wedge_i)}{\Gamma \vdash \langle r, t\rangle : B \wedge C}\,(\equiv)\ [3]}{\Gamma \vdash \langle r, t\rangle : A}\,(\equiv)\ [2]$$

  ($\leftarrow$) análoga a ($\rightarrow$).

- (asso): $\langle t, \langle r, s\rangle\rangle \rightleftarrows \langle\langle t, r\rangle, s\rangle$

  ($\rightarrow$) 1. $\Gamma \vdash \langle t, \langle r, s\rangle\rangle : A$ (Hipótesis)
  2. $A \equiv B \wedge C$
     $\Gamma \vdash t : B$
     $\Gamma \vdash \langle r, s\rangle : C$ (1, Lema 7.4.10)
  3. $C \equiv D \wedge E$
     $\Gamma \vdash r : D$
     $\Gamma \vdash s : E$ (2, Lema 7.4.10)
  4. $B \wedge (D \wedge E) \equiv (B \wedge D) \wedge E$ (Iso. (asso))
  5. $A \equiv B \wedge (D \wedge E)$ (2, 3, congr. ($\equiv$))
  6.
  $$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash t : B \quad \Gamma \vdash r : D}{\Gamma \vdash \langle t, r\rangle : B \wedge D}\,(\wedge_i) \qquad \Gamma \vdash s : E}{\Gamma \vdash \langle\langle t, r\rangle, s\rangle : (B \wedge D) \wedge E}\,(\wedge_i)}{\Gamma \vdash \langle\langle t, r\rangle, s\rangle : B \wedge (D \wedge E)}\,(\equiv)\ [4]}{\Gamma \vdash \langle\langle t, r\rangle, s\rangle : A}\,(\equiv)\ [5]$$

($\leftarrow$) análoga a ($\rightarrow$).

- ($\textsc{dist}_\lambda$): $\lambda x^A.\langle t, r\rangle \rightleftarrows \langle \lambda x^A.t, \lambda x^A.r\rangle$

($\rightarrow$)
1. $\Gamma \vdash \lambda x^A.\langle t, r\rangle : B$ (Hipótesis)
2. $B \equiv A \Rightarrow C$
   $\Gamma, x : A \vdash \langle t, r\rangle : C$ (1, Lema 7.4.10)
3. $C \equiv D \wedge E$
   $\Gamma, x : A \vdash t : D$
   $\Gamma, x : A \vdash r : E$ (2, Lema 7.4.10)
4. $A \Rightarrow (D \wedge E) \equiv (A \Rightarrow D) \wedge (A \Rightarrow E)$ (Iso. ($\textsc{dist}$))
5. $B \equiv A \Rightarrow (D \wedge E)$ (2, 3, congr. ($\equiv$))
6.
$$\dfrac{\dfrac{\dfrac{\Gamma, x : A \vdash t : D}{\Gamma \vdash \lambda x^A.t : A \Rightarrow D}\,(\Rightarrow_i) \quad \dfrac{\Gamma, x : A \vdash r : E}{\Gamma \vdash \lambda x^A.r : A \Rightarrow E}\,(\Rightarrow_i)}{\Gamma \vdash \langle \lambda x^A.t, \lambda x^A.r\rangle : (A \Rightarrow D) \wedge (A \Rightarrow E)}\,(\wedge_i)\ [4]}{\Gamma \vdash \langle \lambda x^A.t, \lambda x^A.r\rangle : A \Rightarrow (D \wedge E)}\,(\equiv)\ [5]$$
$$\dfrac{\Gamma \vdash \langle \lambda x^A.t, \lambda x^A.r\rangle : A \Rightarrow (D \wedge E)}{\Gamma \vdash \langle \lambda x^A.t, \lambda x^A.r\rangle : B}\,(\equiv)$$

($\leftarrow$)
1. $\Gamma \vdash \langle \lambda x^A.t, \lambda x^A.r\rangle : B$ (Hipótesis)
2. $B \equiv C \wedge D$
   $\Gamma \vdash \lambda x^A.t : C$
   $\Gamma \vdash \lambda x^A.r : D$ (1, Lema 7.4.10)
3. $C \equiv A \Rightarrow C'$
   $\Gamma, x : A \vdash t : C'$ (2, Lema 7.4.10)
4. $D \equiv A \Rightarrow D'$
   $\Gamma, x : A \vdash r : D'$ (2, Lema 7.4.10)
5. $(A \Rightarrow C') \wedge (A \Rightarrow D') \equiv A \Rightarrow (C' \wedge D')$ (Iso. ($\textsc{dist}$))
6. $B \equiv (A \Rightarrow C') \wedge (A \Rightarrow D')$ (2, 3, 4, congr. ($\equiv$))
7.
$$\dfrac{\dfrac{\dfrac{\Gamma, x : A \vdash t : C' \quad \Gamma, x : A \vdash r : D'}{\Gamma, x : A \vdash \langle t, r\rangle : C' \wedge D'}\,(\wedge_i)}{\Gamma \vdash \lambda x^A.\langle t, r\rangle : A \Rightarrow (C' \wedge D')}\,(\Rightarrow_i)\ [5]}{\Gamma \vdash \lambda x^A.\langle t, r\rangle : (A \Rightarrow C') \wedge (A \Rightarrow D')}\,(\equiv)\ [6]$$
$$\dfrac{\Gamma \vdash \lambda x^A.\langle t, r\rangle : (A \Rightarrow C') \wedge (A \Rightarrow D')}{\Gamma \vdash \lambda x^A.\langle t, r\rangle : B}\,(\equiv)$$

