Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Sistemas de tipos para λ-cálculo y Lógica Combinatoria

Andrés Ezequiel Viso
LU: 379/05

Tesis de
Licenciatura en Ciencias de la Computación

Director:
Dr. Ariel Arbiser
FCEyN UBA

Buenos Aires, Febrero de 2010

Resumen

El λ-cálculo puede verse como un lenguaje de programación universal en el que las funciones son "ciudadanos de primera clase". Este lenguaje puede representar todas las funciones computables. Resulta de especial interés el estudio de los sistemas tipados. Los tipos permiten una clasificación o "estratificación" del universo de valores y expresiones, en la cual se basan los lenguajes de programación modernos para la rápida detección del uso inapropiado de funciones, métodos, etc.

Como una formulación alternativa del λ-cálculo surge la Lógica Combinatoria, que es un sistema de reescritura en cierto sentido más simple pero sin embargo igualmente expresivo. Y del mismo modo que para el λ-cálculo, existen formulaciones de Lógica Combinatoria tipada.

En esta tesis se dan pruebas de consistencia para distintas versiones del λ-cálculo tipado. Para esto se plantean métodos de demostración puramente sintácticos, basados en la noción de variable positiva. Estas pruebas son comparadas con otras existentes en la literatura para algunos de los sistemas de tipos analizados. Se estudian las ventajas y limitaciones del método propuesto, identificando sistemas para los cuales éste no resulta aplicable, y sobre algunos de ellos se da una demostración adecuada.

Al mismo tiempo, se estudia la Lógica Combinatoria y sus variantes tipadas con el fin de definir un sistema de tipos de segundo orden. Se consideran diferentes opciones para extender el sistema de tipos simples de Curry. El sistema obtenido resulta equivalente al sistema de tipos polimórficos F_η del λ-cálculo, presentado por Mitchell.

Abstract

λ-calculus can be seen as a universal programming language in which functions are "first-class citizens". This language can represent all computable functions. Of special interest is the study of typed systems. Types allow a classification or "stratification" for the universe of values and expressions, on which modern programming languages are based for a quick detection of the inappropriate use of functions, methods, etc.

As an alternative formulation to λ-calculus, Combinatory Logic arises, a rewriting system in a sense simpler but nevertheless equally expressive. And in the same way as for lambda-calculus, there exist typed formulations of combinatory logic.

In the present thesis we give consistency proofs for different typed versions of the λ-calculus. To this effect, we introduce purely syntactical proof methods, based on the notion of positive variable. Said proofs for the systems analyzed are compared with other available in the literature. Advantages and limitations of this method are studied, identifying systems where it cannot be applied, and for some of them an adequate proof is given.

In addition, we study the Combinatory Logic and its typed variants in order to define a second-order type system. Different options are considered, extending Curry's simply-typed system. The resulting system turns out to be equivalent to the F_η polymorphic type system for the λ-calculus, introduced by Mitchell.

Agradecimientos

Quería agradecer y dedicar la presente tesis a mi familia y amigos, por el apoyo durante todo este tiempo. También a los docentes y compañeros que me acompañaron en estos seis años de carrera. Particularmente a Ariel, por la dedicación y el entusiasmo puesto en el desarrollo del trabajo, y a Mariano, fuente inagotable de consultas y chequeo de resultados, por el aporte en todo el trabajo, especialmente con sus conocimientos de LaTeX sin los cuales este informe sería muy diferente.

Andrés E. Viso

Índice general

Capítulo 1

Introducción

Dentro de la reescritura [2, 5], el λ-cálculo [3, 4] puede verse como un lenguaje de programación universal en el que las funciones son, como se suele sostener, "ciudadanos de primera clase": se puede aplicar una función a otra función y se puede devolver una función como resultado de otra. A pesar de la simplicidad de su sintaxis, este lenguaje es suficientemente rico para representar todas las funciones computables. Desde los orígenes de las Ciencias de la Computación, el λ-cálculo ha sido utilizado exitosamente ya que constituye el núcleo de los lenguajes de programación funcional, desde LISP hasta los lenguajes de la familia ML (como Caml, SML y Haskell). Hoy en día este formalismo es una forma de sistema de reescritura, sin embargo aún presenta interés en sí mismo. Resulta de especial interés el estudio de los sistemas tipados [4], que se utilizan para modelar y estudiar los lenguajes tipados, más particularmente la naturaleza de la noción de tipo.

El λ-cálculo admite versiones tipadas [3, 4]. Los tipos permiten clasificar los datos que manipulan los programas. Esta clasificación o "estratificación" del universo de valores y expresiones es un punto clave en el cual se basan los lenguajes de programación modernos para la rápida detección del uso inapropiado de funciones, métodos, etc. Los sistemas de tipos son sistemas deductivos que asignan tipos a las expresiones dentro del lenguaje (así como a programas enteros). Estos sistemas son útiles, por ejemplo, para garantizar que el número y naturaleza de los parámetros pasados a un procedimiento o método sean los adecuados. En general, el hecho de que muchos errores comunes puedan detectarse sin necesidad de ejecutar (o evaluar) un programa dado convierte a los sistemas de tipos en una componente verdaderamente importante de cualquier lenguaje de programación.

Los datos y programas pueden tiparse y esto permite garantizar de mejor manera propiedades de terminación, así como el estudio de los valores en estos sistemas de reescritura. Por otra parte, λ-cálculo es una herramienta fundamental para describir el comportamiento de las demostraciones matemáticas a través de la correspondencia de Curry–Howard [29], en la cual los tipos representan las fórmulas de una lógica determinada. Se usa por tanto λ-cálculo como base de los asistentes de prueba actuales.

Como una formulación alternativa del λ-cálculo surge la lógica combinatoria [28, 12, 13, 14], que es un sistema de reescritura en cierto sentido más simple pero sin embargo igualmente expresivo, permitiendo simular el mecanismo de abstracción de aquél y representar todas las funciones computables. Este sistema se puede presentar dentro del esquema de reescritura de términos de primer orden, lo cual da una mayor simplicidad de representación y estudio puesto que no utiliza el concepto de variable ligada.

Del mismo modo que para el λ-cálculo, existen formulaciones de lógica combinatoria tipada, lo que lleva a un estudio similar al planteado anteriormente sobre propiedades de los términos tipables. Además, estos sistemas guardan estrecha relación con los planteados para λ-cálculo, pues resultan equivalentes bajo cierta noción. Esto plantea la posibilidad de realizar un análisis más en detalle sobre la correspondencia que existe entre ambos formalismos, ya que permite estudiar el comportamiento de los términos de ambos cálculos a través de los tipos que a estos pueden asignarse.

1.1. Objetivos de la tesis

Nuestra tarea será el estudio de la consistencia de los sistemas de tipos principales de las distintas versiones de λ-cálculo. Fundamentalmente estudiamos pruebas de consistencia basadas en métodos sintácticos, más precisamente sobre la aparición de variables en las fórmulas de determinadas maneras, por ejemplo de manera positiva. Según la noción a usar, estas apariciones se preservarán a lo largo de cada paso de inferencia de tipado, lo que permitirá asegurar que no todas las fórmulas son demostrables, o dicho de otro modo, no todos los tipos son habitados, por lo tanto se tiene así la consistencia. Sin embargo, no todos los sistemas permiten las mismas posibilidades, y de lo que se trata es de estudiar esta diversidad así como obtener pruebas alternativas para unos u otros en caso de que sea necesario.

Las pruebas puramente sintácticas como las antedichas tienen interés porque dejan de lado las nociones semánticas de la lógica utilizada, que pueden ser engorrosas o no estar disponibles. Por otra parte, la ventaja de obtener estas pruebas radica en su sencillez: no se hacen necesarias las demostraciones de otras propiedades más costosas del cálculo tipado asociado, o que incluso podrían no valer, entre ellas la normalización fuerte o débil de todos los términos tipables.

Así mismo, haremos extensivo el planteo de sistemas de tipos para la lógica combinatoria, introduciendo un nuevo formalismo que permite la asignación de tipos de segundo orden a términos del cálculo. Estudiaremos las propiedades principales de este nuevo sistema y su relación con el λ-cálculo tipado de segundo orden, conocido como *System F*, y sus variantes propuestas en la literatura. Este nuevo planteo introduce una noción de subtipado propia del sistema equivalente a la presentada por Mitchell en [24], y que a la vez puede ser vista como una generalización de relaciones de subtipado ya conocidas, como la presentada por Barendregt en [4] para los tipos intersección.

1.2. Preliminares

La presente sección tiene como fin introducir todos los elementos que el lector necesita manejar para la mejor comprensión de los distintos puntos de esta tesis. Describimos brevemente los distintos sistemas involucrados en el desarrollo de la tesis, junto con las propiedades relevantes a las cuales haremos referencia a lo largo del mismo. Para más información en lo referente a estos formalismos, se recomienda consultar la bibliografía sugerida en cada caso.

1.2.1. Reescritura básica

Introducimos en primer lugar las nociones básicas de reescritura a las que haremos referencia de manera recurrente a lo largo del documento. Estos formalismos son desarrollados con mayor detalle en [2, 5].

Definición 1.2.1. *Un* sistema abstracto de reescritura *(ARS) es un par* $\mathcal{A} = (A, \to_A)$ *, donde* A *es un conjunto arbitrario de elementos (el* carrier set *de* $\mathcal{A}$*)* $y \to_A$ *es una relación binaria sobre* A*.*

La relación $\to_A$ es usualmente llamada *reducción* y se dice que l reduce en r siempre que $l \to_A r$, con $l, r \in A$.

Denotaremos con $\xrightarrow{=}_A$, $\xrightarrow{+}_A$ y $\xrightarrow{*}_A$ (o alternativamente $\twoheadrightarrow_A$) a las clausuras reflexiva, transitiva y reflexiva-transitiva de $\to_A$ respectivamente. Adicionalmente, definimos la equivalencia $=_A$ como la clausura reflexiva, simétrica y transitiva de la relación de reducción.

Decimos que un elemento $x \in A$ está en *forma normal* si no existe $y \in A$ tal que $x \to_A y$, es decir, x no es reducible. La condición de normalización fuerte garantiza la existencia de formas normales.

Definición 1.2.2. *Un ARS* $\mathcal{A} = (A, \to_A)$ *se dice* fuertemente normalizante *(SN) si no existe una secuencia infinita de reducciones*

$$M_1 \to_A M_2 \to_A \cdots \to_A M_i \to_A \cdots$$

con $(M_i)_{i \in \mathbb{N}} \in A$*.*

Por otro lado, un ARS puede tener la propiedad de ser confluente, lo que nos garantiza la unicidad de formas normales.

Definición 1.2.3. *Un ARS* $\mathcal{A} = (A, \to_A)$ *es* confluente *o, equivalentemente, cumple la propiedad* Church–Rosser *(CR) sii para elementos cualesquiera* M*,* M_1*,* $M_2 \in A$ *vale*

$$M \twoheadrightarrow_A M_1 \wedge M \twoheadrightarrow_A M_2 \Rightarrow \exists\, M_3 \in A \;\; [M_1 \twoheadrightarrow_A M_3 \wedge M_2 \twoheadrightarrow_A M_3]$$

1.2.2. El λ-cálculo clásico

El λ-cálculo fue introducido por Church en los años '30 [9] como formalización del concepto de función computable. Éste sirve como base para el paradigma de lenguajes funcionales, como LISP o la familia de lenguajes ML.

Sintaxis

El λ-cálculo posee dos operaciones básicas: la aplicación y la abstracción. La primera notada con la yuxtaposición de dos términos FN, donde F es considerado la función a la cual se le aplica el argumento N. La abstracción se denota con el constructor λ, representando $\lambda x.M$ a la función de parámetro formal x y cuerpo M. De este modo, el conjunto Λ de términos del cálculo queda definido por:

$$M ::= x \mid MM \mid \lambda x.M$$

donde x es una variable de un conjunto infinito numerable $\mathbb{V}$.

Como convención la aplicación es asociativa a izquierda, de modo que el término $M_1 M_2 \dots M_n$ es interpretado como $(\dots(M_1 M_2)\dots)M_n$. Por otro lado, para simplificar la lectura se abreviarán las abstracciones sucesivas de modo que $\lambda x_1.(\dots(\lambda x_n.M)\dots) = \lambda x_1 \dots x_n.M$.

Convención de variables

El constructor λ es interpretado como un ligador de variables, por esto, diremos que x está *ligada* en el término $\lambda x.N$, quedando definido el conjunto de variables ligadas de un término de la siguiente manera:

Definición 1.2.4 (Variables ligadas de un Λ-término)**.** *Sea* $\mathsf{BV}(\cdot) : \Lambda \to \wp(\mathbb{V})$ *el siguiente mapeo de términos a conjuntos de variables:*

$$\begin{aligned} \mathsf{BV}(x) &= \varnothing \\ \mathsf{BV}(MN) &= \mathsf{BV}(M) \cup \mathsf{BV}(N) \\ \mathsf{BV}(\lambda x.M) &= \mathsf{BV}(M) \cup \{x\} \end{aligned}$$

Por el contrario, toda variable del término que no esté ligada, se dice *libre*. De este modo, se define el conjunto de variables libres de un término M, notado $\mathsf{FV}(M)$, como:

Definición 1.2.5 (Variables libres de un Λ-término)**.** *Sea* $\mathsf{FV}(\cdot) : \Lambda \to \wp(\mathbb{V})$ *el siguiente mapeo de términos a conjuntos de variables:*

$$\begin{aligned} \mathsf{FV}(x) &= \{x\} \\ \mathsf{FV}(MN) &= \mathsf{FV}(M) \cup \mathsf{FV}(N) \\ \mathsf{FV}(\lambda x.M) &= \mathsf{FV}(M) - \{x\} \end{aligned}$$

Diremos que un término es *cerrado* si su conjunto de variables libres es vacio.

La operación de renombre de variables ligadas de un término, por variables que no ocurran en el mismo, es conocida como *α-conversión*. Ésta no altera el comportamiento del término, lo que da lugar a la *convención de variables libres de Barendregt*. Dicha convención consiste en cocientar el conjunto Λ por la relación de equivalencia entre términos inducida por la α-conversión, de modo de no distinguir dos términos que difieren sólo en los nombres de sus variables ligadas.

Esto nos permite asumir, en todo momento, que los conjuntos de variables ligadas y libres de un término son disjuntos o, dicho de otro modo, los nombres de las variables ligadas son siempre distintos a los de las variables libres de un mismo término.

β-reducción y η-reducción

La relación principal entre términos del conjunto Λ es la *β-reducción*. Para definirla formalmente, introducimos la noción de reemplazo de variables por términos:

Definición 1.2.6. *Sean $M_1, M_2, N \in \Lambda$, $x, y \in \mathbb{V}$ tal que $x \neq y$. Se define la operación de sustitución de una variable por un término en otro término como:*

$$\begin{aligned} x[N/x] &= N \\ y[N/x] &= y \\ (M_1 M_2)[N/x] &= M_1[N/x]M_2[N/x] \\ (\lambda y.M_1)[N/x] &= \lambda y.M_1[N/x] \qquad \text{si } y \notin \mathsf{FV}(N) \end{aligned}$$

Notar que la condición de aplicación sobre la última clausula de la definición es necesaria para el buen compotamiento de la operación de sustitución. Pero, al mismo tiempo, siempre se puede asumir su cumplimiento en virtud de la convención de variables libres, pues y se encuentra ligada en $(\lambda y.M_1)$.

De este modo, la β-reducción queda definida como:

Definición 1.2.7. *Sea $\rightarrow_\beta$ la mínima relación sobre el conjunto Λ que cumple*

$$(\lambda x.P)Q \rightarrow_\beta P[Q/x]$$

y es cerrada bajo las reglas

$$\begin{aligned} &M \rightarrow_\beta M' \Rightarrow \forall\, x \in \mathbb{V} \;\; [\lambda x.M \rightarrow_\beta \lambda x.M'] \\ &M \rightarrow_\beta M' \Rightarrow \forall\, N \in \Lambda \;\; [MN \rightarrow_\beta M'N] \\ &M \rightarrow_\beta M' \Rightarrow \forall\, N \in \Lambda \;\; [NM \rightarrow_\beta NM'] \end{aligned}$$

Los términos de la forma $(\lambda x.P)Q$ son llamados *β-redex*.

En [3] se demuestra que el λ-cálculo definido con la reducción β resulta confluente.

Adicionalmente puede incluirse la noción de *η-reducción*, la cual permite la eliminación de abstracciones redundantes. Una abstracción se considera redundante cuando su único propósito es pasar, en forma directa, su argumento a otra función.

Definición 1.2.8. *Sea $\rightarrow_\eta$ la mínima relación sobre el conjunto Λ que cumple*

$$\lambda x.(Mx) \rightarrow_\eta M$$

si $x \notin \mathsf{FV}(M)$ *y es cerrada bajo las reglas*

$$M \to_\eta M' \Rightarrow \forall\, x \in \mathbb{V}\ [\lambda x.M \to_\eta \lambda x.M']$$
$$M \to_\eta M' \Rightarrow \forall\, N \in \Lambda\ [MN \to_\eta M'N]$$
$$M \to_\eta M' \Rightarrow \forall\, N \in \Lambda\ [NM \to_\eta NM']$$

Los η-*redex* son aquellos términos de la forma $\lambda x.(Mx)$ con $x \notin \mathsf{FV}(M)$.

Esta reducción captura la idea de *extensionalidad*: dos términos funcionalmente iguales son considerados equivalentes ($=_\eta$).

Al incluir esta noción de reducción, el cálculo resultante, definido con la relación entre términos $\to_{\beta\eta} = \to_\beta \cup \to_\eta$, resulta nuevamente confluente.

Por otro lado, puede verse que la reducción η es fuertemente normalizante, pues en cada paso se disminuye el tamaño del término en cuestión, por lo que no pueden existir derivaciones infinitas de η-reducciones. Esto no es así para el caso de la β-reducción, y puede ilustrarse en el siguiente ejemplo

$$(\lambda x.xx)(\lambda x.xx) \to_\beta (\lambda x.xx)(\lambda x.xx) \to_\beta \cdots$$

Por esto resulta interesante caracterizar los términos que se encuentran en β-*forma normal*, es decir, los N tales que no existe N' que cumpla $N \to_\beta N'$. Dicha caracterización se presenta con la siguiente sintaxis

$$N ::= xN \ldots N \mid \lambda x.N$$

De este modo, el término $(\lambda x.xx)$ se encuentra en β-forma normal.

1.2.3. Lógica Combinatoria

La Lógica Combinatoria (CL) fue introducida por Curry y Schönfinkel en los años '30 [28, 12], con el objetivo de eliminar la necesidad de variables cuantificadas en la lógica matemática. Sin embargo, su uso más popular se dio en el ámbito de las ciencias de la computación, donde es utilizada como modelo computacional dado que, a pesar de su simplicidad, logra capturar la noción de función computable, al igual que el λ-cálculo.

Este formalismo puede ser interpretado como una variante de λ-cálculo, donde las abstracciones y las variables ligadas son eliminadas para dar lugar a constantes funcionales llamadas *combinadores*.

Sintaxis

El sistema consiste de un conjunto inicial de combinadores (**K** y **S**) y la aplicación entre ellos, para dar lugar a nuevos combinadores. De este modo, el conjunto *CL* de términos de CL queda definido de la siguiente manera:

$$M ::= x \mid \mathbf{K} \mid \mathbf{S} \mid MM$$

donde x es una variable de un conjunto infinito numerable $\mathbb{V}$.

Si bien existen varias interpretaciones de estos combinadores iniciales, informalmente puede verse a **K** como un generador de funciones constantes, mientras que **S** es una versión generalizada de la aplicación.

Al igual que para el λ-cálculo, se asume por convención que la aplicación de términos de *CL* es asociativa a izquierda.

Reglas de reducción

La relación de reducción de CL viene dada por las siguientes dos reglas, asociadas a los combinadores **K** y **S**:

$$\begin{aligned} \mathbf{K}PQ &\to P \\ \mathbf{S}PQR &\to PR(QR) \end{aligned}$$

De este modo se define la relación de reducción $\to_w$ como la mínima relación que resulta de la unión de estas dos reglas, y es cerrada por

$$\begin{aligned} M \to_w M' &\Rightarrow \forall\, N \in CL\ [MN \to_w M'N] \\ M \to_w M' &\Rightarrow \forall\, N \in CL\ [NM \to_w NM'] \end{aligned}$$

Propiedades

Puede demostrarse que la relación de reducción $\to_w$ es confluente [5].

Si bien en CL no existe el concepto de variable ligada, definimos la noción de *variable libre* ya que haremos referencia a ella con frecuencia, pero ésta no es otra cosa que las variables de un término.

Definición 1.2.9 (Variables libres de un *CL*-término)**.** *Sea* $\mathsf{FV}(\cdot) : CL \to \wp(\mathbb{V})$ *el siguiente mapeo de términos a conjuntos de variables:*

$$\begin{aligned} \mathsf{FV}(x) &= \{x\} \\ \mathsf{FV}(\mathbf{K}) = \mathsf{FV}(\mathbf{S}) &= \varnothing \\ \mathsf{FV}(MN) &= \mathsf{FV}(M) \cup \mathsf{FV}(N) \end{aligned}$$

En cuanto a la interpretación de CL como una variante de λ-cálculo sin variables ligadas, se debe a que puede representarse la abstracción de la siguiente manera

$$\begin{aligned} \lambda^* x.x &= \mathbf{SKK} && \\ \lambda^* x.M &= \mathbf{K}M && \text{si } x \notin \mathsf{FV}(M) \\ \lambda^* x.(Mx) &= M && \text{si } x \notin \mathsf{FV}(M) \\ \lambda^* x.(MN) &= \mathbf{S}(\lambda^* x.M)(\lambda^* x.N) && \text{si } x \in \mathsf{FV}(MN) \end{aligned}$$

Esta representación de la abstracción es tal que vale

Proposición 1.2.1. *Sean* $M, N \in CL$*. Luego,*

$$(\lambda^* x.M)N \twoheadrightarrow_w M[N/x]$$

donde $M[N/x]$ *es la sustitución de variables definida de la manera esperada.*

A partir de esto, es posible definir los siguientes mapeos entre los conjuntos Λ y CL

Definición 1.2.10 (Traducción de CL a Λ)**.** *Sea* $(\cdot)_\Lambda : CL \to \Lambda$ *el siguiente mapeo de términos de CL a términos de* λ*-cálculo.*

$$\begin{aligned} x_\Lambda &= x \\ \mathbf{K}_\Lambda &= \lambda xy.x \\ \mathbf{S}_\Lambda &= \lambda xyz.\big(xz(yz)\big) \\ (MN)_\Lambda &= M_\Lambda N_\Lambda \end{aligned}$$

Definición 1.2.11 (Traducción de Λ a CL)**.** *Sea* $(\cdot)_{CL} : \Lambda \to CL$ *el siguiente mapeo de términos de* λ*-cálculo a términos de CL.*

$$\begin{aligned} x_{CL} &= x \\ (MN)_{CL} &= M_{CL} N_{CL} \\ (\lambda x.M)_{CL} &= \lambda^* x.M_{CL} \end{aligned}$$

Una vez definidas las traducciones de un conjunto de términos al otro, puede probarse la siguiente propiedad:

Propiedad 1.2.1. *Sea* $M \in CL$*. Luego,* $(M_\Lambda)_{CL} = M$*.*

Es decir que la traducción a CL es la pseudo-inversa de la traducción a λ-cálculo. Pero es fácil ver que no vale la relación en el otro sentido, pues, por ejemplo

$$\big(\lambda xy.(xy)\big)_{CL} = \lambda^* xy.(xy) = \lambda^* x.x = \mathbf{SKK}$$

y al traducir de vuelta a λ-cálculo, se tiene

$$\Big(\big(\lambda xy.(xy)\big)_{CL}\Big)_\Lambda = (\mathbf{SKK})_\Lambda = \Big(\lambda xyz.\big(xz(yz)\big)\Big)(\lambda xy.x)(\lambda xy.x)$$

Más aún, se tiene

$$(\mathbf{SKK})_\Lambda \twoheadrightarrow_\beta \lambda x.x \neq_\beta \lambda xy.(xy)$$

Pero si se tiene en cuenta la η-reducción, se tiene

$$(\mathbf{SKK})_\Lambda =_{\beta\eta} \lambda xy.(xy)$$

Cabe destacar que existen otras definiciones para λ^* que no incluyen la cláusula

$$\lambda^* x.(Mx) = M \qquad \text{si } x \notin \mathsf{FV}(M)$$

como se prensenta en [3]. Sin embargo, la Propiedad 1.2.1 no se cumpliría si se utiliza dicha definición, y la necesitaremos a lo largo de la tesis.