- (DIST_{app}): $\langle t, r\rangle s \rightleftarrows \langle ts, rs\rangle$

($\rightarrow$) 1. $\Gamma \vdash \langle t, r\rangle s : A$ (Hipótesis)

2. $\Gamma \vdash \langle t, r\rangle : B \Rightarrow A$
   $\Gamma \vdash s : B$ (1, Lema 7.4.10)

3. $B \Rightarrow A \equiv C \wedge D$
   $\Gamma \vdash t : C$
   $\Gamma \vdash r : D$ (2, Lema 7.4.10)

4. $C \equiv B \Rightarrow C'$
   $D \equiv B \Rightarrow D'$
   $A \equiv C' \wedge D'$ (3, Lema 7.4.7)

5. $$[4]\ \frac{\dfrac{\Gamma \vdash t : C}{\Gamma \vdash t : B \Rightarrow C'}\ (\equiv) \qquad \Gamma \vdash s : B}{\Gamma \vdash ts : C'}\ (\Rightarrow_e)$$

6. $$[4]\ \frac{\dfrac{\Gamma \vdash r : D}{\Gamma \vdash r : B \Rightarrow D'}\ (\equiv) \qquad \Gamma \vdash s : B}{\Gamma \vdash rs : D'}\ (\Rightarrow_e)$$

7. $$[4]\ \frac{\dfrac{\overset{(5)}{\overline{\Gamma \vdash ts : C'}} \qquad \overset{(6)}{\overline{\Gamma \vdash rs : D'}}}{\Gamma \vdash \langle ts, rs\rangle : C' \wedge D'}\ (\wedge_i)}{\Gamma \vdash \langle ts, rs\rangle : A}\ (\equiv)$$

($\leftarrow$) 1. $\Gamma \vdash \langle ts, rs\rangle : A$ (Hipótesis)

2. $A \equiv B \wedge C$
   $\Gamma \vdash ts : B$
   $\Gamma \vdash rs : C$ (1, Lema 7.4.10)

3. $\Gamma \vdash t : D \Rightarrow B$
   $\Gamma \vdash s : D$ (2, Lema 7.4.10)

4. $\Gamma \vdash r : E \Rightarrow B$
   $\Gamma \vdash s : E$ (2, Lema 7.4.10)

5. $D \equiv E$ (3, 4, Lema 7.4.11)

6. $D \Rightarrow (B \wedge C) \equiv (D \Rightarrow B) \wedge (D \Rightarrow C)$ (Iso. (DIST))

7. $E \Rightarrow C \equiv D \Rightarrow C$ (6, congr. ($\equiv$))

8.

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash t : D \Rightarrow B \qquad [7]\ \dfrac{\Gamma \vdash r : E \Rightarrow C}{\Gamma \vdash r : D \Rightarrow C}\ (\equiv)}{\Gamma \vdash \langle t, r\rangle : (D \Rightarrow B) \wedge (D \Rightarrow C)}\ (\wedge_i)}{\Gamma \vdash \langle t, r\rangle : D \Rightarrow (B \wedge C)}\ [5]\ (\equiv)}{\Gamma \vdash \langle t, r\rangle s : B \wedge C}\ (\Rightarrow_e)\quad [2]\ (\equiv)$$

$$\Gamma \vdash \langle t, r\rangle s : A$$

- (CURRY): $t\langle r, s\rangle \rightleftarrows trs$

($\rightarrow$)

1. $\Gamma \vdash t\langle r, s\rangle : A$ (Hipótesis)
2. $\Gamma \vdash t : B \Rightarrow A$
   $\Gamma \vdash \langle t, r\rangle : B$ (1, Lema 7.4.10)
3. $B \equiv C \wedge D$
   $\Gamma \vdash r : C$
   $\Gamma \vdash s : D$ (2, Lema 7.4.10)
4. $B \Rightarrow A \equiv (C \wedge D) \Rightarrow A$ (3, congr. ($\equiv$))
5. $(C \wedge D) \Rightarrow A \equiv C \Rightarrow (D \Rightarrow A)$ (Iso. (CURRY))
6. $$\dfrac{\dfrac{[4]\ \dfrac{\Gamma \vdash t : B \Rightarrow A}{\Gamma \vdash t : (C \wedge D) \Rightarrow A}\ (\equiv)}{\Gamma \vdash t : C \Rightarrow (D \Rightarrow A)}\ [5]\ (\equiv) \qquad \Gamma \vdash r : C}{\Gamma \vdash tr : D \Rightarrow A}\ (\Rightarrow_e)$$
7. $$\dfrac{\dfrac{(6)}{\Gamma \vdash tr : D \Rightarrow A} \qquad \Gamma \vdash s : D}{\Gamma \vdash trs : A}\ (\Rightarrow_e)$$

($\leftarrow$)

1. $\Gamma \vdash trs : A$ (Hipótesis)
2. $\Gamma \vdash tr : B \Rightarrow A$
   $\Gamma \vdash s : B$ (1, Lema 7.4.10)
3. $\Gamma \vdash t : C \Rightarrow (B \Rightarrow A)$
   $\Gamma \vdash r : C$ (2, Lema 7.4.10)
4. $C \Rightarrow (B \Rightarrow A) \equiv (C \wedge B) \Rightarrow A$ (Iso. (CURRY))
5. $$\dfrac{[4]\ \dfrac{\Gamma \vdash t : C \Rightarrow (B \Rightarrow A)}{\Gamma \vdash t : (C \wedge B) \Rightarrow A}\ (\equiv) \qquad \dfrac{\Gamma \vdash r : C \qquad \Gamma \vdash s : B}{\Gamma \vdash \langle r, s\rangle : C \wedge B}\ (\wedge_i)}{\Gamma \vdash t\langle r, s\rangle : A}\ (\Rightarrow_e)$$

- ($\text{P-COMM}_{\forall_i \Rightarrow_i}$): $\Lambda X.\lambda x^A.t \rightleftarrows \lambda x^A.\Lambda X.t$

($\rightarrow$)

1. $X \notin FTV_A(A)$ (Hipótesis)
2. $\Gamma \vdash \Lambda X.\lambda x^A.t : B$ (Hipótesis)
3. $B \equiv \forall X.C$
   $\Gamma \vdash \lambda x^A.t : C$
   $X \notin FTV_c(\Gamma)$ (2, Lema 7.4.10)
4. $C \equiv A \Rightarrow D$
   $\Gamma, x : A \vdash t : D$ (3, Lema 7.4.10)
5. $\forall X.(A \Rightarrow D) \equiv A \Rightarrow \forall X.D$ (1, Iso. (P-COMM))
6. $\forall X.C \equiv \forall X.(A \Rightarrow D)$ (4, congr. ($\equiv$))
7. $$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, x : A \vdash t : D}{\Gamma, x : A \vdash \Lambda X.t : \forall X.D}\ (\forall_i)}{\Gamma \vdash \lambda x^A.\Lambda X.t : A \Rightarrow \forall X.D}\ (\Rightarrow_i)}{\Gamma \vdash \lambda x^A.\Lambda X.t : \forall X.(A \Rightarrow D)}\ (\equiv)}{\Gamma \vdash \lambda x^A.\Lambda X.t : \forall X.C}\ (\equiv)}{\Gamma \vdash \lambda x^A.\Lambda X.t : B}\ (\equiv)$$
   [1 3] [5] [6] [3]

($\leftarrow$)

1. $X \notin FTV_A(A)$ (Hipótesis)
2. $\Gamma \vdash \lambda x^A.\Lambda X.t : B$ (Hipótesis)
3. $B \equiv A \Rightarrow C$
   $\Gamma, x : A \vdash \Lambda X.t : C$ (2, Lema 7.4.10)
4. $C \equiv \forall X.D$
   $\Gamma, x : A \vdash t : D$
   $X \notin FTV_c(\Gamma) \cup FTV_A(A)$ (3, Lema 7.4.10)
5. $\forall X.(A \Rightarrow D) \equiv A \Rightarrow \forall X.D$ (1, Iso. (P-COMM))
6. $A \Rightarrow C \equiv A \Rightarrow \forall X.D$ (4, congr. ($\equiv$))
7.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, x : A \vdash t : D}{\Gamma \vdash \lambda x^A.t : A \Rightarrow D}\ (\Rightarrow_i)}{\Gamma \vdash \Lambda X.\lambda x^A.t : \forall X.(A \Rightarrow D)}\ (\forall_i)}{\Gamma \vdash \Lambda X.\lambda x^A.t : A \Rightarrow \forall X.D}\ (\equiv)}{\Gamma \vdash \Lambda X.\lambda x^A.t : A \Rightarrow C}\ (\equiv)}{\Gamma \vdash \Lambda X.\lambda x^A.t : B}\ (\equiv)$$

- ($\text{P-COMM}_{\forall_e \Rightarrow_i}$): $(\lambda x^A.t)[B] \rightleftarrows \lambda x^A.t[B]$

($\rightarrow$)

1. $X \notin FTV_A(A)$ (Hipótesis)
2. $\Gamma \vdash (\lambda x^A.t)[B] : C$ (Hipótesis)
3. $C \equiv [X := B]D$
   $\Gamma \vdash \lambda x^A.t : \forall X.D$ (2, Lema 7.4.10)
4. $\forall X.D \equiv A \Rightarrow E$
   $\Gamma, x : A \vdash t : E$ (3, Lema 7.4.10)
5. $E \equiv \forall X.E'$
   $D \equiv A \Rightarrow E'$ (4, Lema 7.4.9)
6. $A \Rightarrow [X := B]E' = [X := B](A \Rightarrow E')$ (1, Def.)
7. $[X := B](A \Rightarrow E') \equiv [X := B]D$ (5, congr. ($\equiv$))
8. $$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, x : A \vdash t : E}{\Gamma, x : A \vdash t : \forall X.E'}\ (\equiv)}{\Gamma, x : A \vdash t[B] : [X := B]E'}\ (\forall_e)}{\Gamma \vdash \lambda x^A.t[B] : A \Rightarrow [X := B]E'}\ (\Rightarrow_i)}{\Gamma \vdash \lambda x^A.t[B] : [X := B](A \Rightarrow E')}\ (\equiv)}{\Gamma \vdash \lambda x^A.t[B] : [X := B]D}\ (\equiv)}{\Gamma \vdash \lambda x^A.t[B] : C}\ (\equiv)$$

($\leftarrow$)