Por otro lado, vale la siguiente proposición:

Proposición 1.2.2. *Sean* $M, N \in CL$ *tal que* $M \to_w N$*. Luego,* $M_\Lambda \xrightarrow{+}_\beta N_\Lambda$*.*

Lo que muestra que es posible simular cualquier cómputo realizado en CL con al menos un paso de reducción en λ-cálculo.

1.2.4. Sistemas de tipos para λ-cálculo

El λ-cálculo considerado hasta el momento es lo que se conoce como *type-free*, pues sus funciones no tienen dominio e imagen fijos, en contraposición con la noción matemática de función. Desde el punto de vista computacional esto se refleja en que sus funciones no poseen tipos, es decir que se permite tomar una función arbitraria y aplicarle cualquier argumento que se desee.

Para restringir este hecho y verificar la "buena formación" de los términos del cálculo se presenta la noción de *sistema de tipos* aplicada a λ-cálculo. Ésta fue introducida independientemente por Curry [12] y Church [8] dando lugar a dos grandes familias de sistemas: los tipados *à la* Curry y los tipados *à la* Church.

Antes de formalizar las diferencias entre los distintos sistemas de tipos, introducimos los conceptos que son comunes a todos ellos.

Los tipos son términos de naturaleza sintáctica que pueden ser asignados a ciertos términos del cálculo. Llamaremos $\mathcal{T}$ al conjunto de términos de tipo, el cuál tendrá una definición específica en cada uno de los sistemas que presentamos a continuación, pero será inmediato identificar a cuál de ellas nos referimos por contexto.

Este conjunto $\mathcal{T}$ se encuentra típicamente definido sobre un conjunto infinito numerable $\mathcal{V}$ de variables de tipo, el cual asumiremos igual para todos los formalismos. Por convención, notaremos a los elementos de $\mathcal{V}$ con las primeras letras del alfabeto griego $(\alpha, \beta, \gamma \ldots)$, mientras que usaremos σ, τ, ρ, etc. para referirnos a elementos arbitrarios del conjunto $\mathcal{T}$.

Definición 1.2.12.

1. *Sean $M \in \Lambda$ y $\sigma \in \mathcal{T}$, se define* statement *como $M : \sigma$ y se dice que M tiene tipo σ. Se llama a M el* sujeto *y a σ el* predicado *del statement.*

2. *Una* declaración *es un statement cuyo sujeto es una variable.*

3. *Una* base *es un conjunto de declaraciones con sujetos distintos dos a dos.*

En la literatura se llama *contexto* a un conjunto de declaraciones, sin distinguir si los sujetos de estas últimas son distintos o no. En adelante asumiremos que siempre lo son, de modo que llamaremos indistintamente a contextos y bases.

Definición 1.2.13. *Sean Γ un contexto, $M \in \Lambda$ y $\sigma \in \mathcal{T}$. Se define* juicio de tipado *como*

$$\Gamma \vdash M : \sigma$$

y se dice que M tiene tipo σ en el contexto Γ.

Todo sistema de tipos consiste de una serie de reglas que permiten inferir juicios de tipado, de modo que diremos que un juicio es *derivable* en cierto sistema siempre que exista una secuencia de aplicaciones de dichas reglas que permita concluir el juicio en cuestión.

Cuando un juicio de tipado derivable es de la forma $\vdash M : \sigma$, es decir, existe un término M que tiene tipo σ en el contexto vacio, diremos que σ es un tipo *habitado* en el sistema de tipos.

Para diferenciar entre los juicios de tipado pertenecientes a los distintos sistemas tratados, subindexaremos el operador $\vdash$ de modo que sea claro a qué sistema nos referimos en cada caso. Por ejemplo, $\vdash_{\lambda\to}$ se refiere al λ-cálculo simplemente tipado ($\boldsymbol{\lambda}\!\to$).

Una última definición que nos será de utilidad es la de los siguiente operadores sobre contextos de tipado:

Definición 1.2.14. *Sea* $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ *una base.*

1. *Se nota* $\mathrm{dom}(\Gamma) = \{x_1, \ldots, x_n\}$ *y* $\Gamma(x_i) = \sigma_i$. *Es decir, se considera* Γ *una función parcial.*

2. *Sea* V_0 *un conjunto de variables. Luego* $\Gamma\restriction_{V_0} = \big\{x : \sigma \;\big|\; x \in V_0,\ \sigma = \Gamma(x)\big\}$.

A continuación presentamos las diferentes familias de sistemas de tipos, introduciendo para ellas los principales sistemas a los cuales haremos referencia a lo largo de la tesis. Para algunos de estos sistemas existen formulaciones en ambas familias ($\boldsymbol{\lambda}\!\to$, $\boldsymbol{\lambda 2}$), y hay una relación entre sus dos versiones, la cual presentamos una vez introducidos ambos formalismos.

Tipado *à la* Curry

Esta familia de sistemas se caracteriza por usar el mismo conjunto de términos que el λ-cálculo clásico, extendiendo el formalismo con un conjunto de tipos $\mathcal{T}$ y una serie de reglas de inferencia que permiten asignar tipo a distintas expresiones del lenguaje, dando lugar a lo que se conoce como el conjunto de términos tipables.

Este enfoque da lugar al paradigma de programación conocido como *tipado implícito*, donde no es necesario declarar el tipo de las variables o datos que intervienen en un programa dentro del mismo término. De este modo, es el compilador el encargado de decidir si puede o no asignarse un tipo a dicha expresión y, en caso afirmativo, se dice que el programa es correcto.

A continuación presentamos tres de los principales sistemas de tipos *à la* Curry, junto con sus propiedades principales, a las cuales haremos referencias a los largo del desarrollo de la tesis. Estos son: el sistema de tipos simples ($\boldsymbol{\lambda}\!\to$), tipos intersección ($\boldsymbol{\lambda}\cap$) y tipos de segundo orden ($\boldsymbol{\lambda 2}$).

λ-cálculo simplemente tipado

El sistema de tipos simples fue introducido originalmente por Curry [12], realizando su planteo sobre la Teoría de Combinadores. En [13, 14] el sistema fue modificado y formalizado sobre λ-cálculo. Es considerado el mínimo sistema de tipos razonable, pues sólo consiste de un constructor de tipos funcionales ($\to$), el cual interpretamos como asociativo a derecha por convención.

Se define el conjunto de tipos $\mathcal{T}$ con la siguiente sintaxis:

$$\tau ::= \alpha \mid \tau \to \tau$$

donde $\alpha \in \mathcal{V}$.

El conjunto de reglas de inferencia para deducir juicios de tipado es el siguiente:

$$\frac{}{\Gamma, x : \sigma \vdash_{\lambda\to} x : \sigma} \ (axiom)$$

$$\frac{\Gamma, x : \sigma \vdash_{\lambda\to} M : \tau}{\Gamma \vdash_{\lambda\to} \lambda x.M : \sigma \to \tau} \ (\to I) \qquad \frac{\Gamma \vdash_{\lambda\to} M : \sigma \to \tau \quad \Gamma \vdash_{\lambda\to} N : \sigma}{\Gamma \vdash_{\lambda\to} MN : \tau} \ (\to E)$$

donde se permite el tipado de variables mediante la asunción del tipo al incluirlo en el contexto, se construye el tipado de abstracciones (funciones) eliminando la variable ligada del contexto de tipado, y se permite la aplicación de un término a otro (parámetro y función respectivamente) siempre y cuando el tipo del dominio de la función coincida con el tipo del parámetro.

Notar que los tipos σ y τ a los que se hace referencia en las reglas son tipos arbitrarios, por lo que, en rigor, se está definiendo un conjunto de esquemas de reglas. Luego, para cada σ y τ se tiene una regla distinta del sistema de tipos. De aquí en adelante se hará esta asunción para todos los sistemas de tipos involucrados en la tesis, y se llamará reglas de tipado a estos esquemas de reglas, para mayor simplicidad en la lectura.

A modo ilustrativo, veamos que la siguiente derivación es válida:

$$\begin{array}{ll} x : \sigma \vdash_{\lambda\to} x : \sigma & \text{por } (axiom) \\ \vdash_{\lambda\to} \lambda x.x : \sigma \to \sigma & \text{por } (\to I) \end{array}$$

notar que para cada $\sigma \in \mathcal{T}$ se tiene un juicio de tipado diferente para el término $(\lambda x.x)$.

Pueden probarse diferentes propiedades para el sistema $\boldsymbol{\lambda}\!\to$, algunas de las cuales presentaremos más adelante, ya que son comunes a todos los sistemas de tipado *à la* Curry.

Introducimos a continuación el lema de generación para $\boldsymbol{\lambda}\!\to$, tal como se lo presenta en [4]:

Lema 1.2.1 (Generation Lemma)**.**

1. $\Gamma \vdash_{\lambda\to} x : \sigma \Rightarrow (x : \sigma) \in \Gamma$.
2. $\Gamma \vdash_{\lambda\to} MN : \tau \Rightarrow \exists\, \sigma \ [\Gamma \vdash_{\lambda\to} M : \sigma \to \tau \quad y \quad \Gamma \vdash_{\lambda\to} N : \sigma]$.
3. $\Gamma \vdash_{\lambda\to} \lambda x.M : \rho \Rightarrow \exists\, \sigma, \tau \ [\Gamma, x : \sigma \vdash_{\lambda\to} M : \tau \quad y \quad \rho \equiv \sigma \to \tau]$.

Este lema permite deducir la procedencia de un juicio de tipado dado en función del término al cuál se le está asignando tipo.

Tipos intersección

El sistema $\boldsymbol{\lambda}\cap$ de tipos intersección es conocido también como *Torino System*, ya que fue desarrollado originalmente es dicha ciudad italiana por Coppo y Dezani-Ciancaglini [10] entre otros.

Este sistema hace posible la asignación, a una variable x, de dos tipos σ y τ al mismo tiempo. Esto presenta una forma de polimorfismo, que permite asignar a un término un número finito de tipos a la vez, en contraste con el sistema $\boldsymbol{\lambda 2}$ que presentaremos a continuación, en donde el polimorfismo es paramétrico.

El conjunto de tipos $\mathcal{T}$ se define a partir de la sintaxis:

$$\tau ::= \alpha \mid \tau \to \tau \mid \tau \cap \tau$$

donde nuevamente $\alpha \in \mathcal{V}$.

Existen distintos planteos del sistema en la bibliografía, resultando todos ellos equivalentes. A continuación presentamos la variante utilizada por Bunder en [7] para el análisis de los subsistemas de $\boldsymbol{\lambda}\cap$:

$$\frac{}{\Gamma, x : \sigma \vdash_{\lambda\cap} x : \sigma} \ (axiom)$$

$$\frac{\Gamma, x : \sigma \vdash_{\lambda\cap} M : \tau}{\Gamma \vdash_{\lambda\cap} \lambda x.M : \sigma \to \tau} \ (\to I) \qquad \frac{\Gamma \vdash_{\lambda\cap} M : \sigma \to \tau \quad \Gamma \vdash_{\lambda\cap} N : \sigma}{\Gamma \vdash_{\lambda\cap} MN : \tau} \ (\to E)$$

$$\frac{\Gamma \vdash_{\lambda\cap} M : \sigma \quad \Gamma \vdash_{\lambda\cap} M : \tau}{\Gamma \vdash_{\lambda\cap} M : \sigma \cap \tau} \ (\cap I) \qquad \frac{\Gamma \vdash_{\lambda\cap} M : \sigma \cap \tau}{\Gamma \vdash_{\lambda\cap} M : \sigma \quad \Gamma \vdash_{\lambda\cap} M : \tau} \ (\cap E)$$

$$\frac{\Gamma \vdash_{\lambda\cap} \lambda x.(Mx) : \sigma \quad x \notin \mathsf{FV}(M)}{\Gamma \vdash_{\lambda\cap} M : \sigma} \ (\eta)$$

Esta axiomatización extiende el sistema $\boldsymbol{\lambda}\to$ con la posibilidad de introducir y eliminar intersecciones (polimorfismo) y el tipado de η-reductos a través de la regla (η).

En $\boldsymbol{\lambda}\cap$ puede derivarse el siguiente juicio de tipado para el término $(\lambda x.xx)$, que no es tipable en $\boldsymbol{\lambda}\to$:

$$\begin{aligned}
&x : (\sigma \to \tau) \cap \sigma \vdash_{\lambda\cap} x : \sigma \to \tau && \text{por } (\cap E)\\
&x : (\sigma \to \tau) \cap \sigma \vdash_{\lambda\cap} x : \sigma && \text{por } (\cap E)\\
&x : (\sigma \to \tau) \cap \sigma \vdash_{\lambda\cap} xx : \tau && \text{por } (\to E)\\
&\vdash_{\lambda\cap} (\lambda x.xx) : \big((\sigma \to \tau) \cap \sigma\big) \to \tau && \text{por } (\to I)
\end{aligned}$$

Alternativamente, se presenta el sistema con una noción de subtipado, reemplazando las reglas $(\cap E)$ y (η) por:

$$\frac{\Gamma \vdash_{\lambda\cap} M : \sigma \quad \sigma \leq \tau}{\Gamma \vdash_{\lambda\cap} M : \tau} \ (\leq)$$

con la relación $\leq$ definida de la siguiente manera:

$$\begin{array}{rl}
(\text{refl}) & \sigma \leq \sigma \\
(\text{dummy}) & \sigma \leq \sigma \cap \sigma \\
(\text{inst}) & \sigma \cap \tau \leq \sigma \quad \sigma \cap \tau \leq \tau \\
(\text{distr}) & (\sigma \to \tau) \cap (\sigma \to \rho) \leq \sigma \to \tau \cap \rho \\
(\text{trans}) & \sigma \leq \tau \leq \rho \Rightarrow \sigma \leq \rho \\
(\to) & \sigma \leq \sigma', \tau \leq \tau' \Rightarrow \sigma' \to \tau \leq \sigma \to \tau' \\
(\cap) & \sigma \leq \sigma', \tau \leq \tau' \Rightarrow \sigma \cap \tau \leq \sigma' \cap \tau'
\end{array}$$

Entre estas reglas, se destacan (inst) y (distr) que codifican la eliminación y distribución de intersecciones. También se tiene la propiedad de *contravarianza* en el dominio y *covarianza* en el codominio para los tipos funcionales gracias a la regla $(\to)$. Se permite la combinación de subtipos con la regla $(\cap)$, como así también la intersección de un tipo con sí mismo utilizando (dummy). Por último se tiene la reflexividad (refl) y transitividad (trans) de la relación de subtipado.

En [4] se presenta otra axiomatización para esta relación de subtipado, al mismo tiempo que existen varios planteos distintos en la literatura. Todos ellos resultan equivalentes al aquí presentado, que fue tomado de [7].

Este sistema de tipos despierta particular interés pues caracteriza la clase de los términos fuertemente normalizantes [4]. Es decir, los términos tipables en $\boldsymbol{\lambda\cap}$, con el contexto de tipado vacío, son exactamente los términos que representan a las funciones recursivas totales.

Alternativamente, puede extenderse el sistema agregando al conjunto de términos una variable distinguida o constante ω, de modo que todo término sea tipable con dicha constante. Para esto basta agregar la regla de tipado:

$$\frac{}{\Gamma \vdash_{\lambda\cap} M : \omega} \ (\omega)$$

y las siguientes premisas a la relación $\leq$:

$$\begin{array}{rl}
(\omega) & \sigma \leq \omega \\
(\omega\to) & \omega \leq \omega \to \omega
\end{array}$$

De este modo, el tipo ω es una suerte de *Top* del que todos los términos del conjunto $\mathcal{T}$ son subtipo. A lo largo del documento, nos referiremos al sistema de tipos intersección con la constante ω como $\boldsymbol{\lambda\cap^{\omega}}$.

Tipado de segundo orden

El sistema $\boldsymbol{\lambda 2}$, también conocido como *System F*, fue introducido independientemente por Girard [18] y Reynolds [27]. La formulación original se realizó en el paradigma de tipado *à la* Church, siendo planteado en su versión *à la* Curry

tiempo después. La motivación de Girard provenía de la Teoría de la Demostración, mientras que Reynolds buscaba estudiar el polimorfismo en lenguajes de programación.

El conjunto $\mathcal{T}$ para **λ2** queda definido con la sintaxis

$$\tau ::= \alpha \mid \tau \to \tau \mid \forall \alpha.\tau$$

donde nuevamente $\alpha \in \mathcal{V}$. De este modo, se introduce la posibilidad de cuantificar sobre las variables de tipo.

Para introducir formalmente las reglas del sistema es necesario presentar primero la noción de *variable libre* sobre los tipos, y la operación de sustitución de una variable por un tipo:

Definición 1.2.15 (Variables libres de un $\mathcal{T}$-término)**.** *Sea $\mathcal{FV}(\cdot) : \mathcal{T} \to \wp(\mathcal{V})$ el siguiente mapeo de tipos a conjuntos de variables de tipo:*

$$\begin{aligned} \mathcal{FV}(\alpha) &= \{\alpha\} \\ \mathcal{FV}(\sigma \to \tau) &= \mathcal{FV}(\sigma) \cup \mathcal{FV}(\tau) \\ \mathcal{FV}(\forall \alpha.\sigma) &= \mathcal{FV}(\sigma) - \{\alpha\} \end{aligned}$$

Al igual que para el λ-cálculo, diremos que una variable se encuentra libre cuando está fuera del alcance de un ligador, en este caso, los cuantificadores ∀. Esta definición se hace extensiva a contextos de tipado de la siguiente manera:

$$\mathcal{FV}(\Gamma) = \bigcup_{x:\sigma \in \Gamma} \mathcal{FV}(\sigma)$$

Presentamos ahora la operación de sustitución, que permite reemplazar todas las apariciones libres de una variable por un tipo:

Definición 1.2.16. *Sean $\sigma, \tau, \rho \in \mathcal{T}$, $\alpha, \beta \in \mathcal{V}$ tal que $\alpha \neq \beta$. Se define la operación de sustitución de una variable de tipo por un tipo en otro tipo como:*

$$\begin{aligned} \alpha[\alpha := \tau] &= \tau \\ \beta[\alpha := \tau] &= \beta \\ (\sigma \to \rho)[\alpha := \tau] &= \sigma[\alpha := \tau] \to \rho[\alpha := \tau] \\ (\forall \beta.\sigma)[\alpha := \tau] &= \forall \beta.\sigma[\alpha := \tau] \qquad \textit{si } \beta \notin \mathcal{FV}(\tau) \end{aligned}$$

La idea de sustitución es análoga a la presentada para los términos del λ-cálculo, pues se permite que la operación se propague dentro de un cuantificador siempre que la variable ligada sea distinta a la variable a reemplazar y no aparezca libre en el término que se introduce. Esta última condición puede suponerse válida en la mayoría de los casos, pues trabajaremos bajo una convención de variables libres semejante a la introducida para los términos del conjunto Λ, de modo que se asume que los nombres de las variables libres de un término son siempre distintos a los de las variables ligadas.

Por último presentamos el siguiente lema sobre la operación de sustitución, que permite intercambiar el orden en que se realizan varias sustituciones sucesivas, bajo ciertas condiciones:

Lema 1.2.2. *Sean $\sigma, \tau, \rho \in \mathcal{T}$, $\alpha, \beta \in \mathcal{V}$ tal que $\alpha \neq \beta$.*

$$\alpha \notin \mathcal{FV}(\rho) \Rightarrow \sigma[\alpha := \tau][\beta := \rho] = \sigma[\beta := \rho]\big[\alpha := \tau[\beta := \rho]\big]$$

Con las nociones de variable libre y sustitución definidas, introducimos el conjunto de reglas de inferencia para el sistema **λ2**:

$$\frac{}{\Gamma, x : \sigma \vdash_{\boldsymbol{\lambda 2}} x : \sigma} \ (axiom)$$

$$\frac{\Gamma, x : \sigma \vdash_{\boldsymbol{\lambda 2}} M : \tau}{\Gamma \vdash_{\boldsymbol{\lambda 2}} \lambda x.M : \sigma \to \tau} \ (\to I) \qquad \frac{\Gamma \vdash_{\boldsymbol{\lambda 2}} M : \sigma \to \tau \quad \Gamma \vdash_{\boldsymbol{\lambda 2}} N : \sigma}{\Gamma \vdash_{\boldsymbol{\lambda 2}} MN : \tau} \ (\to E)$$

$$\frac{\Gamma \vdash_{\boldsymbol{\lambda 2}} M : \sigma \quad \alpha \notin \mathcal{FV}(\Gamma)}{\Gamma \vdash_{\boldsymbol{\lambda 2}} M : \forall \alpha.\sigma} \ (\forall I) \qquad \frac{\Gamma \vdash_{\boldsymbol{\lambda 2}} M : \forall \alpha.\sigma}{\Gamma \vdash_{\boldsymbol{\lambda 2}} M : \sigma[\alpha := \tau]} \ (\forall E)$$

donde se extiende **λ**→ con la posibilidad de ligar una variable en un tipo inferido, siempre y cuando ésta no sea parte de una asunción en el contexto de tipado, y la opción de reemplazar una variable cuantificada universalmente por cualquier término del conjunto $\mathcal{T}$.

De este modo, puede derivarse:

$$\vdash_{\boldsymbol{\lambda 2}} \lambda x.x : \forall \alpha.(\alpha \to \alpha)$$

quedando el polimorfismo planteado de manera explícita, en contraposición con **λ**→, donde por cada tipo $\sigma \in \mathcal{T}$ se tiene un juicio de tipado diferente de la forma:

$$\vdash_{\boldsymbol{\lambda}\to} \lambda x.x : \sigma \to \sigma$$

Propiedades principales de los sistemas *à la* Curry

A continuación presentamos una serie de lemas y propiedades que son comunes a los sistemas de tipado *à la* Curry introducidos anteriormente.

El siguiente lema permite relacionar los elementos del contexto con el término de un juicio de tipado dado:

Lema 1.2.3 (Base Lemma)**.** *Sea Γ una base.*

1. *Si $\Delta \supseteq \Gamma$ es otra base, entonces $\Gamma \vdash M : \delta \Rightarrow \Delta \vdash M : \delta$.*
2. $\Gamma \vdash M : \delta \Rightarrow \mathsf{FV}(M) \subseteq \mathrm{dom}(\Gamma)$.
3. $\Gamma \vdash M : \delta \Rightarrow \Gamma\restriction_{\mathsf{FV}(M)} \vdash M : \delta$.

El lema de sustitución permite analizar cómo afectan las sustituciones, tanto de términos como de tipos, sobre los juicios de tipado.

Lema 1.2.4 (Substitution Lemma)**.**

1. $\Gamma \vdash M : \sigma \Rightarrow \Gamma[\alpha := \tau] \vdash M : \sigma[\alpha := \tau]$.

2. *Si* $\Gamma, x : \sigma \vdash M : \tau$ *y* $\Gamma \vdash N : \sigma$, *entonces* $\Gamma \vdash M[N/x] : \tau$.

Las siguientes dos propiedades son deseables en todo sistema de tipos. Éstas son: la clausura bajo reducción y la propiedad de normalización fuerte de los términos tipables. La primera nos dice que el tipo de un programa (término) se mantiene a lo largo de su ejecución (reducción), mientras que la última asegura que todo programa correcto termina.

Proposición 1.2.3 (Subject Reduction)**.** *Sean* $M, M' \in \Lambda$ *tal que* $M \twoheadrightarrow_\beta M'$. *Entonces,*

$$\Gamma \vdash M : \sigma \Rightarrow \Gamma \vdash M' : \sigma$$

Proposición 1.2.4 (Strong Normalization)**.** *Sean* $M \in \Lambda$. *Luego,*

$$\exists \Gamma \; \exists \, \sigma \in \mathcal{T} \;\; [\Gamma \vdash M : \sigma] \Rightarrow M \text{ es SN}$$

Tipado *à la* Church

En el paradigma de tipado *à la* Church, los programas deben ser escritos con la declaración de los tipos que involucran. Esto se conoce como *tipado explícito*, y tiene como consecuencia que los términos del cálculo tiene a lo sumo un tipo posible.

Para esto se introduce una variante en los términos de λ-cálculo, que consiste en anotarlos con la declaración de los tipos de las variables ligadas. Esto define un nuevo conjunto $\Lambda_{\mathcal{T}}$ de términos $\mathcal{T}$-anotados, o pseudo-términos, que posee una definición específica para cada sistema de tipos.

A continuación introducimos los sistemas $\boldsymbol{\lambda}\!\!\rightarrow$ y $\boldsymbol{\lambda 2}$ en su versión *à la* Church, y analizamos la relación de estos últimos con su contraparte *à la* Curry. Luego presentamos una noción más general de sistemas de tipos, conocida como *Pure Type Systems* (PTS's), en la que no se diferencia entre los términos del cálculo y los tipos, y se utilizan constantes distinguidas llamadas *sorts* para especificar diferentes sistemas en base a una axiomatización uniforme. Una vez introducidos los PTS's, formularemos lo que se conoce como el λ-*cubo*, o *cubo de Barendregt* [4]: un conjunto de ocho sistemas de tipos que son casos particulares de PTS's.

Tipado simple

Esta versión de λ-cálculo simplemente tipado ($\boldsymbol{\lambda}\!\!\rightarrow$-Church) fue introducida por Church en los años '40 [8]. El conjunto de tipos $\mathcal{T}$ utilizado es el mismo que para el sistema $\boldsymbol{\lambda}\!\!\rightarrow$-Curry, y en base a éste se extiende la sintaxis de los términos del cálculo de la siguiente manera:

$$M ::= x \mid MM \mid \lambda x : \tau . M$$

donde x es una variable de un conjunto infinito numerable $\mathbb{V}$ y $\tau \in \mathcal{T}$.

Las reglas de inferencia son completamente análogas a las vistas para el sistema *à la* Curry, presentándose una modificación en $(\rightarrow I)$, donde se construyen abstracciones con la nueva sintaxis de pseudo-términos:

$$\frac{}{\Gamma, x : \sigma \vdash_{\lambda\rightarrow} x : \sigma} \ (axiom)$$

$$\frac{\Gamma, x : \sigma \vdash_{\lambda\rightarrow} M : \tau}{\Gamma \vdash_{\lambda\rightarrow} \lambda x : \sigma.M : \sigma \rightarrow \tau} \ (\rightarrow I) \quad \frac{\Gamma \vdash_{\lambda\rightarrow} M : \sigma \rightarrow \tau \quad \Gamma \vdash_{\lambda\rightarrow} N : \sigma}{\Gamma \vdash_{\lambda\rightarrow} MN : \tau} \ (\rightarrow E)$$

De este modo, la siguiente derivación es correcta:

$$\begin{array}{ll} x : \sigma \vdash_{\lambda\rightarrow} x : \sigma & \text{por } (axiom) \\ \vdash_{\lambda\rightarrow} \lambda x : \sigma.x : \sigma \rightarrow \sigma & \text{por } (\rightarrow I) \end{array}$$

lo que lleva a que el término $(\lambda x : \sigma.x)$ tenga un único tipo, pues la variable x se declara de tipo σ en el mismo término.

Puede plantearse una función que borra las anotaciones de tipo de un pseudo--término, transformando todo juicio de tipado de **λ**→-Church en uno de **λ**→-Curry.

Definición 1.2.17. *Sea la función* $|\cdot| : \Lambda_{\mathcal{T}} \rightarrow \Lambda$ *definida de la siguiente manera:*

$$\begin{aligned} |x| &= x \\ |MN| &= |M||N| \\ |\lambda x : \sigma.M| &= \lambda x.|M| \end{aligned}$$

Con esta definición puede demostrarse la siguiente correspondencia entre ambos sistemas de tipado simple [4, 29]:

Proposición 1.2.5.

1. *Sea* $M \in \Lambda_{\mathcal{T}}$*, entonces*

$$\Gamma \vdash_{Church} M : \sigma \Rightarrow \Gamma \vdash_{Curry} |M| : \sigma$$

2. *Sea* $M \in \Lambda$*, entonces*

$$\Gamma \vdash_{Curry} M : \sigma \Rightarrow \exists\, M' \in \Lambda_{\mathcal{T}} \ \ [\Gamma \vdash_{Church} M' : \sigma \ \ y \ |M'| = M]$$

Tipado de segundo orden

Como se comentó anteriormente, los planteos originales de Girard [18] y Reynolds [27] de un sistema de tipos polimórfico para λ-cálculo se realizaron en el paradigma de tipado *à la* Church. Nuevamente el conjunto $\mathcal{T}$ posee la misma definición que para **λ2**-Curry, pero la extensión en la sintaxis de los términos es la siguiente:

$$M ::= x \mid MM \mid \lambda x : \tau.M \mid M\tau \mid \Lambda\alpha.M$$

donde $x \in \mathbb{V}$, $\alpha \in \mathcal{V}$ y $\tau \in \mathcal{T}$. Esto permite ligar variables de tipo en los mismos términos del cálculo, y aplicar un tipo a un término.

Con esta nueva definición del conjunto Λ se redefine la noción de β-reducción de modo que:

$$(\lambda x : \tau.P)Q \to_\beta P[Q/x]$$
$$(\Lambda\alpha.M)\tau \to_\beta M[\alpha := \tau]$$

donde la operación de sustitución de una variable de tipo α en un pseudo-término M, recorre las anotaciones de tipo de M reemplazando las apariciones libres de α por τ.

El conjunto de reglas de inferencia es nuevamente similar al presentado para **λ2**-Curry, teniendo en consideración la nueva definición del conjunto $\Lambda_{\mathcal{T}}$:

$$\frac{}{\Gamma, x : \sigma \vdash_{\boldsymbol{\lambda 2}} x : \sigma} \ (axiom)$$

$$\frac{\Gamma, x : \sigma \vdash_{\boldsymbol{\lambda 2}} M : \tau}{\Gamma \vdash_{\boldsymbol{\lambda 2}} \lambda x : \sigma.M : \sigma \to \tau} \ (\to I) \qquad \frac{\Gamma \vdash_{\boldsymbol{\lambda 2}} M : \sigma \to \tau \quad \Gamma \vdash_{\boldsymbol{\lambda 2}} N : \sigma}{\Gamma \vdash_{\boldsymbol{\lambda 2}} MN : \tau} \ (\to E)$$

$$\frac{\Gamma \vdash_{\boldsymbol{\lambda 2}} M : \sigma \quad \alpha \notin \mathcal{FV}(\Gamma)}{\Gamma \vdash_{\boldsymbol{\lambda 2}} \Lambda\alpha.M : \forall\alpha.\sigma} \ (\forall I) \qquad \frac{\Gamma \vdash_{\boldsymbol{\lambda 2}} M : \forall\alpha.\sigma}{\Gamma \vdash_{\boldsymbol{\lambda 2}} M\tau : \sigma[\alpha := \tau]} \ (\forall E)$$

Aquí puede verse que cada regla modifica el término en cuestión de manera única, de modo que puede deducirse fácilmente cuál fue la última regla aplicada durante la derivación de un juicio de tipado. Esto lleva a que el problema de *type-checking* de **λ2**-Church tenga una complejidad computacional más baja que el de **λ2**-Curry. Más aún, se sabe que para el primero el problema es decidible, mientras que para el último no lo es [31].

Ilustramos la inferencia de tipos del sistema con el mismo ejemplo que para **λ2**-Curry, donde se ve cómo el pseudo-término guarda la información de las reglas aplicadas:

$$\begin{aligned} x : \alpha &\vdash_{\boldsymbol{\lambda 2}} x : \alpha && \text{por } (axiom) \\ &\vdash_{\boldsymbol{\lambda 2}} \lambda x : \alpha.x : \alpha \to \alpha && \text{por } (\to I) \\ &\vdash_{\boldsymbol{\lambda 2}} \Lambda\alpha.(\lambda x : \alpha.x) : \forall\alpha.(\alpha \to \alpha) && \text{por } (\forall I) \end{aligned}$$

Al igual que para el caso del tipado simple, se plantea una función de borrado sobre los pseudo-términos:

Definición 1.2.18. *Sea la función* $|\cdot| : \Lambda_{\mathcal{T}} \to \Lambda$ *definida de la siguiente manera:*

$$\begin{aligned} |x| &= x \\ |MN| &= |M||N| \\ |\lambda x : \sigma.M| &= \lambda x.|M| \\ |M\sigma| &= |M| \\ |\Lambda\alpha.M| &= |M| \end{aligned}$$

De modo que se tiene la siguiente equivalencia entre **λ2**-Church y **λ2**-Curry [4]:

Proposición 1.2.6.

1. *Sea* $M \in \Lambda_{\mathcal{T}}$*, entonces*

$$\Gamma \vdash_{Church} M : \sigma \Rightarrow \Gamma \vdash_{Curry} |M| : \sigma$$

2. *Sea* $M \in \Lambda$*, entonces*

$$\Gamma \vdash_{Curry} M : \sigma \Rightarrow \exists\, M' \in \Lambda_{\mathcal{T}} \;\; [\Gamma \vdash_{Church} M' : \sigma \;\; y \;\; |M'| = M]$$

Pure Type Systems

La noción de *Pure Type System* fue desarrollada independientemente por Berardi y Terlouw a fines de la década del '80 (ver [4]). Desde el punto de vista histórico, los PTS's fueron introducidos como una generalización del método mediante el cual se generan los sistemas del llamado λ-*cubo*, pero para facilitar la lectura del documento decidimos introducir en primer lugar los PTS's y luego presentar al cubo como un caso particular de estos.

Estos sistemas se definen sobre un único conjunto de pseudo-términos $\Lambda_{\mathcal{T}}$, donde no hay distinción entre términos del cálculo y tipos. Éste queda definido a partir de la siguiente sintaxis:

$$\tau ::= x \mid c \mid \tau\tau \mid \lambda x : \tau.\tau \mid \Pi x : \tau.\tau$$

donde $x \in \mathbb{V}$ y c pertenece a un conjunto numerable de constantes $\mathcal{C}$.

Cabe destacar que los constructores de pseudo-términos presentados para **λ2** se encuentran codificados en esta nueva sintaxis. Por ejemplo, la aplicación de dos pseudo-términos de PTS's representa a la aplicación de dos términos de λ-cálculo y a la aplicación de un tipo a un término al mismo tiempo. En la siguiente sección veremos que los constructores de tipos $\rightarrow$ y $\forall$ son casos particulares del nuevo ligador de variables Π, mientras que Λ puede definirse en términos de λ.

Definición 1.2.19. *La* especificación *de un PTS es una tupla* $S = (\mathcal{S}, \mathcal{A}, \mathcal{R})$ *donde*

1. $\mathcal{S}$ *es un subconjunto de* $\mathcal{C}$*, llamado* sorts.

2. $\mathcal{A}$ *es un conjunto de* axiomas *de la forma* $c : s$ *con* $c \in \mathcal{C}$ *y* $s \in \mathcal{S}$.

3. $\mathcal{R}$ *es un conjunto de* reglas *de la forma* (s_1, s_2, s_3) *con* $s_1, s_2, s_3 \in \mathcal{S}$.

Definición 1.2.20. *Sea $S = (\mathcal{S}, \mathcal{A}, \mathcal{R})$ una especificación, se define el PTS determinado por S, notado $\lambda S = \lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$, con el siguiente conjunto de axiomas y reglas:*

$$\frac{}{\vdash c : s}\ (axioms) \qquad si\ (c : s) \in \mathcal{A}$$

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}\ (start) \qquad si\ x \notin \Gamma$$

$$\frac{\Gamma \vdash A : B \quad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B}\ (weakening) \qquad si\ x \notin \Gamma$$

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (\Pi x : A.B) : s_3}\ (product) \qquad si\ (s_1, s_2, s_3) \in \mathcal{R}$$

$$\frac{\Gamma \vdash F : (\Pi x : A.B) \quad \Gamma \vdash a : A}{\Gamma \vdash Fa : B[x := a]}\ (application)$$

$$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash (\Pi x : A.B) : s}{\Gamma \vdash (\lambda x : A.b) : (\Pi x : A.B)}\ (abstraction)$$

$$\frac{\Gamma \vdash A : B \quad \Gamma \vdash B' : s \quad B =_\beta B'}{\Gamma \vdash A : B'}\ (conversion)$$

con $s \in \mathcal{S}$ y $x \in \mathbb{V}$.

Los sistemas del λ-cubo

El λ-cubo forma una estructura en la que ocho sistemas de tipado *à la* Church, como **λ→** y **λ2**, son presentados de manera uniforme [4]. Los sistemas del cubo se extienden unos a otros, comenzando con **λ→**, el sistema de tipos razonable más simple, y terminando en el Cálculo de Construcciones (λC) presentado por Coquand y Huet en [11], el sistema más fuerte dentro del cubo.

Muchos de estos sistemas fueron introducidos por diferentes autores en la literatura. Si bien no son presentados en su formulación original dentro del cubo, puede verse que resultan equivalentes [4].

Los sistemas involucrados en el λ-cubo, son los siguientes:

Sistema	*Referencias*
$\boldsymbol{\lambda}\!\rightarrow$	Church [8]
$\boldsymbol{\lambda 2}$	Girard [18], Reynolds [27]
$\boldsymbol{\lambda}$P	de Bruijn [6]
$\boldsymbol{\lambda}$P$\mathbf{2}$	Longo y Moggi [21]
$\boldsymbol{\lambda}\underline{\boldsymbol{\omega}}$	Renardel de Lavalette [20]
$\boldsymbol{\lambda\omega}$	Girard [18]
$\boldsymbol{\lambda}$P$\underline{\boldsymbol{\omega}}$	
$\boldsymbol{\lambda}$P$\boldsymbol{\omega} = \boldsymbol{\lambda}$C	Coquand y Huet [11]

El cubo es introducido con una axiomatización uniforme para todos sus sistemas, que resulta un caso particular de los PTS's. En esta formulación, la especificación de cada uno de los sistemas se realiza sobre los conjuntos:

$$\mathcal{S} = \{*, \Box\} \qquad \mathcal{A} = \big\{(* : \Box)\big\}$$

mientras que el conjunto de reglas $\mathcal{R}$ posee una definición particular para cada sistema:

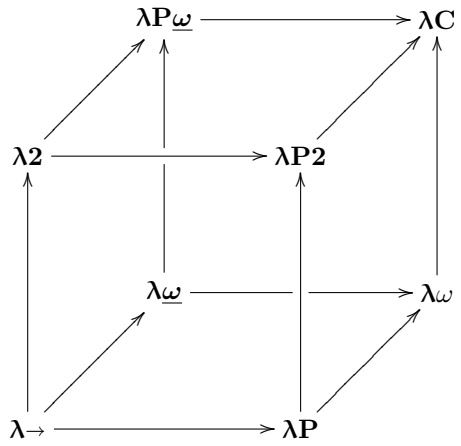
Sistema	*Conjunto de reglas* ($\mathcal{R}$)			
$\boldsymbol{\lambda}\!\rightarrow$	$(*,*)$			
$\boldsymbol{\lambda 2}$	$(*,*)$	$(\Box,*)$		
$\boldsymbol{\lambda}$P	$(*,*)$		$(*,\Box)$	
$\boldsymbol{\lambda}$P$\mathbf{2}$	$(*,*)$	$(\Box,*)$	$(*,\Box)$	
$\boldsymbol{\lambda}\underline{\boldsymbol{\omega}}$	$(*,*)$			$(\Box,\Box)$
$\boldsymbol{\lambda\omega}$	$(*,*)$	$(\Box,*)$		$(\Box,\Box)$
$\boldsymbol{\lambda}$P$\underline{\boldsymbol{\omega}}$	$(*,*)$		$(*,\Box)$	$(\Box,\Box)$
$\boldsymbol{\lambda}$P$\boldsymbol{\omega} = \boldsymbol{\lambda}$C	$(*,*)$	$(\Box,*)$	$(*,\Box)$	$(\Box,\Box)$

donde $(s_1, s_2) = (s_1, s_2, s_2)$ es una notación abreviada de las reglas definidas para los PTS's. Los elementos de $\mathcal{S}$ son denominados *type* $(*)$ y *kind* $(\Box)$, y el axioma en $\mathcal{A}$ dice que todo *type* es un *kind*.

Con estas especificaciones, las reglas de los sistemas del cubo pueden dividirse en dos grupos. El primero de ellos consiste de reglas comunes a los ocho sistemas. Éste está constituido por todas las reglas salvo (*product*), pues es la única que depende del conjunto $\mathcal{R}$. Luego, el segundo conjunto está compuesto de las reglas específicas de cada sistema, es decir, las distintas instanciaciones de (*product*) con los elementos de $\mathcal{R}$ que corresponden en cada sistema.

Este conjunto de ocho sistemas es denominado λ-cubo por la relaciones de inclusión existentes entre ellos. Si bien cada sistema tiene un conjunto de reglas propias, los sistemas se extienden los unos a los otros de modo que, por ejemplo, las reglas propias de $\boldsymbol{\lambda 2}$ incluyen a las de $\boldsymbol{\lambda}\!\rightarrow$. Estas relaciones vienen dadas por los distintos conjuntos $\mathcal{R}$, como se muestra en la tabla anterior. El

siguiente esquema es una manera gráfica de representar al λ-cubo, donde las aristas denotan la relación de inclusión entre los distintos sistemas:



El λ-cubo presenta, a su vez, axiomatizaciones alternativas para sistemas ya conocidos, como es el caso de $\boldsymbol{\lambda}\!\rightarrow$ y $\boldsymbol{\lambda 2}$. A priori puede no resultar clara la relación entre ambas axiomatizaciones, pero puede verse que los contructores de tipos $\rightarrow$ y $\forall$ se codifican en términos de Π, y el ligador de variables Λ de $\boldsymbol{\lambda 2}$, en términos de λ, en los sistemas del cubo:

$$\begin{aligned} A \rightarrow B &\equiv \Pi x : A.B \quad \text{si } x \notin \mathsf{FV}(A) \cup \mathsf{FV}(B) \\ \forall \alpha.A &\equiv \Pi \alpha : *.A \\ \Lambda \alpha.M &\equiv \lambda \alpha : *.M \end{aligned}$$

A continuación ilustramos la equivalencia entre ambas formulaciones de $\boldsymbol{\lambda}\!\rightarrow$ con una derivación para la función identidad. En primer lugar desarrollamos la inferencia para la axiomatización clásica:

$$\begin{aligned} x : \alpha \vdash_{\boldsymbol{\lambda}\rightarrow} x : \alpha & \qquad \text{por } (axiom) \\ \vdash_{\boldsymbol{\lambda}\rightarrow} (\lambda x : \alpha.x) : \alpha \rightarrow \alpha & \qquad \text{por } (\rightarrow I) \end{aligned}$$

Ahora vemos la inferencia para el sistema del λ-cubo:

$$\begin{aligned} \vdash_{\boldsymbol{\lambda}\rightarrow} * : \Box & \qquad \text{por } (axioms) \\ \alpha : * \vdash_{\boldsymbol{\lambda}\rightarrow} \alpha : * & \qquad \text{por } (start) \\ y : \alpha, \alpha : * \vdash_{\boldsymbol{\lambda}\rightarrow} y : \alpha & \qquad \text{por } (start) \\ y : \alpha, \alpha : * \vdash_{\boldsymbol{\lambda}\rightarrow} \alpha : * & \qquad \text{por } (weakening) \\ \alpha : * \vdash_{\boldsymbol{\lambda}\rightarrow} (\Pi y : \alpha.\alpha) : * & \qquad \text{por } (product) \\ x : \alpha, \alpha : * \vdash_{\boldsymbol{\lambda}\rightarrow} x : \alpha & \qquad \text{por } (start) \\ \alpha : * \vdash_{\boldsymbol{\lambda}\rightarrow} (\lambda x : \alpha.x) : (\Pi y : \alpha.\alpha) & \qquad \text{por } (abstraction) \end{aligned}$$

donde $\Pi y : \alpha.\alpha \equiv \alpha \rightarrow \alpha$.

En estos ejemplos se ve como la declaración de variables de tipos queda implicita en la formulación clásica, mientras que en el sistema del cubo se realiza de manera explicita. Esto es necesario para desambiguar, dado que, en este último, las variables de tipos y términos provienen de un mismo conjunto unificado.

Entre las propiedades principales que pueden demostrarse para los ocho sistemas del cubo, se destacan *Strong Normalization (SN)* de términos tipables y *Subject Reduction (SR)*.

Proposición 1.2.7 (Subject Reduction)**.** *Sean $M, M' \in \Lambda_{\mathcal{T}}$ tal que $M \twoheadrightarrow_\beta M'$. Entonces,*

$$\Gamma \vdash M : A \Rightarrow \Gamma \vdash M' : A$$

Proposición 1.2.8 (Strong Normalization)**.** *Sean $M \in \Lambda_{\mathcal{T}}$. Luego,*

$$\exists \Gamma\ \exists\, A \in \Lambda_{\mathcal{T}}\ \ [\Gamma \vdash M : A] \Rightarrow M \text{ es SN}$$

1.2.5. Sistemas de tipos para Lógica Combinatoria

Al igual que para λ-cálculo, pueden definirse sistemas de tipos para la lógica combinatoria. De hecho, la formulación original realizada por Curry en los años '30 [12, 13] fue hecha sobre CL, y planteada para el λ-cálculo tiempo después, vía la correspondencia que existe entre ambos formalismos.

Dicho planteo propone un sistema de tipado simple (CL→) en el que el conjunto de tipos $\mathcal{T}$ queda definido, del mismo modo que para $\boldsymbol{\lambda}\!\!\rightarrow$, por la siguiente sintaxis:

$$\tau ::= \alpha \mid \tau \to \tau$$

con α una variable de un conjunto infinito numerable $\mathcal{V}$.

Las reglas de tipado para CL→ son las siguientes:

$$\frac{}{\Gamma, x : \sigma \vdash_{\text{CL}_\rightarrow} x : \sigma}\ (axiom) \qquad \frac{}{\Gamma \vdash_{\text{CL}_\rightarrow} \mathbf{K} : \sigma \to \tau \to \sigma}\ (K)$$

$$\frac{}{\Gamma \vdash_{\text{CL}_\rightarrow} \mathbf{S} : (\sigma \to \tau \to \rho) \to (\sigma \to \tau) \to \sigma \to \rho}\ (S)$$

$$\frac{\Gamma \vdash_{\text{CL}_\rightarrow} M : \sigma \to \tau \quad \Gamma \vdash_{\text{CL}_\rightarrow} N : \sigma}{\Gamma \vdash_{\text{CL}_\rightarrow} MN : \tau}\ (\to E)$$

Al igual que el $\boldsymbol{\lambda}\!\!\rightarrow$, éste cumple con las propiedades deseables de todo un sistema de tipos como SN para términos tipables y SR, entre otras. Más aún, puede probarse que ambos sistemas resultan equivalentes bajo la siguiente noción:

Proposición 1.2.9.

1. *Sea $M \in CL$, $\Gamma \vdash_{\text{CL}_\rightarrow} M : \sigma \Rightarrow \Gamma \vdash_{\lambda\rightarrow} M_\Lambda : \sigma$.*

2. *Sea* $M \in \Lambda$, $\Gamma \vdash_{\lambda\to} M : \sigma \Rightarrow \Gamma \vdash_{\mathrm{CL}\to} M_{CL} : \sigma$.

Cabe aclarar que al pasar de un sistema al otro, no es necesario traducir los elementos del contexto y los tipos, pues ambos formalismos trabajan sobre el mismo conjunto $\mathcal{T}$, y podemos asumir que tanto λ-cálculo como CL están definidos sobre el mismo conjunto de variables $\mathbb{V}$ (notar que los elementos de Γ son de la forma $x : \sigma$ con $x \in \mathbb{V}$ y $\sigma \in \mathcal{T}$).