1. $X \notin FTV_A(A)$ (Hipótesis)
2. $\Gamma \vdash \lambda x^A.t[B] : C$ (Hipótesis)
3. $C \equiv A \Rightarrow D$
   $\Gamma, x : A \vdash t[B] : D$ (1, Lema 7.4.10)
4. $D \equiv [X := B]E$
   $\Gamma, x : A \vdash t : \forall X.E$ (2, Lema 7.4.10)

5. $A \Rightarrow \forall X.E \equiv \forall X.(A \Rightarrow E)$ (Iso. (P-COMM))
6. $[X := B](A \Rightarrow E) = A \Rightarrow [X := B]E$ (1, Def.)
7. $A \Rightarrow [X := B]E \equiv A \Rightarrow D$ (4, congr. $(\equiv)$)
8.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, x : A \vdash t : \forall X.E}{\Gamma \vdash \lambda x^A.t : A \Rightarrow \forall X.E}\,(\Rightarrow_i)}{\Gamma \vdash \lambda x^A.t : \forall X.(A \Rightarrow E)}\,[5]\,(\equiv)}{\Gamma \vdash (\lambda x^A.t)[B] : [X := B](A \Rightarrow E)}\,(\forall_e)}{\Gamma \vdash (\lambda x^A.t)[B] : A \Rightarrow [X := B]E}\,[6]\,(\equiv)}{\Gamma \vdash (\lambda x^A.t)[B] : A \Rightarrow D}\,[7]\,(\equiv)}{\Gamma \vdash (\lambda x^A.t)[B] : C}\,[3]\,(\equiv)$$

- (P-DIST$_{\forall_i \wedge_i}$): $\Lambda X.\langle t, r\rangle \rightleftarrows \langle \Lambda X.t, \Lambda X.r\rangle$

($\rightarrow$)
1. $\Gamma \vdash \Lambda X.\langle t, r\rangle : A$ (Hipótesis)
2. $A \equiv \forall X.B$
   $\Gamma \vdash \langle t, r\rangle : B$
   $X \notin FTV_c(\Gamma)$ (1, Lema 7.4.10)
3. $B \equiv C \wedge D$
   $\Gamma \vdash t : C$
   $\Gamma \vdash r : D$ (2, Lema 7.4.10)
4. $\forall X.(C \wedge D) \equiv \forall X.C \wedge \forall X.D$ (Iso. (P-DIST))
5. $\forall X.B \equiv \forall X.(C \wedge D)$ (3, congr. $(\equiv)$)
6.

$$\dfrac{\dfrac{\dfrac{[2]\ \dfrac{\Gamma \vdash t : C}{\Gamma \vdash \Lambda X.t : \forall X.C}\,(\forall_i) \qquad [2]\ \dfrac{\Gamma \vdash r : D}{\Gamma \vdash \Lambda X.r : \forall X.D}\,(\forall_i)}{\Gamma \vdash \langle \Lambda X.t, \Lambda X.r\rangle : \forall X.C \wedge \forall X.D}\,(\wedge_i)}{\Gamma \vdash \langle \Lambda X.t, \Lambda X.r\rangle : \forall X.(C \wedge D)}\,[4]\,(\equiv)}{\Gamma \vdash \langle \Lambda X.t, \Lambda X.r\rangle : \forall X.B}\,[5]\,(\equiv)}{\Gamma \vdash \langle \Lambda X.t, \Lambda X.r\rangle : A}\,[2]\,(\equiv)$$

($\leftarrow$)
1. $\Gamma \vdash \langle \Lambda X.t, \Lambda X.r\rangle : A$ (Hipótesis)
2. $A \equiv B \wedge C$
   $\Gamma \vdash \Lambda X.t : B$
   $\Gamma \vdash \Lambda X.r : C$ (1, Lema 7.4.10)

3. $B \equiv \forall X.D$
   $\Gamma \vdash t : D$
   $X \notin FTV_c(\Gamma)$ (2, Lema 7.4.10)
4. $C \equiv \forall X.E$
   $\Gamma \vdash r : E$
   $X \notin FTV_c(\Gamma)$ (2, Lema 7.4.10)
5. $\forall X.(D \wedge E) \equiv \forall X.D \wedge \forall X.E$ (Iso. (P-DIST))
6. $\forall X.D \wedge \forall X.E \equiv B \wedge C$ (3, 4, congr. ($\equiv$))
7.

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash t : D \quad \Gamma \vdash r : E}{\Gamma \vdash \langle t, r\rangle : D \wedge E}\ (\wedge_i)}{[3]\ \Gamma \vdash \Lambda X.\langle t, r\rangle : \forall X.(D \wedge E)}\ (\forall_i)}{[5]\ \Gamma \vdash \Lambda X.\langle t, r\rangle : \forall X.D \wedge \forall X.E}\ (\equiv)}{[6]\ \Gamma \vdash \Lambda X.\langle t, r\rangle : B \wedge C}\ (\equiv)$$

$$[2]\ \Gamma \vdash \Lambda X.\langle t, r\rangle : A\ (\equiv)$$

- (P-DIST$_{\forall_e \wedge_i}$): $\langle t, r\rangle[B] \rightleftarrows \langle t[B], r[B]\rangle$

($\rightarrow$) 1. $\Gamma \vdash \langle t, r\rangle[B] : A$ (Hipótesis)

2. $A \equiv [X := B]C$
   $\Gamma \vdash \langle t, r\rangle : \forall X.C$ (1, Lema 7.4.10)
3. $\forall X.C \equiv D \wedge E$
   $\Gamma \vdash t : D$
   $\Gamma \vdash r : E$ (2, Lema 7.4.10)
4. $D \equiv \forall X.D'$
   $E \equiv \forall X.E'$
   $C \equiv D' \wedge E'$ (3, Lema 7.4.8)
5. $[X := B](D' \wedge E') = [X := B]D' \wedge [X := B]E'$ (Def.)
6. $[X := B]C \equiv [X := B](D' \wedge E')$ (4, congr. ($\equiv$))
7.

$$\dfrac{\dfrac{\dfrac{[4]\ \dfrac{\Gamma \vdash t : D}{\Gamma \vdash t : \forall X.D'}\ (\equiv)}{\Gamma \vdash t[B] : [X := B]D'}\ (\forall_e) \qquad \dfrac{[4]\ \dfrac{\Gamma \vdash r : E}{\Gamma \vdash r : \forall X.E'}\ (\equiv)}{\Gamma \vdash r[B] : [X := B]E'}\ (\forall_e)}{[5]\ \dfrac{\Gamma \vdash \langle t[B], r[B]\rangle : [X := B]D' \wedge [X := B]E'}{[6]\ \dfrac{\Gamma \vdash \langle t[B], r[B]\rangle : [X := B](D' \wedge E')}{[2]\ \dfrac{\Gamma \vdash \langle t[B], r[B]\rangle : [X := B]C}{\Gamma \vdash \langle t[B], r[B]\rangle : A}\ (\equiv)}\ (\equiv)}\ (\equiv)}\ (\wedge_i)$$

($\leftarrow$) 1. $\Gamma \vdash \langle t[B], r[B]\rangle : A$ (Hipótesis)

2. $A \equiv C \wedge D$
   $\Gamma \vdash t[B] : C$
   $\Gamma \vdash r[B] : D$ (1, Lema 7.4.10)

3. $C \equiv [X := B]C'$
   $\Gamma \vdash t : \forall X.C'$ (2, Lema 7.4.10)

4. $D \equiv [X := B]D'$
   $\Gamma \vdash r : \forall X.D'$ (2, Lema 7.4.10)

5. $\forall X.(C' \wedge D') \equiv \forall X.C' \wedge \forall X.D'$ (Iso. (P-DIST))

6. $[X := B](C' \wedge D') = [X := B]C' \wedge [X := B]D'$ (Def.)

7. $[X := B]C' \wedge [X := B]D' \equiv C \wedge D$ (3, 4, congr. ($\equiv$))

8. $$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash t : \forall X.C' \qquad \Gamma \vdash r : \forall Y.D'}{[5]\ \dfrac{\Gamma \vdash \langle t, r\rangle : \forall X.C' \wedge \forall X.D'}{\Gamma \vdash \langle t, r\rangle : \forall X.(C' \wedge D')}\ (\equiv)}\ (\wedge_i)}{[6]\ \dfrac{\Gamma \vdash \langle t, r\rangle[B] : [X := B](C' \wedge D')}{\Gamma \vdash \langle t, r\rangle[B] : [X := B]C' \wedge [X := B]D'}\ (\equiv)}\ (\forall_e)}{[7]\ \dfrac{\Gamma \vdash \langle t, r\rangle[B] : C \wedge D}{[2]\ \dfrac{\Gamma \vdash \langle t, r\rangle[B] : A}{\Gamma \vdash \langle t, r\rangle[B] : A}\ (\equiv)}\ (\equiv)}\ (\equiv)}$$

- ($\text{P-DIST}_{\forall_i \wedge_e}$): $\pi_{\forall X.B}(\Lambda X.t) \rightleftarrows \Lambda X.\pi_B t$

($\rightarrow$) 1. $\Gamma \vdash \pi_{\forall X.B}(\Lambda X.t) : A$ (Hipótesis)

2. $A \equiv \forall X.B$
   $\Gamma \vdash \Lambda X.t : (\forall X.B) \wedge C$ (1, Lema 7.4.10)

3. $(\forall X.B) \wedge C \equiv \forall X.D$
   $\Gamma \vdash t : D$
   $X \notin FTV_c(\Gamma)$ (2, Lema 7.4.10)
4. $C \equiv \forall X.C'$
   $D \equiv B \wedge C'$ (3, Lema 7.4.8)
5.
$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash t : D}{\Gamma \vdash t : B \wedge C'}\,(\equiv)}{\Gamma \vdash \pi_B t : B}\,(\wedge_e)}{\Gamma \vdash \Lambda X.\pi_B t : \forall X.B}\,(\forall_i)}{\Gamma \vdash \Lambda X.\pi_B t : A}\,(\equiv)$$
$$[4]\quad [3]\quad [2]$$

($\leftarrow$)
1. $\Gamma \vdash \Lambda X.\pi_B t : A$ (Hipótesis)
2. $A \equiv \forall X.C$
   $\Gamma \vdash \pi_B t : C$
   $X \notin FTV_c(\Gamma)$ (1, Lema 7.4.10)
3. $B \equiv C$
   $\Gamma \vdash t : C \wedge D$ (2, Lema 7.4.10)
4. $\forall X.(C \wedge D) \equiv \forall X.C \wedge \forall X.D$ (Iso. (P-DIST))
5.
$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash t : C \wedge D}{\Gamma \vdash \Lambda X.t : \forall X.(C \wedge D)}\,(\forall_i)}{\Gamma \vdash \Lambda X.t : \forall X.C \wedge \forall X.D}\,(\equiv)}{\Gamma \vdash \pi_{\forall X.B}(\Lambda X.t) : \forall X.C}\,(\wedge_e)}{\Gamma \vdash \pi_{\forall X.B}(\Lambda X.t) : A}\,(\equiv)$$
$$[4]\quad [2]$$