Como breve ejemplo de esta correspondencia, veamos que puede derivarse el tipo $\alpha \to \alpha$ para el término $(\lambda x.x)_{CL} = \mathbf{SKK}$, ya que este juicio de tipado será utilizado recurrentemente a lo largo de la tesis:

$$\begin{array}{ll}
\vdash_{\mathrm{CL}\to} \mathbf{S} : (\alpha \to (\beta \to \alpha) \to \alpha) \to (\alpha \to \beta \to \alpha) \to \alpha \to \alpha & \text{por } (S) \\
\vdash_{\mathrm{CL}\to} \mathbf{K} : \alpha \to (\beta \to \alpha) \to \alpha & \text{por } (K) \\
\vdash_{\mathrm{CL}\to} \mathbf{SK} : (\alpha \to \beta \to \alpha) \to \alpha \to \alpha & \text{por } (\to E) \\
\vdash_{\mathrm{CL}\to} \mathbf{K} : \alpha \to \beta \to \alpha & \text{por } (K) \\
\vdash_{\mathrm{CL}\to} \mathbf{SKK} : \alpha \to \alpha & \text{por } (\to E)
\end{array}$$

1.2.6. Isomorfismo de Curry–Howard

El isomorfismo de Curry–Howard establece una relación directa entre los distintos sistemas deductivos y los modelos computacionales. Fue introducido originalmente por Curry en los años '30 [12], al observar que los tipos de los combinadores podian ser interpretados como esquemas-axiomas para la lógica proposicional intuicionista (IPL) [29].

Ya en la década del '50, Curry [13] notó que el sistema de tipos simples para la lógica combinatoria coincidía con un fragmento de los sistemas deductivos *à la* Hilbert. Luego, en 1969 fue Howard [19] quien demostró que el fragmento intuicionista del sistema *Natural Deduction* (IPL) podía ser interpretado como la variante tipada del λ-cálculo.

En su formulación más general, la correspondencia de Curry–Howard relaciona diferentes sistemas de prueba lógicos con los distintos sistemas de tipos de modelos computacionales. En particular pueden distinguirse dos correspondencias. La primera de ellas es la que identifica fórmulas lógicas con tipos de programas, de modo que los axiomas (K) y (S) del sistema CL→ pueden ser vistos como los dos primeros axiomas de los sistemas deductivos presentados por Hilbert [16]:

$$\begin{aligned}
&Ax_1 : \phi \to (\psi \to \phi) \\
&Ax_2 : (\phi \to \psi \to \xi) \to (\phi \to \psi) \to \phi \to \xi
\end{aligned}$$

al mismo tiempo que $(\to E)$ es interpretada como la regla de inferencia *Modus Ponens*. Es así como puede demostrarse que toda fórmula derivable en este fragmento del sistema lógico resulta en un tipo habitado en CL→ [29]. Esta correspondencia es conocida con el nombre de *proposition-as-types interpretation* [6, 19].

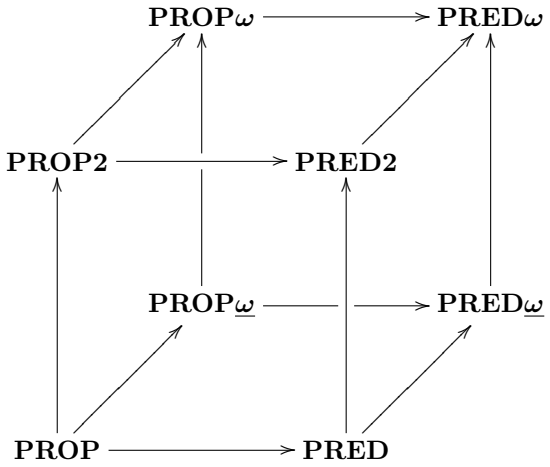
En cuanto a la segunda correspondencia, ésta considera a los juicios de tipado como codificaciones de pruebas lógicas, donde el contexto representa un conjunto

de axiomas auxiliares, el tipo es la fórmula que se demuestra y el término codifica la demostración. De este modo, un tipo habitado en $\boldsymbol{\lambda}\!\!\rightarrow$ se corresponde con una tautologia de IPL. Esta última correspondencia se sustenta en propiedades de los sistemas de tipos como el lema de generación, que permite, dado un juicio de tipado, analizar su procedencia y decodificar así la demostración de la fórmula.

Es a través del isomorfismo que pueden estudiarse propiedades de las lógicas por medio del análisis de los sistemas de tipos. En particular, la propiedad de consistencia: una lógica se dice *consistente* si no existe una fórmula ψ tal que puede derivarse $\vdash \psi$ y $\vdash \neg\psi$.

En muchas lógicas esto implica que vale $\vdash \psi$ para toda fórmula ψ, como es el caso de los sistemas estudiados en la presente tesis. Luego, la consistencia de una lógica es equivalente a la no habitabilidad de al menos un tipo en el sistema de tipos isomorfo a la misma, y surge así la idea de probar la consistencia vía SN de los términos tipables del cálculo (ver [4]).

En [4], el autor expone las correspondencias entre los sistemas del λ-cubo y ocho sistemas de lógica intuicionista, como se muestra en el siguiente gráfico:



PROP	lógica proposicional
PROP2	lógica proposicional de segundo orden
PROP$\underline{\omega}$	lógica proposicional de alto orden débil
PROPω	lógica proposicional de alto orden
PRED	lógica predicativa
PRED2	lógica predicativa de segundo orden
PRED$\underline{\omega}$	lógica predicativa de alto orden débil
PREDω	lógica predicativa de alto orden

Capítulo 2

Consistencia de sistemas de tipos para λ-cálculo

La consistencia de una lógica es una propiedad crucial. Al definir un nuevo sistema de tipos para algún lenguaje, es deseable que se cumpla esta propiedad. Por otro lado, existen hoy en día varias formulaciones de λ-cálculo y sus lógicas asociadas a través del isomorfismos de Curry–Howard, que son objeto de estudio.

En la literatura, las demostraciones de consistencia de varias lógicas se relacionan con fuertes propiedades del λ-cálculo asociado: *Subject Reduction (SR)*, *Strong Normalization (SN)* de términos tipables, y lemas de generación. Todas estas propiedades lidian con los sistemas de tipos de los cálculos.

En este capítulo, presentamos un método para probar la consistencia basado en diferentes variantes de la noción de *variable positiva* [1]. Con la definición precisa, se muestra que esta técnica es adecuada para obtener demostraciones puramente sintácticas, de gran simplicidad al compararlas con las conocidas en la literatura.

Veremos que los distintos sistemas de tipos poseen diferencias. La ventaja de las pruebas desarrolladas a continuación consiste en el hecho de que no involucran semántica, utilizando sólo los constructores y conectivos provistos por el lenguaje. Esto es útil ya que deja de lado las interpretaciones como una pregunta independiente, dado que muchas veces la semántica no está disponible o puede ser muy difícil de definir formalmente.

Además de los sistemas de tipos usuales, derivados del Cubo de Barendregt [4] y los tipos intersección, estamos interesados en subsistemas que resultan de considerar distintos subconjuntos de sus reglas. En [7] el autor considera subsistemas del sistema de tipos $\boldsymbol{\lambda\cap}$, haremos lo propio para el sistema $\boldsymbol{\lambda 2}$.

En la siguiente sección motivamos nuestra idea e ilustramos su funcionamiento.

2.1. Tipado simple

Presentamos a continuación la prueba de consistencia clásica para el fragmento implicativo de la lógica intuicionista (IPL), para ilustrar la complejidad de la misma. Es una pequeña variante de la demostración de [29], la cual prueba la consistencia de la IPL a partir de $\boldsymbol{\lambda}\!\!\rightarrow$ (via el isomorfismo de Curry–Howard), haciendo uso de propiedades no triviales del sistema como SN, SR, etc.:

Proposición 2.1.1 (Consistencia de IPL/$\boldsymbol{\lambda}\!\!\rightarrow$)**.** *Existe un tipo σ tal que para ningún término M se tiene $\vdash_{\lambda\rightarrow} M : \sigma$.*

Demostración. Tomamos $\sigma = \alpha \in \mathcal{V}$. Supongamos que existe un término M tal que $\vdash_{\lambda\rightarrow} M : \alpha$. Luego, por la Proposición 1.2.4 de SN para términos tipables, existe un término N en β-forma normal tal que $M \twoheadrightarrow_\beta N$. Luego, por SR (Proposición 1.2.3), $\vdash_{\lambda\rightarrow} N : \alpha$. Por la caracterización de formas normales de λ-cálculo, N tiene la siguiente forma:

1. $N = xN_1 \dots N_n$ con $n \geq 0$, $x \in \mathbb{V}$ y N_i una forma normal para cada $1 \leq i \leq n$.
2. $N = \lambda x.N'$ con N' una forma normal.

El caso (1) queda descartado por el Lema 1.2.3 (Base Lemma), pues $x \notin \mathrm{dom}(\varnothing)$, por lo que sólo aplica el caso (2). Pero por Lema 1.2.1 (Generation Lemma para $\boldsymbol{\lambda}\!\!\rightarrow$-Curry), existen tipos σ y τ tal que $\alpha = \sigma \rightarrow \tau$, lo cual es absurdo pues α es una variable de tipo.

□

A continuación introducimos la noción de variable positiva para el sistema simplemente tipado. Ésta surge de la interpretación intuitiva de los tipos flecha ($\rightarrow$) como espacios de funciones, por lo que puede ser vista como las conclusiones que no se ven igualadas a premisas que las cancelan.

La definición tiene en cuenta la inclusión opcional de constantes entre los tipos, para las cuales su conjunto de variables positivas es vacío.

Definición 2.1.1 (Variables positivas para IPL/$\boldsymbol{\lambda}\!\!\rightarrow$)**.** *Sea $\mathcal{PV}(\cdot) : \mathcal{T} \rightarrow \wp(\mathcal{V})$ el siguiente mapeo de tipos a conjuntos de variables de tipo (fórmulas y variables proposicionales respectivamente):*

$$\begin{aligned} \mathcal{PV}(c) &= \varnothing \\ \mathcal{PV}(\alpha) &= \{\alpha\} \\ \mathcal{PV}(\sigma \rightarrow \tau) &= \mathcal{PV}(\tau) - \mathcal{PV}(\sigma) \end{aligned}$$

Se definen las variables positivas de un contexto como $\mathcal{PV}(\Gamma) = \bigcup_{x:\sigma\in\Gamma} \mathcal{PV}(\sigma)$.

Como una utilidad adicional, podemos incluir opcionalmente la siguiente regla:

$$\frac{}{\Gamma \vdash_{\lambda\rightarrow} c : \sigma_c} \ (const)$$

donde el término constante c tiene tipo σ_c (fijo) y asumimos para toda constante c del lenguaje, $\mathcal{PV}(\sigma_c) = \varnothing$. Esta asunción es requerida para el primer caso de la siguiente prueba.

Proposición 2.1.2 (Positive Invariance)**.** *Si* $\Gamma \vdash_{\lambda\to} M : \sigma$*, entonces* $\mathcal{PV}(\sigma) \subseteq \mathcal{PV}(\Gamma)$.

Demostración. Por inducción en la prueba de $\Gamma \vdash_{\lambda\to} M : \sigma$, analizando la última regla aplicada:

Caso 1: La última regla aplicada fue $(const)$, lo que implica $\sigma \equiv \sigma_c$. Luego,

$$\mathcal{PV}(\sigma_c) = \varnothing \subseteq \mathcal{PV}(\Gamma)$$

Caso 2: Si se aplicó $(axiom)$, $\Gamma = \Gamma', x : \sigma$. Luego,

$$\mathcal{PV}(\sigma) \subseteq \mathcal{PV}(\Gamma') \cup \mathcal{PV}(\sigma) = \mathcal{PV}(\Gamma)$$

Caso 3: Si se aplicó $(\to I)$, entonces $M \equiv \lambda x.M'$, $\sigma \equiv \tau \to \rho$ y vale

$$\Gamma, x : \tau \vdash_{\lambda\to} M' : \rho$$

Luego, por hipótesis inductiva:

$$\begin{gathered}\mathcal{PV}(\rho) \subseteq \mathcal{PV}(\Gamma, x : \tau) = \mathcal{PV}(\Gamma) \cup \mathcal{PV}(\tau) \\ \Rightarrow \mathcal{PV}(\rho) - \mathcal{PV}(\tau) \subseteq \big(\mathcal{PV}(\Gamma) \cup \mathcal{PV}(\tau)\big) - \mathcal{PV}(\tau) \\ \Rightarrow \mathcal{PV}(\tau \to \rho) \subseteq \mathcal{PV}(\Gamma)\end{gathered}$$

Caso 4: Si se aplicó $(\to E)$, entonces $M \equiv M_1 M_2$ y vale

$$\Gamma \vdash_{\lambda\to} M_1 : \tau \to \sigma \quad \text{y} \quad \Gamma \vdash_{\lambda\to} M_2 : \tau$$

Luego, por hipótesis inductiva:

$$\begin{gathered}\mathcal{PV}(\tau \to \sigma) \subseteq \mathcal{PV}(\Gamma) \quad \text{y} \quad \mathcal{PV}(\tau) \subseteq \mathcal{PV}(\Gamma) \\ \Rightarrow \mathcal{PV}(\sigma) \subseteq \big(\mathcal{PV}(\sigma) - \mathcal{PV}(\tau)\big) \cup \mathcal{PV}(\tau) \subseteq \mathcal{PV}(\Gamma)\end{gathered}$$

□

La Proposición 2.1.2 es la propiedad principal de preservación de las variables positivas durante la deducción. Como corolario inmediato se tiene la consistencia:

Corolario 2.1.1 (Consistencia de IPL/$\boldsymbol{\lambda}\to$)**.** *Existe un tipo* σ *tal que para ningún término* M *se tiene* $\vdash_{\lambda\to} M : \sigma$.

Demostración. Sea $\sigma = \alpha \in \mathcal{V}$. Luego, si $\vdash_{\lambda\to} M : \alpha$, $\mathcal{PV}(\alpha) = \{\alpha\} \subseteq \varnothing$, lo que es absurdo.

□

Remarcamos el hecho de que se trata de un resultado conocido, pero el método de demostración utilizado es considerablemente más simple que los presentados en la literatura. Su simplicidad radica en que no utiliza ninguna propiedad adicional del sistema de tipos, como es el caso de SN y SR en la prueba original, cuyas demostraciones suponen una elevada complejidad. En particular, las pruebas conocidas de SN requieren el uso de la técnica de los candidatos de reducibilidad [4].

Por otro lado, en el caso anterior, se presentó una variante del método que contempla el uso de tipos primitivos. Estos son adecuados para tipar términos constantes, los cuales pueden estar presentes en diferentes lenguajes donde el método sea aplicable. Esto mismo puede argumentarse en los sistemas de tipos que se estudian a continuación, por lo que se presentan variantes sin tipos primitivos y constantes, para mayor simplicidad.

2.2. Tipos intersección

En la presente sección desarrollamos dos pruebas de consistencia para el sistema de tipos $\boldsymbol{\lambda\cap}$, analizando sus dos posibles axiomatizaciones. Esto es, incluyendo la relación de subtipado y la regla $(\leq)$, o reemplazando a esta última por las reglas $(\cap E)$ y (η).

De este modo ilustramos cómo la complejidad de la prueba de consistencia con el método de variables positivas depende directamente de la axiomatización usada.

En ambos casos utilizamos la siguiente definición de variables positivas:

Definición 2.2.1 (Variables positivas para $\boldsymbol{\lambda\cap}$)**.** *Sea $\mathcal{PV}(\cdot) : \mathcal{T} \to \wp(\mathcal{V})$ el siguiente mapeo de tipos a conjuntos de variables de tipo:*

$$\begin{aligned} \mathcal{PV}(\alpha) &= \{\alpha\} \\ \mathcal{PV}(\sigma \to \tau) &= \mathcal{PV}(\tau) - \mathcal{PV}(\sigma) \\ \mathcal{PV}(\sigma \cap \tau) &= \mathcal{PV}(\sigma) \cup \mathcal{PV}(\tau) \end{aligned}$$

Se definen las variables positivas de un contexto de la manera esperada.

En primera instancia desarrollamos la prueba para el sistema con la relación de subtipado, para lo cual se requiere el siguiente lema:

Lema 2.2.1. *Si $\sigma \leq \tau$, entonces $\mathcal{PV}(\tau) \subseteq \mathcal{PV}(\sigma)$.*

Demostración. Por inducción en la prueba de $\sigma \leq \tau$, analizando cada regla.

(refl) y (dummy): Trivialmente se tiene

$$\mathcal{PV}(\sigma \cap \sigma) = \mathcal{PV}(\sigma) \subseteq \mathcal{PV}(\sigma)$$

(inst): Nuevamente es facil ver que

$$\mathcal{PV}(\sigma) \subseteq \mathcal{PV}(\sigma) \cup \mathcal{PV}(\tau) = \mathcal{PV}(\sigma \cap \tau)$$

Del mismo modo,

$$\mathcal{PV}(\tau) \subseteq \mathcal{PV}(\sigma \cap \tau)$$

(distr): Por definición

$$\begin{aligned}\mathcal{PV}\big((\sigma \to \tau) \cap (\sigma \to \rho)\big) &= \big(\mathcal{PV}(\tau) - \mathcal{PV}(\sigma)\big) \cup \big(\mathcal{PV}(\rho) - \mathcal{PV}(\sigma)\big) \\ &= \big(\mathcal{PV}(\tau) \cup \mathcal{PV}(\rho)\big) - \mathcal{PV}(\sigma) \\ &= \mathcal{PV}\big(\sigma \to (\tau \cap \rho)\big)\end{aligned}$$

(trans): Por hipótesis inductiva

$$\mathcal{PV}(\tau) \subseteq \mathcal{PV}(\sigma) \quad \text{y} \quad \mathcal{PV}(\rho) \subseteq \mathcal{PV}(\tau)$$

Luego,

$$\mathcal{PV}(\rho) \subseteq \mathcal{PV}(\sigma)$$

(→): Por hipótesis inductiva

$$\mathcal{PV}(\sigma') \subseteq \mathcal{PV}(\sigma) \quad \text{y} \quad \mathcal{PV}(\tau') \subseteq \mathcal{PV}(\tau)$$

Luego,

$$\begin{aligned}\mathcal{PV}(\sigma \to \tau') &= \mathcal{PV}(\tau') - \mathcal{PV}(\sigma) \\ &\subseteq \mathcal{PV}(\tau') - \mathcal{PV}(\sigma') \\ &\subseteq \mathcal{PV}(\tau) - \mathcal{PV}(\sigma') \\ &= \mathcal{PV}(\sigma' \to \tau)\end{aligned}$$

(∩): Por hipótesis inductiva

$$\mathcal{PV}(\sigma') \subseteq \mathcal{PV}(\sigma) \quad \text{y} \quad \mathcal{PV}(\tau') \subseteq \mathcal{PV}(\tau)$$

Luego,

$$\begin{aligned}\mathcal{PV}(\sigma' \cap \tau') &= \mathcal{PV}(\sigma') \cup \mathcal{PV}(\tau') \\ &\subseteq \mathcal{PV}(\sigma) \cup \mathcal{PV}(\tau) \\ &= \mathcal{PV}(\sigma \cap \tau)\end{aligned}$$

□

Una vez demostrado el Lema 2.2.1, estamos en condiciones de probar la propiedad de invarianza.

Proposición 2.2.1 (Positive Invariance)**.** *Si* $\Gamma \vdash_{\lambda\cap} M : \sigma$*, entonces* $\mathcal{PV}(\sigma) \subseteq \mathcal{PV}(\Gamma)$.

Demostración. Por inducción en la prueba de $\Gamma \vdash_{\lambda\cap} M : \sigma$, analizando la última regla aplicada. Los casos de las reglas $(axiom)$, $(\to I)$ y $(\to E)$ son análogos a los desarrollados en la prueba de la Proposición 2.1.2, por lo que sólo se muestran las reglas faltantes.

Caso 1: Si se aplicó $(\cap I)$, entonces vale $\Gamma \vdash_{\lambda\cap} M : \tau$, $\Gamma \vdash_{\lambda\cap} M : \rho$ y $\sigma \equiv \tau \cap \rho$. Luego, por hipótesis inductiva

$$\mathcal{PV}(\tau) \subseteq \mathcal{PV}(\Gamma) \quad \text{y} \quad \mathcal{PV}(\rho) \subseteq \mathcal{PV}(\Gamma)$$

lo que implica

$$\mathcal{PV}(\tau) \cup \mathcal{PV}(\rho) = \mathcal{PV}(\tau \cap \rho) \subseteq \mathcal{PV}(\Gamma)$$

Caso 2: Si se aplicó $(\leq)$, se tiene $\Gamma \vdash_{\lambda\cap} M : \tau$ y $\tau \leq \sigma$ y por hipótesis inductiva

$$\mathcal{PV}(\tau) \subseteq \mathcal{PV}(\Gamma)$$

Luego, por Lema 2.2.1

$$\mathcal{PV}(\sigma) \subseteq \mathcal{PV}(\Gamma)$$

□

Nuevamente la consistencia es corolario inmediato de la propiedad de invarianza, del mismo modo que para $\boldsymbol{\lambda}\!\!\rightarrow$.

Corolario 2.2.1 (Consistencia de $\boldsymbol{\lambda}\cap$)**.** *Existe un tipo σ tal que para ningún término M se tiene $\vdash_{\lambda\cap} M : \sigma$.*

Demostración. Se aplica el mismo argumento que en el caso del sistema $\boldsymbol{\lambda}\!\!\rightarrow$.
□

A continuación presentamos la prueba de la Proposición 2.2.1 considerando el sistema $\boldsymbol{\lambda}\cap$ con las reglas $(\cap E)$ y (η) en reemplazo de $(\leq)$.

Demostración. Por inducción en la prueba de $\Gamma \vdash_{\lambda\cap} M : \sigma$, analizando la última regla aplicada. Sólo desarrollamos los casos no analizados hasta el momento.

Caso 1: Si se aplicó $(\cap E)$, entonces vale $\Gamma \vdash_{\lambda\cap} M : \sigma \cap \tau$. Luego, por hipótesis inductiva

$$\mathcal{PV}(\tau \cap \sigma) = \mathcal{PV}(\tau) \cup \mathcal{PV}(\sigma) \subseteq \mathcal{PV}(\Gamma)$$

lo que implica

$$\mathcal{PV}(\sigma) \subseteq \mathcal{PV}(\Gamma) \quad \text{y} \quad \mathcal{PV}(\tau) \subseteq \mathcal{PV}(\Gamma)$$

Caso 2: Si se aplicó (η), el tipo no se modifica, por lo que directamente de la hipótesis inductiva se tiene

$$\mathcal{PV}(\sigma) \subseteq \mathcal{PV}(\Gamma)$$

□

Esta última prueba se incluye a modo ilustrativo, para enfatizar el hecho de que las demostraciones son puramente sintácticas y, por lo tanto, dependen en gran nivel de la forma en que se plantee el sistema. Puede verse que esta última es mucho más corta y sencilla que la presentada en primera instancia, ya que no requiere de la demostración de ningún lema auxiliar.

Además, como se comentó en la sección anterior, ambas demostraciones pueden extenderse fácilmente para contemplar la inclusión de tipos constantes, por lo que se tiene también la consistencia del sistema de tipos intersección completo $\boldsymbol{\lambda}\cap^{\boldsymbol{\omega}}$ (incluyendo la constante ω). Cabe destacar que, en caso de incluir $(\leq)$, se deben contemplar las reglas que involucran a ω en la demostración del Lema 2.2.1.

En [4] se presenta una definición de la relación de subtipado diferente a la usada en este trabajo, pero puede probarse fácilmente que son relaciones equivalentes y el Lema 2.2.1 vale para ambas.

También puede verse que el resultado de consistencia es aplicable a todos los subsistemas de $\boldsymbol{\lambda}\cap^{\boldsymbol{\omega}}$ presentados en [7], ya que la prueba considera las reglas por separado.

Remarcamos que la consistencia de $\boldsymbol{\lambda}\cap^{\boldsymbol{\omega}}$ no es un resultado trivial. De hecho, un problema muy relacionado, como es el de decibilidad de la habitabilidad de un tipo, estuvo abierto por más de 20 años [4, 7].

2.3. Tipado de segundo orden

En la presente sección consideramos dos subsistemas propios de $\boldsymbol{\lambda 2}$ para los cuales el método es aplicable en forma independiente, pero al unificarlos y obtener nuevamente el sistema completo no es inmediato ver que la definiciones propuestas sean adecuadas para la prueba de consistencia.