- (P-DIST$_{\forall_e \wedge_e}$): $(\pi_{\forall X.B} t)[C] \rightleftarrows \pi_{[X:=C]B}(t[C])$

($\rightarrow$)
1. $\Gamma \vdash t : \forall X.(B \wedge D)$ (Hipótesis)
2. $\Gamma \vdash (\pi_{\forall X.B} t)[C] : A$ (Hipótesis)
3. $A \equiv [X := C]E$
   $\Gamma \vdash \pi_{\forall X.B} t : \forall X.E$ (2, Lema 7.4.10)
4. $\forall X.E \equiv \forall X.B$
   $\Gamma \vdash t : \forall X.E \wedge F$ (3, Lema 7.4.10)
5. $E \equiv B$ (4, congr. $(\equiv)$)
6. $[X := C](B \wedge D) = [X := C]B \wedge [X := C]D$ (Def.)

7. $[X := C]B \equiv [X := C]E$ (5, congr. $(\equiv)$)
8. $$\dfrac{\dfrac{\dfrac{\Gamma \vdash t : \forall X.(B \wedge D)}{\Gamma \vdash t[C] : [X := C](B \wedge D)}\ (\forall_e)}{\Gamma \vdash t[C] : [X := C]B \wedge [X := C]D}\ (\equiv)\ [6]}{\dfrac{\Gamma \vdash \pi_{[X:=C]B}(t[C]) : [X := C]B}{\Gamma \vdash \pi_{[X:=C]B}(t[C]) : [X := C]E}\ (\equiv)\ [7]}\ (\wedge_e)$$ $$\dfrac{\Gamma \vdash \pi_{[X:=C]B}(t[C]) : [X := C]E}{\Gamma \vdash \pi_{[X:=C]B}(t[C]) : A}\ (\equiv)\ [3]$$

($\leftarrow$)
1. $\Gamma \vdash t : \forall X.(B \wedge D)$ (Hipótesis)
2. $\Gamma \vdash \pi_{[X:=C]B}(t[C]) : A$ (Hipótesis)
3. $A \equiv [X := C]B$
   $\Gamma \vdash t[C] : A \wedge E$ (2, Lema 7.4.10)
4. $\forall X.(B \wedge D) \equiv \forall X.B \wedge \forall X.D$ (Iso. (P-DIST))
5. $$\dfrac{\dfrac{\dfrac{\Gamma \vdash t : \forall X.(B \wedge D)}{\Gamma \vdash t : \forall X.B \wedge \forall X.D}\ (\equiv)\ [4]}{\Gamma \vdash \pi_{\forall X.B} t : \forall X.B}\ (\wedge_e)}{\Gamma \vdash (\pi_{\forall X.B} t)[C] : [X := C]B}\ (\forall_e)$$ $$\dfrac{\Gamma \vdash (\pi_{\forall X.B} t)[C] : [X := C]B}{\Gamma \vdash (\pi_{\forall X.B} t)[C] : A}\ (\equiv)\ [3]$$

- (β_λ): si $\Gamma \vdash s : A$, $(\lambda x^A.r)s \hookrightarrow [x := s]r$
  1. $\Gamma \vdash s : A$ (Hipótesis)
  2. $\Gamma \vdash \lambda x^A.r : B$ (Hipótesis)
  3. $\Gamma \vdash \lambda x^A.r : A \Rightarrow B$ (2, Lema 7.4.10)
  4. $A \Rightarrow B \equiv A \Rightarrow C$
     $\Gamma, x : A \vdash r : C$ (3, Lema 7.4.10)
  5. $B \equiv C$ (4, congr. $(\equiv)$)
  6. $\Gamma \vdash [x := s]r : C$ (1, 4, Lema 7.4.12)
  7. $\Gamma \vdash [x := s]r : B$ (5, 6, regla $(\equiv)$)

- (β_Λ): $(\Lambda X.r)[A] \hookrightarrow [X := A]r$

1. $\Gamma \vdash (\Lambda X.r)[A] : B$ (Hipótesis)
2. $B \equiv [X := A]C$
   $\Gamma \vdash \Lambda X.r : \forall X.C$ (1, Lema 7.4.10)
3. $\forall X.C \equiv \forall X.D$
   $\Gamma \vdash r : D$
   $X \notin FTV_c(\Gamma)$ (2, Lema 7.4.10)
4. $C \equiv D$ (3)
5. $\Gamma \vdash r : C$ (4, regla ($\equiv$))
6. $[X := A]\Gamma \vdash \Gamma \vdash [X := A]r : [X := A]C$ (5, Lema 7.4.12)
7. $\Gamma \vdash [X := A]r : B$ (2, 3, 7, regla ($\equiv$))

- (π): si $\Gamma \vdash r : A$, $\pi_A\langle r, s\rangle \hookrightarrow r$

1. $\Gamma \vdash r : A$ (Hipótesis)
2. $\pi_A\langle r, s\rangle : B$ (Hipótesis)
3. $B \equiv A$
   $\Gamma \vdash \langle r, s\rangle : A \wedge C$ (2, Lema 7.4.10)
4. $\Gamma \vdash \pi_A\langle r, s\rangle : A$ (2, 3, regla ($\equiv$))