Las restricciones provienen de considerar el sistema sin las reglas $(\forall E)$ y $(\forall I)$, y las llamaremos $\boldsymbol{\lambda 2}^i$ y $\boldsymbol{\lambda 2}^e$ respectivamente. Notar que si se eliminan ambas reglas al mismo tiempo se obtiene el sistema $\boldsymbol{\lambda}\!\!\rightarrow$, al cual ya hemos analizado.

Para cada subsistema motivamos y definimos su propia noción de variable positiva.

2.3.1. El sistema $\boldsymbol{\lambda 2}^i$

En primer lugar analizamos $\boldsymbol{\lambda 2}^i$. Éste no permite la eliminación de cuantificadores universales, por lo que, simplemente interpretando a estos últimos como negadores de variables, puede demostrarse la propiedad de invarianza.

Definición 2.3.1 (Variables positivas para $\boldsymbol{\lambda 2}^i$)**.** *Sea $\mathcal{PV}(\cdot) : \mathcal{T} \to \wp(\mathcal{V})$ el siguiente mapeo de tipos a conjuntos de variables de tipo:*

$$\begin{aligned} \mathcal{PV}(\alpha) &= \{\alpha\} \\ \mathcal{PV}(\sigma \to \tau) &= \mathcal{PV}(\tau) - \mathcal{PV}(\sigma) \\ \mathcal{PV}(\forall \alpha.\sigma) &= \mathcal{PV}(\sigma) - \{\alpha\} \end{aligned}$$

Se definen las variables positivas de un contexto de la manera esperada.

Proposición 2.3.1 (Positive Invariance para $\mathbf{\lambda 2}^i$)**.** *Si* $\Gamma \vdash_i M : \sigma$*, entonces* $\mathcal{PV}(\sigma) \subseteq \mathcal{PV}(\Gamma)$*.*

Demostración. Por inducción en la prueba de $\Gamma \vdash_i M : \sigma$, analizando la última regla aplicada. Desarrollamos sólo el caso de la regla $(\forall I)$:

Se tiene $\Gamma \vdash_i M : \sigma$ y $\alpha \notin \mathcal{FV}(\Gamma)$. Luego, por hipótesis inductiva

$$\mathcal{PV}(\sigma) \subseteq \mathcal{PV}(\Gamma)$$

de lo que se deduce

$$\mathcal{PV}(\forall \alpha.\sigma) = \mathcal{PV}(\sigma) - \{\alpha\} \subseteq \mathcal{PV}(\Gamma)$$

□

Notar que la definición anterior no es apropiada si se incluye la regla $(\forall E)$, pues la siguiente inferencia sería valida

$$x : \forall \alpha.\alpha \vdash x : \beta$$

aplicando $(\forall E)$ con $\sigma = \alpha$ y $\tau = \beta$, pero al considerar las variables positivas se obtiene

$$\mathcal{PV}(\beta) = \{\beta\} \not\subseteq \varnothing = \mathcal{PV}(\forall \alpha.\alpha)$$

Desde el punto de vista general, al intentar probar el caso de la regla $(\forall E)$ en la demostración de la propiedad de invarianza, si bien se tendría por hipótesis inductiva

$$\mathcal{PV}(\sigma) - \{\alpha\} \subseteq \mathcal{PV}(\Gamma)$$

nada puede afirmarse sobre $\mathcal{PV}(\tau)$.

2.3.2. El sistema $\mathbf{\lambda 2}^e$

Ahora analizamos el subsistema $\mathbf{\lambda 2}^e$. Éste permite la eliminación de cuantificadores universales, pero no su introducción. Es así que, al menos el primer cuantificador eliminado debe provenir del contexto, ya que no pudo ser introducido de otra manera. Por esto, encontramos apropiada la siguiente definición de variables positivas:

Definición 2.3.2 (Variables positivas para $\mathbf{\lambda 2}^e$)**.** *Sea* $\mathcal{PV}(\cdot) : \mathcal{T} \to \wp(\mathcal{V})$ *el siguiente mapeo de tipos a conjuntos de variables de tipo:*

$$\begin{aligned} \mathcal{PV}(\alpha) &= \{\alpha\} \\ \mathcal{PV}(\sigma \to \tau) &= \mathcal{PV}(\tau) - \mathcal{PV}(\sigma) \\ \mathcal{PV}(\forall \alpha.\sigma) &= \bigcup_{\tau \in \mathcal{T}} \mathcal{PV}\big(\sigma[\alpha := \tau]\big) \end{aligned}$$

Las variables positivas de un contexto se definen igual que en los casos anteriores.

Esta definición tiene en cuenta la posibilidad de que una variable ligada sea positiva y considera el caso en que ésta sea reemplazada por cualquier elemento de $\mathcal{T}$, quedando como único negador de variables el constructor de tipos funcionales.

De este modo, puede demostrarse la propiedad de preservación de variables positivas a lo largo de la derivación.

Proposición 2.3.2 (Positive Invariance para $\boldsymbol{\lambda 2}^e$)**.** *Si* $\Gamma \vdash_e M : \sigma$*, entonces* $\mathcal{PV}(\sigma) \subseteq \mathcal{PV}(\Gamma)$.

Demostración. Por inducción en la prueba de $\Gamma \vdash_e M : \sigma$, analizando la última regla aplicada. Presentamos solamente el caso de la regla $(\forall E)$:

Se tiene $\Gamma \vdash_e M : \forall\alpha.\sigma$. Luego, por hipótesis inductiva

$$\mathcal{PV}(\forall\alpha.\sigma) = \bigcup_{\rho\in\mathcal{T}} \mathcal{PV}\big(\sigma[\alpha := \rho]\big) \subseteq \mathcal{PV}(\Gamma)$$

por lo que, en particular, vale

$$\mathcal{PV}\big(\sigma[\alpha := \tau]\big) \subseteq \mathcal{PV}(\Gamma)$$

□

Ambos sistemas representan restricciones propias de la lógica proposicional intuicionista de segundo orden (PROP**2**). La consistencia de estos últimos se obtiene como corolario inmediato de las Proposiciones 2.3.1 y 2.3.2 respectivamente.

Corolario 2.3.1 (Consistencia de $\boldsymbol{\lambda 2}^i$ y $\boldsymbol{\lambda 2}^e$)**.** *Existe un tipo* σ *tal que para ningún término* M *se tiene* $\vdash_{i,e} M : \sigma$.

Demostración. Se aplica el mismo argumento que para los sistemas analizados anteriormente.

□

Cabe destacar que al identificar subsistemas de un sistema de tipos, se debe verificar que las propiedades relevantes se mantengan en éstos, como por ejemplo la condición de clausura. En [7] se consideran distintos subsistemas de $\boldsymbol{\lambda}\cap$, pero no se menciona tal condición. La siguiente proposición justifica el uso de los subsistemas aquí planteados:

Proposición 2.3.3. *Los sistemas de tipos* $\boldsymbol{\lambda 2}^i$ *y* $\boldsymbol{\lambda 2}^e$ *son cerrados bajo reducción, i.e. satisfacen* Subject Reduction.

Demostración. Ver Apéndice A. □

Esto implica que los subsistemas considerados son apropiados desde el punto de vista computacional, ya que el tipo de un programa (término) se mantiene a lo largo de la ejecución (reducción).

En el Apéndice A también se muestra que el los subsistemas considerados en esta sección tienen un poder expresivo propio, en cuanto a los juicios de tipado derivables en cada uno.

Al analizar el sistema completo (**λ2**) con la definición anteriormente propuesta para $\mathbf{\lambda 2}^e$, todo parece indicar que la propiedad de invarianza de mantiene. Aunque en una primera lectura puede resultar poco intuitivo pues, a priori, de $\mathcal{PV}(\sigma) \subseteq \mathcal{PV}(\Gamma)$ no puede deducirse $\mathcal{PV}(\forall\alpha.\sigma) \subseteq \mathcal{PV}(\Gamma)$.

Sin embargo, en un análisis más detallado de la regla $(\forall I)$, la condición $\alpha \notin \mathcal{FV}(\Gamma)$ resulta ser de gran utilidad y es gracias a ésta que se eliminan los casos patológicos. Una prueba formal de la propiedad requeriría de la demostración de una serie de lemas previos, con los cuales poder relacionar las nociones de variables libres y positivas de un término, y extender este concepto a contextos de tipado. En particular no vale la inclusión de las variables positivas en las libres, ni viceversa, forzando a plantear una relacion no clara entre ambas nociones:

$$\mathcal{PV}(\forall\alpha.\alpha) = \mathbb{V} \not\subseteq \varnothing = \mathcal{FV}(\forall\alpha.\alpha)$$
$$\mathcal{PV}(\alpha \to \alpha) = \varnothing \not\supseteq \{\alpha\} = \mathcal{FV}(\alpha \to \alpha)$$

La demostración buscada fue planteada de forma parcial, es decir que no contamos con una versión completa de la misma, ya que reviste un nivel de dificultad inesperado. Si bien creemos que la propiedad vale y el método es aplicable al sistema **λ2** con la misma definición que lo es en $\mathbf{\lambda 2}^e$, la complejidad de la prueba parece mayor a las obtenidas con otros métodos de demostración, como el expuesto para la Proposición 2.1.1.

2.3.3. Consistencia de λ2

Por último presentamos una prueba de consistencia al estilo clásico, para mostrar que el sistema es efectivamente consistente. En la literatura hay pocas demostraciones de esta propiedad para **λ2**-Curry. En el caso del sistema *à la* Church, puede verse como un corolario inmediato de la Proposición 5.2.31 de [4] y de la propiedad de SN de términos tipables. En este caso, damos una prueba específica para el sistema, al estilo de la presentada para la Proposición 2.1.1.

La demostración hace uso de las propiedades de SN y SR de **λ2**-Curry, como así también de la siguiente definición y lema presentados en [4]:

Definición 2.3.3.

1. *Sean* $\sigma, \tau \in \mathcal{T}$. $\sigma > \tau$ *si*

$$\tau \equiv \forall\alpha.\sigma, \text{ para algún } \alpha \in \mathcal{V}$$

o bien

$$\sigma \equiv \forall\alpha.\sigma_1 \text{ y } \tau \equiv \sigma_1[\alpha := \pi], \text{ para algún } \pi \in \mathcal{T}$$

2. $\geq$ *es la clausura reflexiva y transitiva de* $>$.

3. *Sea* $(\cdot)^o : \mathcal{T} \to \mathcal{T}$ *la función de borrado de un tipo definida como:*

$$\begin{aligned}\alpha^o &= \alpha \\ (\sigma \to \tau)^o &= \sigma \to \tau \\ (\forall \alpha.\sigma)^o &= \sigma^o\end{aligned}$$

Lema 2.3.1 (Generation Lemma)**.**

1. $\Gamma \vdash_{\boldsymbol{\lambda 2}} x : \sigma \Rightarrow \exists\, \sigma' \geq \sigma \;\; \big[(x : \sigma') \in \Gamma\big]$.
2. $\Gamma \vdash_{\boldsymbol{\lambda 2}} MN : \tau \Rightarrow \exists \sigma \exists\, \tau' \geq \tau \;\; [\Gamma \vdash_{\boldsymbol{\lambda 2}} M : \sigma \to \tau' \quad y \quad \Gamma \vdash_{\boldsymbol{\lambda 2}} N : \sigma]$.
3. $\Gamma \vdash_{\boldsymbol{\lambda 2}} \lambda x.M : \rho \Rightarrow \exists\, \sigma, \tau \;\; [\Gamma, x : \sigma \vdash_{\boldsymbol{\lambda 2}} M : \tau \quad y \quad \sigma \to \tau \geq \rho]$.

Antes de dar la prueba de consistencia de **λ2** probamos los siguientes lemas auxiliares:

El siguiente lema nos permite ver que la sustitución aplicada a un tipo no puede eliminar un constructor de tipo funcional ya existente, o, visto de otro modo, que si el borrado del tipo sustituido es una variable, entonces el borrado del tipo original también lo es:

Lema 2.3.2. *Sean* $\sigma, \tau \in \mathcal{T}, \alpha \in \mathcal{V}$. $\big(\sigma[\alpha := \tau]\big)^o \in \mathcal{V} \Rightarrow \sigma^o \in \mathcal{V}$.

Demostración. Por inducción en σ, usando la convención de variables libres.

Caso 1: $\sigma \equiv \beta \in \mathcal{V}$. Como $\sigma^o = \beta$, la implicación vale trivialmente, ya sea $\beta = \alpha$ o no.

Caso 2: $\sigma \equiv \sigma_1 \to \sigma_2$. Entonces se tiene

$$\sigma[\alpha := \tau] = (\sigma_1 \to \sigma_2)[\alpha := \tau] = \sigma_1[\alpha := \tau] \to \sigma_2[\alpha := \tau]$$

Luego

$$\big(\sigma[\alpha := \tau]\big)^o = \sigma_1[\alpha := \tau] \to \sigma_2[\alpha := \tau] \notin \mathcal{V}$$

por lo que la implicación vale trivialmente.

Caso 3: $\sigma \equiv \forall \beta.\sigma_1$. Por convención de variables libres, podemos asumir $\beta \neq \alpha$ y $\beta \notin \mathcal{FV}(\tau)$. Luego

$$\sigma[\alpha := \tau] = (\forall \beta.\sigma_1)[\alpha := \tau] = \forall \beta.\sigma_1[\alpha := \tau]$$

Entonces, por hipótesis inductiva

$$\big(\sigma[\alpha := \tau]\big)^o = \big(\forall \beta.\sigma_1[\alpha := \tau]\big)^o = \big(\sigma_1[\alpha := \tau]\big)^o \in \mathcal{V} \Rightarrow \sigma^o = \sigma_1^o \in \mathcal{V}$$

□

Los siguientes dos lemas muestran que si un tipo es mayor o igual a una variable, entonces su borrado está en $\mathcal{V}$. Primero lo hacemos para la definición de $>$ y luego lo extendemos a su clausura reflexiva-transitiva:

Lema 2.3.3. *Sean $\sigma, \tau \in \mathcal{T}$. $\tau^o \in \mathcal{V} \wedge \sigma > \tau \Rightarrow \sigma^o \in \mathcal{V}$.*

Demostración. Si $\sigma > \tau$, por definición

$$\tau \equiv \forall\alpha.\sigma, \text{ para algún } \alpha \in \mathcal{V} \tag{2.1}$$

o bien

$$\sigma \equiv \forall\alpha.\sigma_1 \text{ y } \tau \equiv \sigma_1[\alpha := \pi], \text{ para algún } \pi \in \mathcal{T} \tag{2.2}$$

Si aplica (2.1) se tiene $\sigma^o = (\forall\alpha.\sigma)^o = \tau^o \in \mathcal{V}$.
Si es el caso de (2.2), por Lema 2.3.2 vale

$$\big(\sigma_1[\alpha := \pi]\big)^o = \tau^o \in \mathcal{V} \Rightarrow \sigma_1^o \in \mathcal{V}$$

Luego

$$\sigma^o = (\forall\alpha.\sigma_1)^o = \sigma_1^o \in \mathcal{V}$$

□

Lema 2.3.4. *Sean $\sigma \in \mathcal{T}, \alpha \in \mathcal{V}$. $\sigma \geq \alpha \Rightarrow \sigma^o \in \mathcal{V}$.*

Demostración. Por definición, si $\sigma \geq \alpha$ entonces

$$\sigma = \alpha, \text{ por reflexividad} \tag{2.3}$$

o bien

$$\exists\, c_1, \ldots, c_n \in \mathcal{T}\ \left[\sigma = c_n > \cdots > c_1 > \alpha\right], \text{ por transitividad} \tag{2.4}$$

Si vale (2.3), entonces $\sigma = \alpha \Rightarrow \sigma^o = \alpha \in \mathcal{V}$.
Si aplica (2.4), veamos que $\exists\, c_1, \ldots, c_n \in \mathcal{T}\ \left[\sigma = c_n > \cdots > c_1 > \alpha\right] \Rightarrow \sigma^o \in \mathcal{V}$, por inducción en n:

$n = 1$**:** $\sigma = c_1 > \alpha$. Como $\alpha^o = \alpha \in \mathcal{V}$, por Lema 2.3.3, $\sigma^o \in \mathcal{V}$.

$n = k + 1$**:** $\sigma = c_{k+1} > c_k > \cdots > c_1 > \alpha$. Por hipótesis inductiva, $c_k^o \in \mathcal{V}$. Luego, por Lema 2.3.3, $c_k^o \in \mathcal{V} \wedge \sigma > c_k \Rightarrow \sigma^o \in \mathcal{V}$.

□

Ahora pasamos a probar la consistencia de **λ2**:

Proposición 2.3.4 (Consistencia de **λ2**)**.** *Existe un tipo σ tal que para ningún término M se tiene $\vdash_{\lambda 2} M : \sigma$.*

Demostración. Supongamos $\vdash_{\lambda 2} M : \alpha$, con $\alpha \in \mathcal{V}$. Por la propiedad de SN para términos tipables (Proposición 1.2.4), existe N β-forma normal tal que $M \twoheadrightarrow_\beta N$. Además, por la Proposición 1.2.3 (SR), $\vdash_{\lambda 2} N : \alpha$. Por caracterización de las formas normales se tiene:

1. $N = xN_1 \dots N_n$ con $n \geq 0$, $x \in \mathbb{V}$ y N_i una forma normal para cada $1 \leq i \leq n$.
   Lo cual es absurdo, pues $x \in \mathsf{FV}(xN_1 \dots N_n)$ y $x \notin \mathrm{dom}(\varnothing)$.

2. $N = \lambda x.N'$ con N' una forma normal.
   Por Lema 2.3.1 (Generation Lemma),

   $$\vdash_{\boldsymbol{\lambda 2}} \lambda x.N' : \alpha \Rightarrow \exists\, \sigma, \tau \;\; [x : \sigma \vdash_{\boldsymbol{\lambda 2}} N' : \tau \quad \text{y} \quad \sigma \to \tau \geq \alpha]$$

   pero por Lema 2.3.4

   $$\sigma \to \tau \geq \alpha \Rightarrow (\sigma \to \tau)^o = \sigma \to \tau \in \mathcal{V}$$

   lo que nuevamente es absurdo.

El absurdo viene de suponer $\vdash M : \alpha$, por lo que se pueba la consistencia de **λ2**-Curry.

□

Con estos resultados tenemos evidencia de que el método propuesto parece no tener la simplicidad que se suponía en un principio. En particular, vimos que al considerar subsistemas propios de **λ2** podía encontrarse una definición intuitiva y adecuada del conjunto de variables positivas de un término, de modo que la propiedad de invarianza se satisfaga inmediatamente y teniendo como corolario la consistencia de dichas restricciones. Pero, al intentar aplicar estas definiciones al sistema completo, se ve que una de ellas no resulta adecuada, mientras que la otra trae aparejada dificultad.

Atribuimos este hecho a que se está trabajando con un sistema de tipos de orden mayor a los considerados hasta el momento, que posee un poder expresivo claramente superior desde el punto de vista semántico.

2.4. Tipado *à la* Church

Hasta el momento hemos considerado los sistemas de tipos *à la* Curry, como los define el autor en [4]. Para los sistemas **λ**→ y **λ2** existen sus correspondientes versiones *à la* Church, para las cuales valen las Proposiciones 1.2.5 y 1.2.6 respectivamente. De estos resultados se deduce automáticamente la consistencia de ambos sistemas. De hecho es fácil verificar que el método de variables positivas es aplicable en **λ**→-Church del mismo modo que se ilustro para **λ**→-Curry, y que se presentan los mismos inconvenientes al intentar utilizarlo para **λ2**-Church, pues la correspondencia entre los sistemas es regla a regla.

A continuación estudiaremos la aplicación del método para los sistemas de tipos *à la* Church presentados en [4], más precisamente para el caso del λ-cubo.

En este planteo, los conjuntos de términos y de tipos se encuentran unificados en un único conjunto de pseudo-términos $\Lambda_{\mathcal{T}}$, por lo que en adelante no haremos distinción entre éstos al momento de referirnos a las variables libres o definir las

variables positivas de dichos términos. Es así que usaremos, de ahora en más, la notación FV y PV sobre ambas clases de términos, en lugar de $\mathcal{FV}$ y $\mathcal{PV}$ sobre los tipos.

Por otro lado, es necesario modificar el enunciado de la propiedad de consistencia al trabajar con PTS's, pues en los mismos las declaraciones de variables se realizan de manera explicita. Esto lleva a definir como términos *habitados* en un PTS $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ a aquellos $B \in \Lambda_{\mathcal{T}}$ para los cuales vale $\Gamma \vdash A : B$ con $A \in \Lambda_{\mathcal{T}}$ y Γ un contexto que consiste sólo de (cero o más) declaraciones de variables de la forma $x : s$ con $x \in \mathbb{V}$ y $s \in \mathcal{S}$. Luego, un PTS dado es *consistente* si existe al menos un término $B \in \Lambda_{\mathcal{T}}$ no habitado.

Los sistemas del λ-cubo, entre los que se incluyen $\boldsymbol{\lambda}\!\!\rightarrow$ y $\boldsymbol{\lambda 2}$, se presentan como casos particulares de PTS's. En esta axiomatización uniforme, se tiene un conjunto de reglas comunes a los ocho sistemas, y luego una serie de reglas específicas. Estas últimas consisten en una versión parametrizada de la introducción del constructor Π:

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (\Pi x : A.B) : s_2}$$

donde $s_1, s_2 \in \{*, \Box\}$ son constantes distinguidas de los sistemas llamadas *sorts*.

Como se comentó anteriormente, el método define las variables positivas de un tipo constante como el conjunto vacío, por lo que, en principio, se podría decir que estas reglas específicas de cada sistema no tiene incidencia en la demostración de la consistencia. Luego, bastaría analizar la reglas generales, comunes a los ocho sistemas del cubo.

Entre estas seis reglas restantes, se pueden identificar tres que requieren un análisis en detalle antes de definir la noción de variable positiva para el cubo. Puede verificarse fácilmente que las otras tres reglas no presentan mayores inconvenientes a la hora de demostrar la propiedad de invarianza.

Las reglas a destacar son:

$$\frac{\Gamma \vdash F : (\Pi x : A.B) \quad \Gamma \vdash a : A}{\Gamma \vdash Fa : B[x := a]} \ (application)$$

$$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash (\Pi x : A.B) : s}{\Gamma \vdash (\lambda x : A.b) : (\Pi x : A.B)} \ (abstraction)$$

$$\frac{\Gamma \vdash A : B \quad \Gamma \vdash B' : s \quad B =_\beta B'}{\Gamma \vdash A : B'} \ (conversion)$$

con $s \in \{*, \Box\}$ un *sort*.

En el caso de (*conversion*) la dificultad viene dada por el hecho de que las condiciones $\Gamma \vdash B' : s$ y $B =_\beta B'$ no permiten asegurar nada sobre las variables positivas de B'.

Para analizar las reglas (*application*) y (*abstraction*), recordamos que puede definirse el operador $\rightarrow$ en términos de Π:

$$A \rightarrow B \equiv \Pi x : A.B \quad \text{si } x \notin \mathsf{FV}(A) \cup \mathsf{FV}(B)$$

De este modo, dichas reglas codifican a $(\rightarrow E)$ y $(\rightarrow I)$ respectivamente, tal como se las plantea en $\boldsymbol{\lambda}\!\rightarrow$. Esto motiva la definición de variables positivas del constructor Π como:

$$\mathsf{PV}(\Pi x : A.B) = \mathsf{PV}(B) - \mathsf{PV}(A) - \{x\}$$

Pero, por otro lado, pueden definirse los operadores $\forall$ y Λ a partir de Π y λ de la siguiente manera:

$$\forall \alpha.A \equiv \Pi \alpha : *.A$$
$$\Lambda \alpha.M \equiv \lambda \alpha : *.M$$

Lo que muestra que (*application*) y (*abstraction*) a su vez codifican a $(\forall E)$ y $(\forall I)$ de $\boldsymbol{\lambda 2}$. Esto, junto con la definición de $\rightarrow$ en términos de Π, respalda la idea planteada para el sistema $\boldsymbol{\lambda 2}^i$, de interpretar a los cuantificadores como negadores de variables. Pero, como se vio anteriormente, esta definición de variables positivas trae problemas al analizar la regla $(\forall E)$ o, en este caso, (*application*).