□

Capítulo 8

Conclusiones y trabajo futuro

8.1 Conclusiones

En este trabajo expusimos un problema de rigidez en los sistemas de tipos en lenguajes de programación, y en la relación de derivación en los sistemas de pruebas. Los mismos distinguen elementos que tienen diferente forma (es decir difieren sintácticamente) pero mismo significado, como pueden ser las pruebas de las conjunciones $A \wedge B$ y $B \wedge A$, de las implicaciones $(A \wedge B) \Rightarrow C$ y $A \Rightarrow B \Rightarrow C$, o de la cuantificación e implicación, respectivamente, $\forall X.(A \Rightarrow B)$ y $A \Rightarrow \forall X.B$ cuando $X \notin FTV_A(A)$. De esta manera impide, por ejemplo, que una prueba de la proposición $A \wedge B$ constituya una prueba para $B \wedge A$, cuando se puede demostrar mediante la existencia de un isomorfismo que dichas proposiciones son efectivamente equivalentes.

Analizamos el trabajo realizado en torno a la familia de sistemas módulo isomorfismos, en particular Sistema I, y propusimos una extensión a dicho sistema con polimorfismo, a la que llamamos Sistema I Polimórfico. Describimos las diferentes características particulares del mismo, a saber el conjunto de isomorfismos a identificar y las consecuencias en la relación de tipado, reducción y equivalencia entre tipos y términos. Expusimos los motivos de las decisiones de inclusión y exclusión de isomorfismos y de equivalencias entre tipos y términos. Y finalmente probamos la propiedad de preservación de tipos para el cálculo.

8.2 Trabajo futuro

8.2.1 Adición de conectivas

Un área a explorar es la de agregar más conectivas. Una de ellas podría ser un elemento neutro para la conjunción, $\top$. Esto implicaría más isomorfismos para el conjunto de base, en particular $A \wedge \top \equiv A$, $A \Rightarrow \top \equiv \top$ y $\top \Rightarrow A \equiv A$ [9]. Agregar la equivalencia $\top \Rightarrow \top \equiv \top$ permitiría construir el término $\Omega = (\lambda x^{\top}.xx)(\lambda x^{\top}.xx) : \top$, que al tener tipo $\top$ podría forzarse a reducir a algún término normalizante, como podría ser el que descarte su argumento, mediante restricciones a la regla beta: "si $A \equiv \top$, entonces $(\lambda x^{A}.r)s \to [x := \star]r$", siendo $\vdash \star : \top$ la regla de introducción para $\top$.

8.2.2 Terminación

La normalización fuerte en Sistema I se probó [11] utilizando una reformulación no trivial de la prueba clásica de Tait para tipos simples. No puede usarse la noción de términos neutrales [19] (usualmente definidos como aquellos de eliminación, como son los de aplicación y proyección) ni la propiedad CR3 porque, tanto en Sistema I como en Sistema I Polimórfico, la neutralidad no es estable respecto a la equivalencia $\rightleftarrows$. Por ejemplo, $\langle r, s\rangle t$ es una aplicación y por lo tanto es neutra, pero el término equivalente $\langle rt, st\rangle$ es un par y por lo tanto no es neutral. Conjeturamos que es posible la extensión de la técnica de Sistema I a Sistema I Polimórfico.

8.2.3 Eta-expansion rule

Una extensión de una versión preliminar de Sistema I [10] se implementó en Haskell [13] con la adición de algunas reglas para lograr la propiedad de progresión (es decir que solo las introducciones sean formas normales), por ejemplo "si $s : B$, entonces $(\lambda x^{A}.\lambda y^{B}.r)s \to \lambda x^{A}.(\lambda y^{B}.r)s$". Esta regla, junto a otras de las introducidas en la implementación, es un caso particular de la η-expansión. Con la regla $t \to \lambda x^{A}.tx$ es posible obtener $(\lambda x^{A}.\lambda y^{B}.r)s \to \lambda z^{A}.(\lambda x^{A}.\lambda y^{B}.r)sz \rightleftarrows^{*} \lambda z^{A}.(\lambda x^{A}.\lambda y^{B}.r)zs \to \lambda z^{A}.((\lambda y^{B}.r[z/x])s)$. Recientemente se extendió Sistema I con η-expansiones, mostrando la propiedad de progresión [12]. Dejamos como trabajo futuro extender SIP de la misma manera.

8.2.4 Implementación y punto fijo

Como se mencionó en la sección anterior, existe [13] una implementación de una extensión de una versión preliminar de Sistema I. Además de las reglas mencionadas, se agregó un operador de punto fijo. Planeamos extender dicha implementación siguiendo el diseño de Sistema I Polimórfico. También se consideran formalizaciones de otros lenguajes como Coq o Agda para trabajo futuro, con el fin de desarrollar bibliotecas o extensiones de los mismos.

Referencias

[1] Beniamino Accattoli y Alejandro Díaz-Caro. The distributive λ-calculus. Aceptado en FLOPS 2020. Por aparecer en LNCS, 2020.

[2] Henk Barendregt. *The lambda calculus: its syntax and semantics*, tomo 103 de *Studies in logic and the foundations of mathematics*. Nord-Holland, 1981.