Del análisis realizado hasta el momento, podemos decir que la aplicación del método a los sistemas del cubo parece (al igual que para $\boldsymbol{\lambda 2}$) no ser directa, como lo fue para $\boldsymbol{\lambda}\!\rightarrow$ y $\boldsymbol{\lambda}\!\cap$. Más aún, exponemos a continuación que, efectivamente, el método no es aplicable al λ-cubo.

Veamos que si el método puede aplicarse al λ-cubo, entonces puede aplicarse a un PTS arbitrario, utilizando en ambos casos la axiomatización uniforme presentada en [4].

Supongamos que existe una definición de PV tal que se prueba la propiedad de invarianza para los sistemas del cubo. Esto quiere decir que las reglas comunes a los ocho sistemas del cubo preservan las variables positivas a lo largo de la derivación del tipo. Más aún, por lo visto anteriormente:

$$\mathsf{PV}(c) = \varnothing \quad \text{si } c \text{ es una constante}$$

por lo que este resultado no se modifica si se consideran las reglas utilizando constantes $s \in \mathcal{S}$ un conjunto de *sorts* arbitrario. Además la regla (*product*) para PTS's también preserva las variables positivas, pues ésta concluye

$$\Gamma \vdash (\Pi x : A.B) : s_3$$

con $s_3 \in \mathcal{S}$ y trivialmente

$$\mathsf{PV}(s_3) = \varnothing \subseteq \mathsf{PV}(\Gamma)$$

Luego, la misma definición de PV que satisface la propiedad para los sistemas del cubo, la satisface para $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ un PTS arbitrario.

Con esto se tendría como corolario la consistencia de $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$. Pero, por otro lado, se sabe que $\lambda^* = \lambda\big(\{*\}, \{* : *\}, \{(*, *)\}\big)$ es un PTS inconsistente [4],

lo que lleva a un absurdo. Éste viene de suponer que se tiene una definición de PV adecuada para los sistemas del cubo de Barendregt.

Este resultado nos muestra que el método no puede ser aplicado a la axiomatización uniforme presentada en [4] de los sistemas del λ-cubo, del mismo modo que falla al plantearlo sobre los *Pure Type Systems*. Esto lleva a concluir que no es suficiente con considerar las variables del sistema de tipos para analizar su consistencia, pues los PTS's sólo difieren en cuanto a las constantes y las reglas que relacionan a estas últimas.

Sin embargo, hemos visto que, si se tienen en cuenta otras axiomatizaciones de algunos de estos sistemas, el método es perfectamente válido y puede derivarse la consistencia con sólo analizar las variables, como es el caso de $\boldsymbol{\lambda}\!\!\rightarrow$.

Como comentario final sobre el tipado *à la* Church, remarcamos que puede derivarse la consistencia de algunos PTS's a partir de la propiedad de SN, pues la Proposición 5.2.31 presentada en [4] nos muestra que el tipo

$$\Pi\alpha : *.\alpha$$

sólo puede ser habitado por términos que no poseen forma normal, en todo PTS que extienda al sistema $\boldsymbol{\lambda 2}$.

2.5. Discusión

En este capítulo hemos estudiado la propiedad de consistencia de distintos sistemas de tipos para λ-cálculo. Presentamos un método original para este fin, que consiste en el análisis de las variables de tipos a lo largo de la inferencia de los juicios de tipado, utilizando la noción de apariciones positivas de las mismas.

En primer lugar pudimos ver que el nuevo método funciona favorablemente para los sistemas $\boldsymbol{\lambda}\!\!\rightarrow$ y $\boldsymbol{\lambda}\cap$, destacando en este último la diferencia en complejidad al momento de demostrar la propiedad de invarianza positiva sobre las distintas axiomatizaciones del mismo. Esta diferencia era esperable, puesto que se trata de un método que deja completamente de lado la semántica, analizando sólo las construcciones sintácticas del sistema.

Luego buscamos extender el planteo a sistemas de alto orden, como es el caso de $\boldsymbol{\lambda 2}$. Previamente vimos que era posible aplicar el método a subsistemas propios de $\boldsymbol{\lambda 2}$, cada uno de los cuales requería de su propia definición para el concepto de variable positiva. Al extender este análisis al sistema completo nos topamos con dificultades que sugieren que la demostración tiene mayor grado de dificultad.

Por último estudiamos la aplicación del método en los sistemas *à la* Church. Dada la relación existente entre estas versiones de $\boldsymbol{\lambda}\!\!\rightarrow$ y $\boldsymbol{\lambda 2}$, y sus respectivos planteos *à la* Curry, se puede ver directamente que el método es aplicable en el primero del mismo modo que en $\boldsymbol{\lambda}\!\!\rightarrow$-Curry, y presenta, para el segundo caso, las mismas dificultades que para $\boldsymbol{\lambda 2}$-Curry.

El realizar el análisis sobre la axiomatización uniforme para los sistemas del λ-cubo presentada en [4] obtuvimos un resultado que llama la atención. Si bien

era esperable que la simplicidad del método de variables positivas no fuese la misma que en los casos de $\boldsymbol{\lambda}\!\!\rightarrow$ y $\boldsymbol{\lambda}\!\!\cap$ (pues $\boldsymbol{\lambda 2}$ es parte del cubo y otros sistemas del mismo lo extienden), el estudio resultó en que no es aplicable a ninguno de los 8 sistemas involucrados. Más aún, tampoco puede ser planteado sobre los *Pure Type Systems* (PTS's), pues estos se definen sobre la misma axiomatización uniforme planteada para el λ-cubo. Esto se debe a que todos estos sistemas sólo difieren en constantes, las cuales son ignoradas por el método propuesto.

Esta situación llamó la atención, pues $\boldsymbol{\lambda}\!\!\rightarrow$ también es parte del λ-cubo y hemos visto que la consistencia de este sistema puede efectivamente derivarse con el método de variables positivas. El punto interesante, es que se tiene un ejemplo de dos axiomatizaciones de un mismo sistema, para una de las cuales el método es aplicable y para la otra no.

Finalmente podemos concluir que el planteo presentado en este capítulo para probar la consistencia de sistemas de tipos de manera completamente sintáctica, resulta interesante por su simplicidad y facilidad de aplicación en algunos sistemas. Pero, a medida que se estudian sistemas con mayor poder expresivo, las limitaciones del métodos quedan en evidencia, al elevarse su dificultad de aplicación e incluso llegar al punto de no ser aplicable a varios de los sistemas de alto orden estudiados.

Capítulo 3

Tipado de segundo orden para Lógica Combinatoria

Así como resulta de interés el análisis de la correspondencia entre λ-cálculo y CL, puede hacerse lo propio para sus respectivos sistemas de tipos. De este modo, $\boldsymbol{\lambda}\!\!\rightarrow$ resulta equivalente a CL→ bajo la siguiente noción:

1. Sea $M \in CL$, $\Gamma \vdash_{\mathrm{CL}\rightarrow} M : \sigma \Rightarrow \Gamma \vdash_{\boldsymbol{\lambda}\rightarrow} M_\Lambda : \sigma$.

2. Sea $M \in \Lambda$, $\Gamma \vdash_{\boldsymbol{\lambda}\rightarrow} M : \sigma \Rightarrow \Gamma \vdash_{\mathrm{CL}\rightarrow} M_{CL} : \sigma$.

Este mismo análisis fue planteado por Dezani-Ciancaglini y Hindley en [15] para los sistemas de tipos intersección, quedando definido el sistema CL∩, que resulta equivalente a $\boldsymbol{\lambda}\cap$ con el criterio anterior.

Entre los principales sistemas de tipos *à la* Curry, se destacan $\boldsymbol{\lambda}\!\!\rightarrow$, $\boldsymbol{\lambda}\cap$ y $\boldsymbol{\lambda 2}$, siendo este último el único para el cual no se encuentra en la bibliografía una versión equivalente planteada sobre CL. A partir de esto, surge el interés de estudiar las posibles variantes para formalizar un sistema de tipos de segundo orden para CL.

En el presente capítulo, introducimos los distintos análisis que llevaron a una axiomatización definitiva del sistema, al que denominamos CL**2**. Haremos planteos preliminares, exponiendo las motivaciones de cada uno, como así también sus limitaciones, y proponemos posibles soluciones hasta llegar a un sistema que cumple con todas las condiciones buscadas (presentado en la sección 3.4).

En un primer lugar analizamos la extensión ingenua del sistema CL→ con introducción y eliminación de cuantificadores. Para lidiar con las limitaciones propias de este planteo, hacemos un estudio del sistema buscado a través de la lógica a la que, creemos, éste debe ser isomorfo vía la correspondencia de Curry–Howard. Por último, una vez identificados los problemas y limitaciones a solucionar, planteamos el conjunto de reglas definitivo que definen CL**2** y permiten demostrar la equivalencia buscada.

3.1. Extensión básica de CL→

En primer lugar, buscamos plantear el sistema de tipos de alto orden sobre CL extendiendo el ya conocido CL→ con las reglas de introducción y eliminación de cuantificadores, del mismo modo de que **λ2** extiende **λ**→.

$$\frac{\Gamma \vdash_{\mathbf{CL2}} M : \sigma \quad \alpha \notin \mathcal{FV}(\Gamma)}{\Gamma \vdash_{\mathbf{CL2}} M : \forall\alpha.\sigma} \ (\forall I) \qquad \frac{\Gamma \vdash_{\mathbf{CL2}} M : \forall\alpha.\sigma}{\Gamma \vdash_{\mathbf{CL2}} M : \sigma[\alpha := \tau]} \ (\forall E)$$

Esto nos permite derivar nuevos juicios de tipado deseados como:

$$\vdash_{\mathbf{CL2}} \mathbf{SKK} : \forall\alpha.(\alpha \to \alpha)$$

del mismo modo que puede derivarse:

$$\vdash_{\boldsymbol{\lambda 2}} \lambda x.x : \forall\alpha.(\alpha \to \alpha)$$

Pero al mismo tiempo resulta en un poder expresivo menor al buscado, pues al analizar los posibles tipos del combinador **K** en **λ2**, podemos ver que el siguiente juicio es válido:

$$\vdash_{\boldsymbol{\lambda 2}} \lambda xy.x : \sigma \to \forall\alpha.(\tau \to \sigma)$$

con $\alpha \notin \mathcal{FV}(\sigma)$. Claramente esto no es derivable en CL**2**, pues no se pueden introducir cuantificadores dentro de un tipo funcional ya establecido, como es el caso de:

$$\vdash_{\mathbf{CL2}} \mathbf{K} : \sigma \to \tau \to \sigma$$

dado por el axioma (K).

Esto muestra la necesidad de no sólo introducir y eliminar cuantificadores, sino también tener la posibilidad de manipularlos una vez introducidos. De este modo, buscamos poder distribuir cuantificadores de un modo análogo al presentado en el ejemplo bajo las hipótesis adecuadas.

3.2. Planteo vía el isomorfismo de Curry–Howard

Para formalizar esta noción de manipulación de cuantificadores, analizamos sistemas lógicos que incluyen dichos operadores y al mismo tiempo tienen relación con el sistema a formalizar.

Al considerar el sistema CL→ vía la correspondencia de Curry–Howard, se tiene que éste es isomorfo al fragmento de los sistemas deductivos de Hilbert dado por los esquemas-axiomas:

$$Ax_1 : \phi \to (\psi \to \phi)$$
$$Ax_2 : (\phi \to \psi \to \xi) \to (\phi \to \psi) \to \phi \to \xi$$

junto con la regla de inferencia *Modus Ponens* y la asunción de hipótesis.

A partir de esto, surge la idea de analizar los axiomas para el manejo de cuantificadores universales planteados para dichos sistemas [16, 22, 26], como modelos para formalizar las nuevas reglas del sistema CL**2**. Estos son:

$$\begin{array}{ll} Ax_4 : \forall x.\phi \to \phi[x := t] & \\ Ax_5 : \forall x.(\phi \to \psi) \to (\phi \to \forall x.\psi) & \text{con } x \text{ no libre en } \phi \\ Ax_6 : \phi \to \forall x.\phi & \text{con } x \text{ no libre en } \phi \end{array}$$

junto con la nueva regla de inferencia *Generalization*

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \forall x.\phi} \text{ si } x \text{ no aparece libre en } \Gamma$$

que es la versión en lógica de la regla $(\forall I)$ de los sistemas de tipos.

Algunas axiomatizaciones no incluyen esta última regla ya que es consecuencia de los axiomas y el *Modus Ponens* (Teorema de la Generalización) [16].

Este análisis lleva al siguiente planteo del sistema CL**2**, extendiendo CL$\to$ con:

$$\frac{\alpha \notin \mathcal{FV}(\Gamma) \cup \mathcal{FV}(\sigma)}{\Gamma \vdash_{\text{CL}\mathbf{2}} \mathbf{X}_1 : \sigma \to \forall\alpha.\sigma}\ (\forall I) \qquad \frac{}{\Gamma \vdash_{\text{CL}\mathbf{2}} \mathbf{X}_2 : \forall\alpha.\sigma \to \sigma[\alpha := \tau]}\ (\forall E)$$

$$\frac{\alpha \notin \mathcal{FV}(\Gamma) \cup \mathcal{FV}(\sigma)}{\Gamma \vdash_{\text{CL}\mathbf{2}} \mathbf{X}_3 : \forall\alpha.(\sigma \to \tau) \to (\sigma \to \forall\alpha.\tau)}\ (\forall D)$$

donde $\mathbf{X}_1$, $\mathbf{X}_2$ y $\mathbf{X}_3$ son términos a determinar del conjunto CL tal que sean derivables los juicios anteriores para $(\mathbf{X}_1)_\Lambda$, $(\mathbf{X}_2)_\Lambda$ y $(\mathbf{X}_3)_\Lambda$ en $\boldsymbol{\lambda 2}$ respectivamente.

Para los casos de $\mathbf{X}_1$ y $\mathbf{X}_2$, el término en cuestión es claramente el combinador **I** o, equivalentemente, **SKK**. Mientras que para el caso de $\mathbf{X}_3$ buscamos un término tal que $(\mathbf{X}_3)_\Lambda =_\beta (\lambda xy.xy)$, es decir, que su traducción sea equivalente a la identidad de abstracciones en λ-cálculo, y al mismo tiempo pueda derivarse:

$$\Gamma \vdash_{\boldsymbol{\lambda 2}} (\mathbf{X}_3)_\Lambda : \forall\alpha.(\sigma \to \tau) \to (\sigma \to \forall\alpha.\tau)$$

Por el momento asumiremos que dicho término existe y supondremos, al mismo tiempo, que vale $\mathbf{X}_3 MN \twoheadrightarrow_w MN$.

La propiedad principal que se desea que cumpla un sistema de tipos es la clausura bajo reducción. Ésta nos dice que el tipo de un programa (término) se mantienen a lo largo de la ejecución (reducción).

Es inmediato ver que el conjunto de reglas presentado lleva a un sistema que no tiene dicha característica, pues se da la siguiente situación:

$$\begin{array}{ll} x : \sigma \vdash_{\text{CL}\mathbf{2}} x : \sigma & \text{por } (axiom) \\ x : \sigma \vdash_{\text{CL}\mathbf{2}} \mathbf{SKK} : \sigma \to \forall\alpha.\sigma & \text{por } (\forall I) \text{ con } \alpha \notin \mathcal{FV}(\sigma) \\ x : \sigma \vdash_{\text{CL}\mathbf{2}} \mathbf{SKK}x : \forall\alpha.\sigma & \text{por } (\to E) \end{array}$$

y se tiene $\mathbf{SKK}x \twoheadrightarrow_w x$, pero al mismo tiempo:

$$x : \sigma \not\vdash_{\mathrm{CL}\mathbf{2}} x : \forall\alpha.\sigma$$

Esto se debe a que la versión equivalente al Teorema de Generalización, planteada sobre esta axiomatización de CL**2**, efectivamente introduce un cuantificador universal por delante del tipo inferido, pero al mismo tiempo modifica el término sobre el cual se realiza la inferencia, de un modo similar al que se ilustra en el ejemplo anterior.

Para solucionar este nuevo inconveniente, volvamos al análisis de los sistemas deductivos de Hilbert y veamos que puede obviarse el axioma Ax_6 en presencia de *Generalization*, pues la siguiente inferencia es válida:

$\vdash \big(\phi \to (\psi \to \phi) \to \phi\big) \to \big(\phi \to (\psi \to \phi)\big) \to \phi \to \phi$	por Ax_2
$\vdash \phi \to \big((\psi \to \phi) \to \phi\big)$	por Ax_1
$\vdash \big(\phi \to (\psi \to \phi)\big) \to \phi \to \phi$	por MP
$\vdash \phi \to (\psi \to \phi)$	por Ax_1
$\vdash \phi \to \phi$	por MP
$\vdash \forall x.(\phi \to \phi)$	por Gen con x no libre en ϕ
$\vdash \forall x.(\phi \to \phi) \to (\phi \to \forall x.\phi)$	por Ax_5
$\vdash \phi \to \forall x.\phi$	por MP

De esto surge un nuevo planteo del CL**2**, reemplazando la regla $(\forall I)$ por su versión análoga a la regla de generalización, de modo que el término sobre el cual se infiere el tipo no se modifique. Ésta no es otra que la regla de introducción clásica tomada de **λ2**:

$$\frac{\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M : \sigma \quad \alpha \notin \mathcal{FV}(\Gamma)}{\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M : \forall\alpha.\sigma} \ (\forall I)$$

Con este nuevo planteo, el contraejemplo para la propiedad de SR presentado anteriormente ya no es válido. Pero con un análisis similar podemos ver que las reglas $(\forall E)$ y $(\forall D)$ definidas acarrean el mismo inconveniente. Es por esto que se llega a una nueva variante en el sistema, la cual extiende CL→ con el siguiente conjunto de reglas:

$$\frac{\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M : \sigma \quad \alpha \notin \mathcal{FV}(\Gamma)}{\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M : \forall\alpha.\sigma} \ (\forall I) \qquad \frac{\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M : \forall\alpha.\sigma}{\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M : \sigma[\alpha := \tau]} \ (\forall E)$$

$$\frac{\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M : \forall\alpha.(\sigma \to \tau) \quad \alpha \notin \mathcal{FV}(\sigma)}{\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M : \sigma \to \forall\alpha.\tau} \ (\forall D)$$

que provienen de interpretar los axiomas lógicos de los sistemas deductivos de Hilbert como reglas de inferencia.

En este punto del análisis tenemos un sistema que nos permite introducir, eliminar y distribuir cuantificadores universales de la manera buscada y, al mismo tiempo, guarda estrecha relación con los sistemas deductivos *à la* Hilbert.

Veamos a continuación que las diferencias entre las nociones de reducción de λ-cálculo y CL llevan a que la equivalencia buscada entre **λ2** y CL**2** no se cumpla para la presente formulación.

3.3. La extensionalidad en CL

Con la formalización presentada hasta el momento, buscamos probar la siguiente equivalencia entre CL**2** y **λ2** de modo que valga:

1. Sea $M \in CL$, $\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M : \sigma \Rightarrow \Gamma \vdash_{\boldsymbol{\lambda}\mathbf{2}} M_\Lambda : \sigma$.
2. Sea $M \in \Lambda$, $\Gamma \vdash_{\boldsymbol{\lambda}\mathbf{2}} M : \sigma \Rightarrow \Gamma \vdash_{\mathrm{CL}\mathbf{2}} M_{CL} : \sigma$.

Al intentar demostrar esta correspondencia surge el siguiente inconveniente con la regla $(\forall D)$. En CL**2** el siguiente juicio de tipado es válido

$$x : \forall\alpha.(\sigma \to \tau) \vdash_{\mathrm{CL}\mathbf{2}} x : \sigma \to \forall\alpha.\tau$$

para todo $\sigma, \tau \in \mathcal{T}$ asumiendo $\alpha \notin \mathcal{FV}(\sigma)$. Pero al traducir los términos a λ-cálculo se tiene

$$x : \forall\alpha.(\sigma \to \tau) \not\vdash_{\boldsymbol{\lambda}\mathbf{2}} x : \sigma \to \forall\alpha.\tau$$

en el caso general, lo cual muestra que la propiedad buscada no vale para la formulación actual de CL**2**.

Pero al analizar las causas por las que fue posible construir el contraejemplo, se ve que el problema no se encuentra en el sistema planteado sobre CL, sino en su contraparte sobre λ-cálculo. Esto se debe a que, en **λ2**, la manipulación de cuantificadores se realiza mediante la introducción de η-*expansiones*, de modo que vale:

$$\begin{array}{ll}
x : \forall\alpha.(\sigma \to \tau), y : \sigma \vdash_{\boldsymbol{\lambda}\mathbf{2}} x : \sigma \to \tau & \text{por } (\forall E) \\
x : \forall\alpha.(\sigma \to \tau), y : \sigma \vdash_{\boldsymbol{\lambda}\mathbf{2}} xy : \tau & \text{por } (\to E) \\
x : \forall\alpha.(\sigma \to \tau), y : \sigma \vdash_{\boldsymbol{\lambda}\mathbf{2}} xy : \forall\alpha.\tau & \text{por } (\forall I) \text{ con } \alpha \notin \mathcal{FV}(\sigma) \\
\quad x : \forall\alpha.(\sigma \to \tau) \vdash_{\boldsymbol{\lambda}\mathbf{2}} (\lambda y.xy) : \sigma \to \forall\alpha.\tau & \text{por } (\to I)
\end{array}$$

pero, al mismo tiempo, el sistema no posee la propiedad SR para la η-reducción.

Esta noción no es capturada por la lógica combinatoria, por lo que se hace difícil plantear un sistema de tipos que cumpla la equivalencia y al mismo tipo resulte apropiado para el formalismo.

En una primera instancia, se podría "simular" la η-expansión al momento de aplicar $(\forall D)$, como ilustramos a continuación.

Consideremos el término $\mathbf{S}(\mathbf{K}M)(\mathbf{SKK})$. Analizando la relación de reducción de CL, se tiene que:

$$\mathbf{S}(\mathbf{K}M)(\mathbf{SKK})N \twoheadrightarrow_w MN$$

Al mismo tipo, vale:

$$\big(\mathbf{S}(\mathbf{K}M)(\mathbf{SKK})\big)_\Lambda \twoheadrightarrow_\beta (\lambda x.Mx)$$

con $x \notin \mathsf{FV}(M)$. De modo que el término $\mathbf{S}(\mathbf{K}M)(\mathbf{SKK})$ es una suerte de "η-expansión" de M.

De esta manera, podría reemplazarse la regla $(\forall D)$ por:

$$\frac{\Gamma \vdash_{\mathrm{CL2}} M : \forall\alpha.(\sigma \to \tau) \quad \alpha \notin \mathcal{FV}(\sigma)}{\Gamma \vdash_{\mathrm{CL2}} \mathbf{S}(\mathbf{K}M)(\mathbf{SKK}) : \sigma \to \forall\alpha.\tau} \ (\forall D)$$

Al considerar el sistema de manera independiente (sin tener en cuenta la correspondencia con **λ2**), reglas como ésta resultan inconvenientes principalmente porque adoptan una codificación específica de las η-expansiones. Al mismo tiempo, se concluye en un término de mayor complejidad, cuando esto no es estrictamente necesario en un sistema de tipos para CL (aún en el supuesto de que esto preserve la correspondencia buscada).

Otro planteo posible es buscar la equivalencia con el sistema de tipos para λ-cálculo que resulta de agregar a **λ2** las condiciones necesarias para que éste cumpla *η-Subject Reduction*.

A continuación presentamos dicho sistema y sus propiedades principales. Luego, en lo que resta del capítulo, exponemos la formalización definitiva de CL**2** de modo que se cumpla la correspondencia deseada y analizamos las propiedades del nuevo sistema de tipos para lógica combinatoria.

3.3.1. System $\mathbf{F}_\eta$

System F_η fue introducido por Mitchell en [24]. Este sistema resulta de agregar a System F (**λ2**) la regla de tipado para η-reducciones:

$$\frac{\Gamma \vdash \lambda x.(Mx) : \sigma \quad x \notin \mathsf{FV}(M)}{\Gamma \vdash M : \sigma} \ (\eta)$$

De este modo, se tiene trivialmente la propiedad de *η-Subject Reduction*, pues la misma regla lo fuerza, teniendo en cuenta que inductivamente se mantendrá en contextos.

Entre las propiedades principales que posee el sistema, se destacan SN de términos tipables y *$\beta\eta$-Subject Reduction*.

Al mismo tiempo, el conjunto de términos habitados en este sistema es el mismo que en el caso de **λ2**, pues se deduce directamente de la siguiente propiedad [24]:

Propiedad 3.3.1. *Sean Γ un contexto, $M \in \Lambda$ y $\sigma \in \mathcal{T}$. Entonces,*

1. $\Gamma \vdash_{\boldsymbol{\lambda 2}} M : \sigma \Rightarrow \Gamma \vdash_{F_\eta} M : \sigma$.
2. $\Gamma \vdash_{F_\eta} M : \sigma \Rightarrow \exists\, M' \in \Lambda \ [M' \twoheadrightarrow_\eta M \wedge \Gamma \vdash_{\boldsymbol{\lambda 2}} M' : \sigma]$.

Del mismo modo que para $\boldsymbol{\lambda\cap}$, se presenta en [24] un planteo equivalente del sistema utilizando una relación de subtipado $\leq$, y reemplazando las reglas $(\forall E)$ y (η) por $(\leq)$:

$$\frac{}{\Gamma, x : \sigma \vdash_{F_\eta} x : \sigma} \ (axiom)$$

$$\frac{\Gamma, x : \sigma \vdash_{F_\eta} M : \tau}{\Gamma \vdash_{F_\eta} \lambda x.M : \sigma \to \tau} \ (\to I) \qquad \frac{\Gamma \vdash_{F_\eta} M : \sigma \to \tau \quad \Gamma \vdash_{F_\eta} N : \sigma}{\Gamma \vdash_{F_\eta} MN : \tau} \ (\to E)$$

$$\frac{\Gamma \vdash_{F_\eta} M : \sigma \quad \alpha \notin \mathcal{FV}(\Gamma)}{\Gamma \vdash_{F_\eta} M : \forall\alpha.\sigma} \ (\forall I) \qquad \frac{\Gamma \vdash_{F_\eta} M : \sigma \quad \sigma \leq \tau}{\Gamma \vdash_{F_\eta} M : \tau} \ (\leq)$$

De modo que se tiene el tipado de variables $(axiom)$, abstracciones $(\to I)$ y aplicaciones $(\to E)$ de la manera clásica, se permite la introducción de cuantificadores $(\forall I)$ y al mismo tiempo se puede tipar un término con un tipo más general que el inferido mediante la aplicación de $(\leq)$. Donde la relación $\leq$ está dada por las siguientes reglas:

$$\begin{array}{rl}
\text{(refl)} & \sigma \leq \sigma \\
\text{(dummy)} & \sigma \leq \forall\alpha.\sigma \quad \text{si } \alpha \notin \mathcal{FV}(\sigma) \\
\text{(inst)} & \forall\alpha.\sigma \leq \sigma[\alpha := \tau] \\
\text{(distr)} & \forall\alpha.(\sigma \to \tau) \leq \sigma \to \forall\alpha.\tau \quad \text{si } \alpha \notin \mathcal{FV}(\sigma) \\
\text{(trans)} & \sigma \leq \tau \leq \rho \Rightarrow \sigma \leq \rho \\
(\to) & \sigma \leq \sigma', \tau \leq \tau' \Rightarrow \sigma' \to \tau \leq \sigma \to \tau' \\
(\forall) & \sigma \leq \tau \Rightarrow \forall\alpha.\sigma \leq \forall\alpha.\tau
\end{array}$$

Donde se destacan (inst) y (distr) que codifican la eliminación y distribución de cuantificadores. También se tiene la propiedad de contravarianza en el dominio y covarianza en el codominio para los tipos funcionales gracias a la regla $(\to)$. Se permite la inserción de cuantificadores a ambos lados de la desigualdad por $(\forall)$, como así también la de cuantificadores que no ligan variables en el tipo por (dummy). Por último se tiene la reflexividad (refl) y transitividad (trans) de la relación de subtipado.

Al igual que para $\boldsymbol{\lambda\cap}$, existen varias axiomatizaciones de $\leq$ que resultan equivalentes. Decidimos trabajar con ésta, tomada de [30], por su simplicidad y similitud con la presentada para el sistema de tipos intersección.

3.4. El sistema CL2

De aquí en adelante se presenta de manera completa del sistema CL**2**, se expone una prueba de la equivalencia con System F_η, y a continuación se analizan las distintas propiedades que cumple el nuevo sistema de tipos.

3.4.1. Reglas del sistema

El siguiente planteo surge de extender CL→ con las reglas $(\forall I)$ y $(\leq)$, del mismo modo que System F_η extiende $\boldsymbol{\lambda}\!\!\rightarrow$. De este modo, el sistema CL**2** queda definido con el conjunto de reglas:

$$\frac{}{\Gamma, x : \sigma \vdash_{\mathbf{CL2}} x : \sigma}\ (\text{axiom}) \qquad \frac{}{\Gamma \vdash_{\mathbf{CL2}} \mathbf{K} : \sigma \to \tau \to \sigma}\ (K)$$

$$\frac{}{\Gamma \vdash_{\mathbf{CL2}} \mathbf{S} : (\sigma \to \tau \to \rho) \to (\sigma \to \tau) \to \sigma \to \rho}\ (S)$$

$$\frac{\Gamma \vdash_{\mathbf{CL2}} M : (\sigma \to \tau) \quad \Gamma \vdash_{\mathbf{CL2}} N : \sigma}{\Gamma \vdash_{\mathbf{CL2}} MN : \tau}\ (\to E)$$

$$\frac{\Gamma \vdash_{\mathbf{CL2}} M : \sigma \quad \alpha \notin \mathcal{FV}(\Gamma)}{\Gamma \vdash_{\mathbf{CL2}} M : \forall\alpha.\sigma}\ (\forall I) \qquad \frac{\Gamma \vdash_{\mathbf{CL2}} M : \sigma \quad \sigma \leq \tau}{\Gamma \vdash_{\mathbf{CL2}} M : \tau}\ (\leq)$$

Agregando al poder expresivo de CL→ la posibilidad de introducir cuantificadores universales y generalizar los tipos inferidos utilizando la misma noción de subtipado presentada anteriormente.

3.4.2. Correspondencia con System F_η

El objetivo de esta sección es demostrar la correspondencia con System F_η. Presentamos a continuación una serie de lemas que sirven a dicho propósito, junto con sus correspondiente demostraciones y una breve descripción de su utilidad.

El siguiente lema muestra que la inferencia del tipo de un término M depende sólo de las variables libres de éste, de modo que se pueden agregar hipótesis al contexto de tipado (siempre y cuando éstas no contradigan las asunciones ya hechas) o restringir estas últimas a las que involucran variables de M.

Lema 3.4.1 (Base Lemma)**.** *Sea* Γ *una base.*

1. *Si* $\Delta \supseteq \Gamma$ *es otra base, entonces* $\Gamma \vdash_{\mathbf{CL2}} M : \delta \Rightarrow \Delta \vdash_{\mathbf{CL2}} M : \delta$.
2. $\Gamma \vdash_{\mathbf{CL2}} M : \delta \Rightarrow \mathsf{FV}(M) \subseteq \mathrm{dom}(\Gamma)$.
3. $\Gamma \vdash_{\mathbf{CL2}} M : \delta \Rightarrow \Gamma\restriction_{\mathsf{FV}(M)} \vdash_{\mathbf{CL2}} M : \delta$.

Demostración. Por inducción en la derivación de $\Gamma \vdash_{\mathbf{CL2}} M : \delta$, analizando la última regla aplicada.

1. **Caso 1:** Se aplicó (*axiom*). Luego $M \equiv x$ y vale

$$\Gamma = \Gamma', x : \delta \vdash_{\mathbf{CL2}} x : \delta$$

Como $\Delta \supseteq \Gamma$ es una base y $x : \delta \in \Gamma$, entonces $\Delta = \Delta', x : \delta$. Luego, por $(axiom)$

$$\Delta \vdash_{\mathrm{CL2}} x : \delta$$

Caso 2: Se usó (K), por lo que $M \equiv \mathbf{K}$ y $\delta \equiv \sigma \to \tau \to \sigma$. Luego, por (K)

$$\Delta \vdash_{\mathrm{CL2}} \mathbf{K} : \sigma \to \tau \to \sigma$$

El caso en que la última regla aplicada fue (S) es completamente análogo a este último.

Caso 3: Si la última regla aplicada fue $(\to E)$, se tiene $M \equiv M_1 M_2$ y valen

$$\Gamma \vdash_{\mathrm{CL2}} M_1 : \sigma \to \delta \quad \text{y} \quad \Gamma \vdash_{\mathrm{CL2}} M_2 : \sigma$$

Entonces, por hipótesis inductiva

$$\Delta \vdash_{\mathrm{CL2}} M_1 : \sigma \to \delta \quad \text{y} \quad \Delta \vdash_{\mathrm{CL2}} M_2 : \sigma$$

Luego, por $(\to E)$ vale

$$\Delta \vdash_{\mathrm{CL2}} M_1 M_2 : \delta$$

Caso 4: Si se aplicó $(\forall I)$, se tiene $\delta \equiv \forall \alpha.\delta'$ con $\alpha \notin \mathcal{FV}(\Gamma)$ y vale $\Gamma \vdash_{\mathrm{CL2}} M : \delta'$. Podemos asumir $\alpha \notin \mathcal{FV}(\Delta)$ por convención de variables. Por hipótesis inductiva se tiene

$$\Delta \vdash_{\mathrm{CL2}} M : \delta'$$

Luego, por $(\forall I)$

$$\Delta \vdash_{\mathrm{CL2}} M : \forall \alpha.\delta'$$

Caso 5: Si la última regla fue $(\leq)$, vale $\Gamma \vdash_{\mathrm{CL2}} M : \rho$ y $\rho \leq \delta$. Por hipótesis inductiva vale

$$\Delta \vdash_{\mathrm{CL2}} M : \rho$$

Luego, por $(\leq)$, se tiene

$$\Delta \vdash_{\mathrm{CL2}} M : \delta$$

2. **Caso 1:** Se aplicó $(axiom)$, por lo que $M \equiv x$ y vale

$$\Gamma = \Gamma', x : \delta \vdash_{\mathrm{CL2}} x : \delta$$

Luego,

$$\mathsf{FV}(x) = \{x\} \subseteq \mathrm{dom}(\Gamma') \cup \{x\} = \mathrm{dom}(\Gamma)$$

Caso 2: Se usaron las reglas (K) o (S). Luego, $M \equiv \mathbf{K}$ o $M \equiv \mathbf{S}$ por lo que vale

$$\mathsf{FV}(\mathbf{K}) = \mathsf{FV}(\mathbf{S}) = \varnothing \subseteq \mathrm{dom}(\Gamma)$$

Caso 3: La última regla usada fue $(\to E)$. Entonces $M \equiv M_1 M_2$ y, por hipótesis inductiva

$$\mathsf{FV}(M_1) \subseteq \mathrm{dom}(\Gamma) \quad \text{y} \quad \mathsf{FV}(M_2) \subseteq \mathrm{dom}(\Gamma)$$

Luego,

$$\mathsf{FV}(M_1 M_2) = \mathsf{FV}(M_1) \cup \mathsf{FV}(M_2) \subseteq \mathrm{dom}(\Gamma)$$

Caso 4: Se aplicaron las reglas $(\forall I)$ o $(\leq)$. En ambos casos el término no se modifica, por lo que de la hipótesis inductiva se tiene

$$\mathsf{FV}(M) \subseteq \mathrm{dom}(\Gamma)$$

3. **Caso 1:** Si se aplicó $(axiom)$, entonces $M \equiv x$ y $\Gamma = \Gamma', x : \delta$. Por definición, $\Gamma\restriction_{\mathsf{FV}(x)} = \{x : \delta\}$. Luego, por $(axiom)$ se tiene

$$\Gamma\restriction_{\mathsf{FV}(x)} = \{x : \delta\} \vdash_{\mathrm{CL}^2} x : \delta$$

Caso 2: Se usaron las reglas (K) o (S). Analizamos solamente el caso de la primera, pues son completamente análogos.
Se tiene $M \equiv \mathbf{K}$ y $\delta \equiv \sigma \to \tau \to \sigma$. Por (K), vale

$$\Gamma\restriction_{\mathsf{FV}(\mathbf{K})} = \varnothing \vdash_{\mathrm{CL}^2} \mathbf{K} : \sigma \to \tau \to \sigma$$

Caso 3: Se aplicó la regla $(\to E)$. Luego, se tiene $M \equiv M_1 M_2$ y por hipótesis inductiva

$$\Gamma\restriction_{\mathsf{FV}(M_1)} \vdash_{\mathrm{CL}^2} M_1 : \rho \to \delta$$
$$\text{y} \quad \Gamma\restriction_{\mathsf{FV}(M_2)} \vdash_{\mathrm{CL}^2} M_2 : \rho$$

Como $\Gamma\restriction_{\mathsf{FV}(M_1 M_2)} \supseteq \Gamma\restriction_{\mathsf{FV}(M_1)}$ y $\Gamma\restriction_{\mathsf{FV}(M_1 M_2)} \supseteq \Gamma\restriction_{\mathsf{FV}(M_2)}$, por (1) vale

$$\Gamma\restriction_{\mathsf{FV}(M_1 M_2)} \vdash_{\mathrm{CL}^2} M_1 : \rho \to \delta$$
$$\text{y} \quad \Gamma\restriction_{\mathsf{FV}(M_1 M_2)} \vdash_{\mathrm{CL}^2} M_2 : \rho$$

Entonces, aplicando $(\to E)$ se tiene

$$\Gamma\restriction_{\mathsf{FV}(M_1 M_2)} \vdash_{\mathrm{CL}^2} M_1 M_2 : \delta$$

Caso 4: Se usaron las reglas $(\forall I)$ o $(\leq)$. Exponemos en detalle el caso de la primera, pues para $(\leq)$ el razonamiento es análogo.
Se tiene $\delta \equiv \forall \alpha . \delta'$ y $\alpha \notin \mathcal{FV}(\Gamma)$. Por hipótesis inductiva

$$\Gamma\restriction_{\mathsf{FV}(M)} \vdash_{\mathrm{CL}^2} M : \delta'$$

y además

$$\alpha \notin \mathcal{FV}(\Gamma) \Rightarrow \alpha \notin \mathcal{FV}\big(\Gamma\restriction_{\mathsf{FV}(M)}\big)$$

Luego, por $(\forall I)$

$$\Gamma\restriction_{\mathsf{FV}(M)} \vdash_{\mathrm{CL}^2} M : \forall \alpha . \delta'$$

□

El Lema 3.4.1 nos permite manipular el contexto de tipado de diversas maneras, siendo la que se utilizará con más frecuencia la siguiente

$$\Gamma, x : \sigma \vdash_{\mathbf{CL2}} M : \tau \quad \text{y} \quad x \notin \mathsf{FV}(M) \Rightarrow \Gamma \vdash_{\mathbf{CL2}} M : \tau$$

Esto vale, pues $(\Gamma, x : \sigma)\restriction_{\mathsf{FV}(M)} \subseteq \Gamma$, y por los ítems (1) y (3) del lema, se tiene $\Gamma \vdash_{\mathbf{CL2}} M : \tau$.

A continuación presentamos una versión restringida del lema de generación para el sistema CL**2**. Si bien creemos que puede formularse una versión completa, que incluya los casos de los combinadores **S** y **K**, no es inmediato llegar a un planteo apropiado del mismo. Por esto nos limitamos a introducir los casos de las variables y aplicaciones, que resultan intuitivos para su uso y de vital utilidad a la hora de demostrar la equivalencia buscada.

Lema 3.4.2 (Generation Lemma)**.** *Si* $\Gamma \vdash_{\mathbf{CL2}} M : \delta$*, entonces:*

1. $M \equiv x \Rightarrow \exists\, \sigma \leq \delta \;\; [x : \sigma \in \Gamma]$.
2. $M \equiv M_1 M_2 \Rightarrow \exists \sigma \exists\, \tau \leq \delta \;\; [\Gamma \vdash_{\mathbf{CL2}} M_1 : \sigma \to \tau \quad y \quad \Gamma \vdash_{\mathbf{CL2}} M_2 : \sigma]$.

Demostración. Por inducción en la derivación de $\Gamma \vdash_{\mathbf{CL2}} M : \delta$.

1. $M \equiv x$. Sólo se involucran la reglas $(axiom)$, $(\forall I)$ y $(\leq)$:

   Caso 1: Se aplicó $(axiom)$, por lo que vale

   $$\Gamma, x : \delta \vdash_{\mathbf{CL2}} x : \delta$$

   Luego, basta tomar $\sigma = \delta$.

   Caso 2: Si se usó $(\forall I)$, $\delta \equiv \forall \alpha.\delta'$, $\alpha \notin \mathcal{FV}(\Gamma)$ y vale $\Gamma \vdash_{\mathbf{CL2}} x : \delta'$. Por hipótesis inductiva

   $$\exists\, \sigma \leq \delta' \;\; [x : \sigma \in \Gamma]$$

   Como $x : \sigma \in \Gamma$ y $\alpha \notin \mathcal{FV}(\Gamma)$, entonces $\alpha \notin \mathcal{FV}(\sigma)$. Luego, por $(\forall)$

   $$\forall \alpha.\sigma \leq \forall \alpha.\delta' \equiv \delta$$

   y por (dummy)

   $$\sigma \leq \forall \alpha.\sigma$$

   Entonces, por (trans) se tiene $\sigma \leq \delta$.

   Caso 3: Se usó la regla $(\leq)$. Entonces vale $\Gamma \vdash_{\mathbf{CL2}} x : \rho$ y $\rho \leq \delta$. Por hipótesis inductiva

   $$\exists\, \sigma \leq \rho \;\; [x : \sigma \in \Gamma]$$

   Luego, por (trans) $\sigma \leq \delta$.

2. $M \equiv M_1 M_2$, por lo que sólo intervienen las reglas $(\to E)$, $(\forall I)$ y $(\leq)$.

Caso 1: Se aplicó $(\to E)$. Luego, valen

$$\Gamma \vdash_{\mathbf{CL2}} M_1 : \rho \to \delta \quad \text{y} \quad \Gamma \vdash_{\mathbf{CL2}} M_2 : \rho$$

Entonces, tomamos $\sigma = \rho$ y $\tau = \delta$ y se tiene

$$\exists\sigma\exists\,\tau \leq \delta \;\; [\Gamma \vdash_{\mathbf{CL2}} M_1 : \sigma \to \tau \quad \text{y} \quad \Gamma \vdash_{\mathbf{CL2}} M_2 : \sigma]$$

Caso 2: Si la última regla aplicada fue $(\forall I)$, $\delta \equiv \forall\alpha.\delta'$ con $\alpha \notin \mathcal{FV}(\Gamma)$ y vale $\Gamma \vdash_{\mathbf{CL2}} M_1M_2 : \delta'$. Por hipótesis inductiva

$$\exists\sigma\exists\,\tau \leq \delta' \;\; [\Gamma \vdash_{\mathbf{CL2}} M_1 : \sigma \to \tau \quad \text{y} \quad \Gamma \vdash_{\mathbf{CL2}} M_2 : \sigma]$$

Por (inst)

$$\forall\alpha.\sigma \leq \sigma[\alpha := \alpha] \equiv \sigma$$

Luego, por $(\to)$

$$\sigma \to \tau \leq \forall\alpha.\sigma \to \tau$$

Entonces, por $(\leq)$

$$\Gamma \vdash_{\mathbf{CL2}} M_1 : \forall\alpha.\sigma \to \tau$$

Como $\alpha \notin \mathcal{FV}(\Gamma)$, por $(\forall I)$

$$\Gamma \vdash_{\mathbf{CL2}} M_1 : \forall\alpha.(\forall\alpha.\sigma \to \tau)$$

Además, como $\alpha \notin \mathcal{FV}(\forall\alpha.\sigma)$, por (distr) y $(\leq)$

$$\Gamma \vdash_{\mathbf{CL2}} M_1 : \forall\alpha.\sigma \to \forall\alpha.\tau$$

Por otro lado vale

$$\Gamma \vdash_{\mathbf{CL2}} M_2 : \forall\alpha.\sigma$$

por $(\forall I)$, pues $\alpha \notin \mathcal{FV}(\Gamma)$. Luego, basta tomar $\sigma' = \forall\alpha.\sigma$ y $\tau' = \forall\alpha.\tau$ y se tiene

$$\exists\sigma'\exists\,\tau' \leq \delta \;\; [\Gamma \vdash_{\mathbf{CL2}} M_1 : \sigma' \to \tau' \quad \text{y} \quad \Gamma \vdash_{\mathbf{CL2}} M_2 : \sigma']$$

pues, por $(\forall)$, $\tau' \equiv \forall\alpha.\tau \leq \forall\alpha.\delta' \equiv \delta$.

Caso 3: Si se usó $(\leq)$, se tiene $\Gamma \vdash_{\mathbf{CL2}} M_1M_2 : \rho$ y $\rho \leq \delta$. Por hipótesis inductiva

$$\exists\sigma\exists\,\tau \leq \rho \;\; [\Gamma \vdash_{\mathbf{CL2}} M_1 : \sigma \to \tau \quad \text{y} \quad \Gamma \vdash_{\mathbf{CL2}} M_2 : \sigma]$$

Como $\tau \leq \rho$ y $\rho \leq \delta$, por (trans), $\tau \leq \delta$.

□

La interpretación intuitiva del lema anterior es que el tipo derivado de una variable debe ser la generalización, en términos de subtipado, de una declaración incluida en el contexto. Del mismo modo, para el caso de las aplicaciones, puede utilizarse la regla $(\leq)$ luego de construida la aplicación, por lo que el tipo de la imagen funcional del término izquierdo debe ser subtipo del inferido para el término resultante.

Para demostrar la equivalencia entre CL**2** y System F_η es necesario ver que puede simularse el tipado de abstracciones en CL, bajo las mismas condiciones que se requieren en λ-cálculo. El siguiente lema tiene ese propósito:

Lema 3.4.3 (Abstraction Simulation)**.** $\Gamma, x : \sigma \vdash_{\text{CL2}} M : \tau \Rightarrow \Gamma \vdash_{\text{CL2}} \lambda^* x.M : \sigma \to \tau$.

Demostración. Por inducción en el término, analizando la definición de $\lambda^* x.M$.