[3] Henk Barendregt. *Lambda Calculi with Types*, página 117–309. Oxford University Press, 1993. ISBN 0198537611.

[4] Gérard Boudol. Lambda-calculi for (strict) parallel functions. *Inf. and Comp.*, 108(1):51–127, 1994.

[5] Antonio Bucciarelli, Thomas Ehrhard, y Giulio Manzonetto. A relational semantics for parallelism and non-determinism in a functional setting. *APAL*, 163(7):918–934, 2012.

[6] Haskell B. Curry. Grundlagen der kombinatorischen logik. *American Journal of Mathematics*, 52(3):509–536, 1930. ISSN 00029327, 10806377.

[7] Ugo de'Liguoro y Adolfo Piperno. Non deterministic extensions of untyped λ-calculus. *Inf. and Comp.*, 122(2):149–177, 1995.

[8] Mariangiola Dezani-Ciancaglini, Ugo de'Liguoro, y Adolfo Piperno. A filter model for concurrent λ-calculus. *SIAM JComp.*, 27(5):1376–1419, 1998.

[9] Roberto Di Cosmo. *Isomorphisms of types: from λ-calculus to information retrieval and language design*. Progress in Theoretical Computer Science. Birkhauser, 1995. ISBN 1461275857.

[10] Alejandro Díaz-Caro y Gilles Dowek. Non determinism through type isomorphism. En Delia Kesner y Petrucio Viana, editores, *Proceedings of the 7th Workshop on Logical and Semantic Frameworks, with Applications (LSFA 2012)*, tomo 113 de *Electronic Proceedings in Theoretical Computer Science*, páginas 137–144. Open Publishing Association, 2013.

[11] Alejandro Díaz-Caro y Gilles Dowek. Proof normalisation in a logic identifying isomorphic propositions. En Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, tomo 131 de *Leibniz International Proceedings in Informatics (LIPIcs)*, páginas 14:1–14:23. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

[12] Alejandro Díaz-Caro y Gilles Dowek. Extensional proofs in a propositional logic modulo isomorphisms. *Comunicación privada*, 2020.

[13] Alejandro Díaz-Caro y Pablo E. Martínez López. Isomorphisms considered as equalities: Projecting functions and enhancing partial application through an implementation of λ^+. En *Proceedings of the 27th Symposium on the Implementation and Application of Functional Programming Languages*, IFL '15, páginas 9:1–9:11. ACM, 2015.

[14] Gilles Dowek y Jean-Jacques Lvy. *Introduction to the Theory of Programming Languages*. Springer Publishing Company, Incorporated, 1st edición, 2010. ISBN 0857290754.

[15] Jacques Garrigue y Hassan Aït-Kaci. The typed polymorphic label-selective λ-calculus. En Hans-J. Boehm, Bernard Lang, y Daniel Yellin, editores, *Proceedings of the 21st ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages*, POPL '94, página 35–47. Association for Computing Machinery, 1994. ISBN 0897916360.

[16] Gerhard Gentzen. Untersuchungen über das logische schließen. i. *Mathematische Zeitschrift*, 39(1):176–210, 1935. ISSN 1432-1823. doi: 10.1007/BF01201353.

[17] Gerhard Gentzen. Untersuchungen über das logische schließen. ii. *Mathematische Zeitschrift*, 39(1):405–431, 1935. ISSN 1432-1823. doi: 10.1007/BF01201363.

[18] Jean-Yves Girard. Une extension de l'interpretation de Göedel à l'analyse, et son application à l'élimination des coupures dans l'analyse

et la théorie des types. En Jens Erik Fenstad, editor, *Proceedings of the second Scandinavian logic symposium*, páginas 63–92. 1971. ISBN 0720422590.

[19] Jean-Yves Girard. *Proofs and types*. Cambridge University Press, 1989. ISBN 0521371813.

[20] Mark P. Jones. *Qualified Types: Theory and Practice*. Cambridge University Press, USA, 1995. ISBN 0521472539.

[21] Per Martin-Löf. An intuitionistic theory of types: predicative part. En Harvey E. Rose y J. C. Shepherdson, editores, *Proceedings of the Logic Colloquium*, tomo 80 de *Studies in logic and the foundations of mathematics*, páginas 73–118. North-Holland, 1973.

[22] Per Martin-Löf. *Intuitionistic type theory*. Bibliopolis, 1984.

[23] Elliott Mendelson. *Introduction to mathematical logic*. Chapman & Hall, 4a edición, 1997. ISBN 0412808307.

[24] Michele Pagani y Simona Ronchi Della Rocca. Linearity, non-determinism and solvability. *Fund. Inf.*, 103(1–4):173–202, 2010.

[25] Moses Schönfinkel. Über die bausteine der mathematischen logik. *Mathematische Annalen*, 92:305–316, 2013.

[26] Morten Heine Sørensen y Pawel Urzyczyn. *Lectures on the Curry-Howard Isomorphism, Volume 149 (Studies in Logic and the Foundations of Mathematics)*. Elsevier Science Inc., USA, 2006. ISBN 0444520775.

[27] Cristian Sottile, Alejandro Díaz-Caro, y Pablo E. Martínez López. Hacia un Sistema I Polimórfico. En *XXV Congreso Argentino de Ciencias de la Computación - CACIC*. 2019.

[28] Philip Wadler. Propositions as types. *Commun. ACM*, 58(12):75–84, Nov. 2015. ISSN 0001-0782. doi:10.1145/2699407.