Caso 1: $M \equiv x$ y $\lambda^* x.x = \mathbf{SKK}$. Se tiene

$$\Gamma, x : \sigma \vdash_{\text{CL2}} x : \tau$$

Por Lema 3.4.2 (1)

$$\exists\, \sigma' \leq \tau \;\; \big[x : \sigma' \in (\Gamma, x : \sigma)\big]$$

Luego, $\sigma' = \sigma$ y $\sigma \leq \tau$. Por (S)

$$\Gamma \vdash_{\text{CL2}} \mathbf{S} : \big(\sigma \to (\rho \to \sigma) \to \sigma\big) \to \big(\sigma \to (\rho \to \sigma)\big) \to \sigma \to \sigma$$

Además, por (K)

$$\Gamma \vdash_{\text{CL2}} \mathbf{K} : \sigma \to (\rho \to \sigma) \to \sigma \qquad \text{y} \qquad \Gamma \vdash_{\text{CL2}} \mathbf{K} : \sigma \to \rho \to \sigma$$

Luego, por $(\to E)$ 2 veces, se tiene

$$\Gamma \vdash_{\text{CL2}} \mathbf{SKK} : \sigma \to \sigma$$

Como $\sigma \leq \sigma$ y $\sigma \leq \tau$, por $(\to)$ y $(\leq)$

$$\Gamma \vdash_{\text{CL2}} \mathbf{SKK} : \sigma \to \tau$$

Caso 2: Veamos el caso en que $x \notin \mathsf{FV}(M)$, por ejemplo $M \equiv y$ con $y \neq x$ o $M \equiv \mathbf{S}$. Tenemos $\lambda^* x.M = \mathbf{K}M$. Luego, por Lema 3.4.1 se tiene

$$\Gamma \vdash_{\text{CL2}} M : \tau$$

Además, por (K)

$$\Gamma \vdash_{\text{CL2}} \mathbf{K} : \tau \to \sigma \to \tau$$

Luego, por $(\to E)$ se tiene

$$\Gamma \vdash_{\text{CL2}} \mathbf{K}M : \sigma \to \tau$$

Caso 4: En este caso $M \equiv M'x$ con $x \notin \mathsf{FV}(M')$ y $\lambda^* x.M'x = M'$. Vale

$$\Gamma, x : \sigma \vdash_{\mathrm{CL}_2} M'x : \tau$$

Por Lema 3.4.2 (2)

$$\exists \sigma' \exists \tau' \leq \tau \ [\Gamma, x : \sigma \vdash_{\mathrm{CL}_2} M' : \sigma' \to \tau' \quad \text{y} \quad \Gamma, x : \sigma \vdash_{\mathrm{CL}_2} x : \sigma']$$

Ahora por Lema 3.4.2 (1)

$$\exists\, \sigma'' \leq \sigma' \ \left[x : \sigma'' \in (\Gamma, x : \sigma)\right]$$

Luego, $\sigma = \sigma'' \leq \sigma'$. Como $\sigma \leq \sigma'$ y $\tau' \leq \tau$, por $(\to)$ y $(\leq)$

$$\Gamma, x : \sigma \vdash_{\mathrm{CL}_2} M' : \sigma \to \tau$$

Además $x \notin \mathsf{FV}(M')$, entonces, por Lema 3.4.1

$$\Gamma \vdash_{\mathrm{CL}_2} M' : \sigma \to \tau$$

Caso 3: $M \equiv M_1 M_2$ con $x \in \mathsf{FV}(M)$. Por definición

$$\lambda^* x.(M_1 M_2) = \mathbf{S}(\lambda^* x.M_1)(\lambda^* x.M_2)$$

Por Lema 3.4.2 (2)

$$\exists \sigma' \exists \tau' \leq \tau \ [\Gamma, x : \sigma \vdash_{\mathrm{CL}_2} M_1 : \sigma' \to \tau' \quad \text{y} \quad \Gamma, x : \sigma \vdash_{\mathrm{CL}_2} M_2 : \sigma']$$

Como $\sigma' \leq \sigma'$ y $\tau' \leq \tau$ por $(\to)$ y $(\leq)$

$$\Gamma, x : \sigma \vdash_{\mathrm{CL}_2} M_1 : \sigma' \to \tau$$

Luego, por hipótesis inductiva

$$\Gamma \vdash_{\mathrm{CL}_2} \lambda^* x.M_1 : \sigma \to \sigma' \to \tau \qquad \text{y} \qquad \Gamma \vdash_{\mathrm{CL}_2} \lambda^* x.M_2 : \sigma \to \sigma'$$

Por (S)

$$\Gamma \vdash_{\mathrm{CL}_2} \mathbf{S} : (\sigma \to \sigma' \to \tau) \to (\sigma \to \sigma') \to \sigma \to \tau$$

Luego, por $(\to E)$ 2 veces, se tiene

$$\Gamma \vdash_{\mathrm{CL}_2} \mathbf{S}(\lambda^* x.M_1)(\lambda^* x.M_2) : \sigma \to \tau$$

□

Con los resultados presentados hasta el momento, estamos en condiciones de demostrar la equivalencia entre los dos sistemas:

Proposición 3.4.1. *Sea Γ un contexto, entonces:*

1. $M \in CL$, $\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M : \sigma \Rightarrow \Gamma \vdash_{F_\eta} M_\Lambda : \sigma$.
2. $M \in \Lambda$, $\Gamma \vdash_{F_\eta} M : \sigma \Rightarrow \Gamma \vdash_{\mathrm{CL}\mathbf{2}} M_{CL} : \sigma$.

Demostración. Por inducción en la derivación del juicio de tipado, analizando la última regla aplicada.

1. **Caso 1:** Se aplicó $(axiom)$, entonces $M \equiv x$ y

$$\Gamma, x : \sigma \vdash_{\mathrm{CL}\mathbf{2}} x : \sigma$$

Como $x_\Lambda = x$, por $(axiom)$

$$\Gamma, x : \sigma \vdash_{F_\eta} x : \sigma$$

Caso 2: Se usó (K). $M \equiv \mathbf{K}$, $\mathbf{K}_\Lambda = \lambda xy.x$ y $\sigma \equiv \tau \to \rho \to \tau$. Luego, considerando el contexto $\Gamma' = (\Gamma, x : \tau, y : \rho)$, vale la siguiente inferencia

$$\begin{array}{rl} \Gamma, x : \tau, y : \rho \vdash_{F_\eta} x : \tau & \text{por } (axiom) \\ \Gamma, x : \tau \vdash_{F_\eta} \lambda y.x : \rho \to \tau & \text{por } (\to I) \\ \Gamma \vdash_{F_\eta} \lambda xy.x : \tau \to \rho \to \tau & \text{por } (\to I) \end{array}$$

Caso 3: Se usó (S). Análogamente al caso anterior, se tiene $M \equiv \mathbf{S}$, $\mathbf{S}_\Lambda = \lambda xyz.\big(xz(yz)\big)$ y $\sigma \equiv (\tau \to \rho \to \delta) \to (\tau \to \rho) \to \tau \to \delta$. Tomamos los contextos

$$\begin{aligned} \Gamma' &= \Gamma, x : \tau \to \rho \to \delta \\ \Gamma'' &= \Gamma', y : \tau \to \rho \\ \Gamma''' &= \Gamma'', z : \tau \end{aligned}$$

y vale

$$\begin{array}{rl} \Gamma''' \vdash_{F_\eta} x : \tau \to \rho \to \delta & \text{por } (axiom) \\ \Gamma''' \vdash_{F_\eta} y : \tau \to \rho & \text{por } (axiom) \\ \Gamma''' \vdash_{F_\eta} z : \tau & \text{por } (axiom) \\ \Gamma''' \vdash_{F_\eta} xz : \rho \to \delta & \text{por } (\to E) \\ \Gamma''' \vdash_{F_\eta} yz : \rho & \text{por } (\to E) \\ \Gamma''' \vdash_{F_\eta} xz(yz) : \delta & \text{por } (\to E) \\ \Gamma'' \vdash_{F_\eta} \lambda z.\big(xz(yz)\big) : \tau \to \delta & \text{por } (\to I) \\ \Gamma' \vdash_{F_\eta} \lambda yz.\big(xz(yz)\big) : (\tau \to \rho) \to \tau \to \delta & \text{por } (\to I) \\ \Gamma \vdash_{F_\eta} \lambda xyz.\big(xz(yz)\big) : \\ \qquad (\tau \to \rho \to \delta) \to (\tau \to \rho) \to \tau \to \delta & \text{por } (\to I) \end{array}$$

Caso 4: Los casos $(\rightarrow E)$, $(\forall I)$ y $(\leq)$ se deducen directamente de la hipótesis inductiva, pues las tres reglas son iguales en ambos sistemas. Mostramos a continuación el caso para la regla $(\rightarrow E)$. Se tiene $M \equiv M_1 M_2$ y vale

$$\Gamma \vdash_{\mathrm{CL2}} M_1 : \tau \rightarrow \sigma \quad \text{y} \quad \Gamma \vdash_{\mathrm{CL2}} M_2 : \tau$$

Por hipótesis inductiva se tiene

$$\Gamma \vdash_{F_\eta} (M_1)_\Lambda : \tau \rightarrow \sigma \quad \text{y} \quad \Gamma \vdash_{F_\eta} (M_2)_\Lambda : \tau$$

y por $(\rightarrow E)$ vale

$$\Gamma \vdash_{F_\eta} (M_1 M_2)_\Lambda : \sigma$$

2. Los casos en que las últimas reglas aplicadas fueron $(axiom)$, $(\rightarrow E)$, $(\forall I)$ o $(\leq)$ son completamente análogos a los expuestos en (1). Analizamos únicamente el caso $(\rightarrow I)$:

   Se tiene $M \equiv \lambda x.M'$, $M_{CL} = \lambda^* x.M'_{CL}$, $\sigma \equiv \tau \rightarrow \rho$ y vale

   $$\Gamma, x : \tau \vdash_{F_\eta} M' : \rho$$

   Por hipótesis inductiva

   $$\Gamma, x : \tau \vdash_{\mathrm{CL2}} M'_{CL} : \rho$$

   Luego, por Lema 3.4.3 vale

   $$\Gamma \vdash_{\mathrm{CL2}} \lambda^* x.M'_{CL} : \tau \rightarrow \rho$$

□

Esto demuestra la equivalencia entre CL**2** y System F_η. De este modo, se tiene un sistema de tipos polimórficos de segundo orden apropiado para CL, logrando el objetivo planteado al inicio de la presente tesis.

Antes de pasar al análisis de las propiedades del sistema exponemos un ejemplo de derivación, para ilustrar su funcionamiento.

En [15] los autores presentan el sistema CL$\cap$ con la siguiente variante en el axioma del combinador **S**:

$$\frac{}{\Gamma \vdash_{\mathrm{CL}\cap} \mathbf{S} : (\sigma \rightarrow \tau \rightarrow \rho) \rightarrow (\delta \rightarrow \tau) \rightarrow \sigma \cap \delta \rightarrow \rho} \quad (S)$$

Una versión análoga de este axioma, sobre tipos polimórficos de segundo orden, que hemos analizado en versiones preliminares del sistema CL**2** (las cuales no se exponen en el presente trabajo) es la siguiente:

$$\frac{}{\Gamma \vdash \mathbf{S} : (\sigma \rightarrow \tau \rightarrow \rho) \rightarrow (\delta \rightarrow \tau) \rightarrow \forall \alpha.\alpha \rightarrow \rho} \quad (S)$$

Veamos ahora que puede derivarse este juicio de tipado en CL**2**. En primer lugar, la desigualdad (1) es válida por el siguiente análisis:

$$\begin{aligned}
\forall\alpha.\alpha \leq \alpha[\alpha := \delta] \equiv \delta \quad & \text{por (inst)} \\
\tau \leq \tau \quad & \text{por (refl)} \\
\delta \to \tau \leq \forall\alpha.\alpha \to \tau \quad & \text{por } (\to) \\
\forall\alpha.\alpha \to \rho \leq \forall\alpha.\alpha \to \rho \quad & \text{por (refl)} \\
(\forall\alpha.\alpha \to \tau) \to \forall\alpha.\alpha \to \rho \leq (\delta \to \tau) \to \forall\alpha.\alpha \to \rho \quad & \text{por } (\to) \qquad (1)
\end{aligned}$$

Luego, puede deducirse (2) con la siguiente inferencia:

$$\begin{aligned}
\forall\alpha.\alpha \leq \sigma \quad & \text{por (inst)} \\
\tau \to \rho \leq \tau \to \rho \quad & \text{por (refl)} \\
\sigma \to \tau \to \rho \leq \forall\alpha.\alpha \to \tau \to \rho \quad & \text{por } (\to) \\
(\forall\alpha.\alpha \to \tau \to \rho) \to (\forall\alpha.\alpha \to \tau) \to \forall\alpha.\alpha \to \rho \leq \quad & \text{por } (\to) \\
(\sigma \to \tau \to \rho) \to (\delta \to \tau) \to \forall\alpha.\alpha \to \rho \quad & \text{con (1)} \qquad (2)
\end{aligned}$$

Finalmente se deriva el juicio deseado, así:

$$\begin{aligned}
&\Gamma \vdash_{\text{CL2}} \mathbf{S} : (\forall\alpha.\alpha \to \tau \to \rho) \to (\forall\alpha.\alpha \to \tau) \to \forall\alpha.\alpha \to \rho \quad \text{por } (S) \\
&\Gamma \vdash_{\text{CL2}} \mathbf{S} : (\sigma \to \tau \to \rho) \to (\delta \to \tau) \to \forall\alpha.\alpha \to \rho \quad \text{por } (\leq) \text{ con } (2)
\end{aligned}$$

Aquí se ve que el axioma (S) es más general que el propuesto en versiones preliminares, pues puede deducirse este último a partir del primero.

3.4.3. Propiedades del sistema

Una vez demostrada la correspondencia entre CL**2** y System F_η, pasamos a estudiar las propiedades principales del sistema (SN de términos tipables, SR, etc.), muchas de las cuales se "heredan" de este último.

En primer lugar veamos que la propiedad de clausura puede deducirse a partir de la equivalencia:

Proposición 3.4.2 (Subject Reduction)**.** *Sean $M, M' \in CL$ tal que $M \to_w M'$. Entonces*

$$\Gamma \vdash_{\text{CL2}} M : \sigma \Rightarrow \Gamma \vdash_{\text{CL2}} M' : \sigma$$

Demostración. Por inducción en el término M. Los casos en que el término es un átomo valen trivialmente ya que no existe un M' tal que $M \to_w M'$. Analizamos a continuación las diferentes posibilidades para la aplicación.

Caso 1: La reducción se produce en la raíz, con $M = \mathbf{K}M_1M_2$ y $M' = M_1$. Por la Proposición 3.4.1 (1)

$$\Gamma \vdash_{\text{CL2}} \mathbf{K}M_1M_2 : \sigma \Rightarrow \Gamma \vdash_{F_\eta} (\mathbf{K}M_1M_2)_\Lambda : \sigma$$

Además

$$(\mathbf{K}M_1M_2)_\Lambda = (\lambda xy.x)(M_1)_\Lambda(M_2)_\Lambda \twoheadrightarrow_\beta (M_1)_\Lambda$$

Luego, por SR de System F_η

$$\Gamma \vdash_{F_\eta} (M_1)_\Lambda : \sigma$$

Como $\big((M_1)_\Lambda\big)_{CL} = M_1$, por la Proposición 3.4.1 (2)

$$\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M_1 : \sigma$$

Caso 2: Si la reducción se produce en la raíz, con $M = \mathbf{S}M_1M_2M_3$, el razonamiento es completamente análogo al caso anterior. Se tiene $M' = M_1M_3(M_2M_3)$ y valen

$$M_\Lambda \twoheadrightarrow_\beta M'_\Lambda \text{ y } (M'_\Lambda)_{CL} = M'$$

Caso 3: La reducción es interna, con $M = M_1M_2$, $M' = M'_1M_2$ y $M_1 \to_w M'_1$. Vale $\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M_1M_2 : \sigma$ y por Lema 3.4.2 (2)

$$\exists\rho\exists\,\tau \leq \sigma \ [\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M_1 : \rho \to \tau \quad \text{y} \quad \Gamma \vdash_{\mathrm{CL}\mathbf{2}} M_2 : \tau]$$

Luego, por hipótesis inductiva

$$\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M'_1 : \rho \to \tau$$

Entonces, por $(\to E)$ y $(\leq)$ con $\tau \leq \sigma$ se tiene

$$\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M'_1M_2 : \sigma$$

Análogamente se prueba el caso en que $M_1M_2 \to_w M_1M'_2$.

□

Una vez probada la propiedad de clausura para un paso se reducción, es inmediato extender el resultado a muchos pasos:

Corolario 3.4.1. *Sean $M, M' \in CL$ tal que $M \twoheadrightarrow_w M'$. Entonces*

$$\Gamma \vdash_{\mathrm{CL}\mathbf{2}} M : \sigma \Rightarrow \Gamma \vdash_{\mathrm{CL}\mathbf{2}} M' : \sigma$$

Demostración. Por inducción en la longitud de la derivación $M \twoheadrightarrow_w M'$.

□

Este resultado tiene como corolario la confluencia, pues se sabe que la relación $\to_w$ de CL es confluente y, al ser CL**2** cerrado por reducciones, dicha propiedad se mantiene en el fragmento de términos tipables del conjunto *CL*.

Veamos ahora que también puede derivarse la propiedad de normalización fuerte para términos tipables a partir de la correspondencia:

Proposición 3.4.3 (Strong Normalization)**.** *Sean $M \in CL$. Luego,*

$$\exists \Gamma\ \exists\, \sigma \in \mathcal{T}\ \ [\Gamma \vdash_{\mathrm{CL2}} M : \sigma] \Rightarrow M \text{ es SN}$$

Demostración. Supongamos que existe $M \in CL$ no SN, tal que

$$\exists \Gamma\ \exists\, \sigma \in \mathcal{T}\ \ [\Gamma \vdash_{\mathrm{CL2}} M : \sigma]$$

Luego, por la Proposición 3.4.1 (1)

$$\Gamma \vdash_{F_\eta} M_\Lambda : \sigma$$

y por SN de System F_η, M_Λ es SN.
Por otro lado, M no SN implica que

$$\exists\, M_1 \ldots M_n \ldots \in CL\ \ [M \to_w M_1 \to_w \cdots \to_w M_n \to_w \cdots]$$

se tiene una derivación infinita a partir de M. Luego, por la Proposición 1.2.2

$$M_\Lambda \xrightarrow{+}_\beta (M_1)_\Lambda \xrightarrow{+}_\beta \cdots \xrightarrow{+}_\beta (M_n)_\Lambda \xrightarrow{+}_\beta \cdots$$

se tiene una derivación infinita a partir de M_Λ, lo cual es absurdo pues M_Λ es SN.

□

Otro resultado que se tiene como consecuencia de la equivalencia es la consistencia del sistema:

Proposición 3.4.4 (Consistencia de CL**2**)**.** *Existe un tipo σ tal que para ningún término M se tiene $\vdash_{\mathrm{CL2}} M : \sigma$.*

Demostración. Veamos primero que System F_η resulta consistente. Como se comentó anteriormente, el conjunto de tipos habitados para este sistema es el mismo que para **λ2**. De aquí se deduce automáticamente la consistencia pues, como se vio en la prueba de la Proposición 2.3.4, no existe término $M \in \Lambda$ tal que $\vdash_{\lambda 2} M : \alpha$, con $\alpha \in \mathcal{V}$. Luego, α tampoco es habitado en System F_η.

Supongamos ahora que existe $M \in CL$ tal que $\vdash_{\mathrm{CL2}} M : \alpha$. Por la Proposición 3.4.1 (1) se tiene $\vdash_{F_\eta} M_\Lambda : \alpha$, lo cual es absurdo por lo expuesto anteriormente. Esto prueba la consistencia de CL**2**.

□

Una noción sobre los tipos que resulta de interés es la de *tipos principales*. Ésta se refiere a la existencia, dado un término cerrado y tipable M, de un tipo σ que representa todos los posibles tipos asignables a M. Es decir,

$\forall M$ cerrado y tipable

$$\exists\, \sigma \in \mathcal{T}\ \ \Big[\forall\, \tau \in \mathcal{T}\ \ \big[\vdash M : \tau \Rightarrow \exists \text{ sustitución } S\ \ [\ \tau \equiv S(\sigma)\ ]\big]\Big]$$

Un resultado conocido en la literatura es la falta de tipos principales para algunos términos tipables en **λ2** y System F_η [25, 23]. Lo mismo puede afirmarse sobre CL**2** en virtud de la correspondencia demostrada en la Proposición 3.4.1.

Por último, es interesante destacar que el problema del subtipado, para tipos polimórficos de segundo orden, es indecidible [30]. Esto es, dados dos tipos σ y τ, concluir si $\sigma \leq \tau$ (σ es subtipo de τ) no es computable. En [30] los autores analizan este problema independientemente del cálculo sobre el cual se aplica la relación de subtipado, para términar concluyendo que *type-checking* de System F_η no es decidible. Veamos que esto mismo puede argumentarse para CL**2**.

Type-checking es el problema en el cual, dados un contexto Γ, un término M y un tipo σ, se desea determinar si el juicio $\Gamma \vdash_{\mathrm{CL2}} M : \sigma$ es derivable. Es inmediato ver que puede concluirse $\sigma \leq \tau$ a partir del juicio de tipado $x : \sigma \vdash_{\mathrm{CL2}} x : \tau$, pues, por Lema 3.4.2 (1):

$$\exists\, \sigma' \leq \tau \ \left[x : \sigma' \in \{x : \sigma\}\right]$$

luego $\sigma \equiv \sigma' \leq \tau$. Entonces, de ser computable *type-checking*, también debería serlo el problema del subtipado.

3.5. Discusión

A lo largo del presente capítulo analizamos las diferentes opciones y problemáticas que se presentan al intentar formalizar un sistema de tipos de segundo orden para CL que resulta equivalente al ya conocido **λ2**. Esta noción de equivalencia buscada pretendía que el tipo de un término dado, en alguno de los dos sistemas, se mantuviese al traducir dicho término al otro sistema.

Vimos que el principal problema al intentar definir un sistema sobre Lógica Combinatoria que cumpla esta correspondencia, es que la noción de extensionalidad no es capturada por este formalismo, al mismo tiempo que ésta juega un papel importante en λ-cálculo.

Para subsanar este inconveniente, buscamos agregar a **λ2** las condiciones necesarias para que la η-reducción no resulte un problema, pues el sistema no posee la propiedad de *η-Subject Reduction*. Como consecuencia, la equivalencia fue finalmente planteada entre nuestro sistema CL**2** y System F_η, presentado por Mitchell en [24].

De este modo, se planteó CL**2** con una relación de subtipado para tipos polimórficos de segundo orden, del mismo modo que Dezani-Ciancaglini y Hindley expusieron una versión de tipos intersección para CL en [15].

Con esta formalización del sistema CL**2** fue posible demostrar que se cumple la equivalencia buscada y, al mismo tiempo, exhibir que el mismo cumple con las propiedades principales que se desea tenga todo sistema de tipos.

Por otro lado, queda abierto el planteo de una versión completa del lema de generación para el sistema, que contemple los casos de los combinadores **S** y **K**. Creemos que ésta permitiría presentar una prueba de la propiedad de clausura bajo reducción independiente de la correspondencia con System F_η, y que no

dependa de la traducción entre términos de λ-cálculo y CL utilizada para probar dicha equivalencia.

Capítulo 4

Conclusiones

En esta tesis hemos estudiado distintas formulaciones de sistemas de tipos para λ-cálculo y CL, analizando sus propiedades principales. Entre estas últimas se destaca la posibilidad de derivar la consistencia de la lógica asociada a cada sistema, vía el isomorfismo de Curry–Howard, a partir de la propiedad de *Strong Normalization* de los términos tipables. Nuestro trabajo se vio motivado por conseguir formalizar un método que pueda derivar la consistencia buscada sin necesidad del uso de propiedades fuertes del sistema de tipos asociado. Por otro lado, se planteó un sistema de tipos de segundo orden para CL, dado que en la literatura no se propone dicha variante tipada para el formalismo.

En primer lugar presentamos un método original para probar la consistencia de diferentes sistemas de tipo para λ-cálculo, que se basa en el análisis de las variables de tipos a lo largo de la inferencia de los juicios de tipado, utilizando una noción de variable positiva específica para cada sistema. A priori, la ventaja de esta propuesta radica en su sencillez, pues busca demostrar la consistencia mediante el análisis sintáctico de los tipos, sin involucrar propiedades de los sistemas que suelen ser difíciles de demostrar o pueden no estar disponibles.

Vimos que el método es perfectamente aplicable a sistemas de primer orden como $\boldsymbol{\lambda}\!\!\rightarrow$ y $\boldsymbol{\lambda}\!\!\cap$, y que su complejidad está estrechamente relacionada con la axiomatización utilizada para definir a éstos. Esto último resulta esperable pues se trata de un enfoque sintáctico que sólo involucra a las variables de los tipos, analizando su posición dentro de los mismos.

Al intentar extender el planteo a sistemas de alto orden, como $\boldsymbol{\lambda 2}$, la dificultad en la aplicación del método creció considerablemente, al punto de no llegar a plantear una prueba completa de la propiedad de invarianza positiva. Sin embargo, vimos que el método mantiene su simplicidad al considerar distintos subsistemas de $\boldsymbol{\lambda 2}$, cada uno de los cuales requería de una definición propia para el conjunto de variables positivas.

Es por esto que consideramos que el método propuesto no resulta conveniente para sistemas como $\boldsymbol{\lambda 2}$, y atribuimos este hecho a la noción de variable positiva utilizada. Creemos puntualmente que la interpretación dada al contructor de

tipos funcionales ($\rightarrow$) no es lo suficientemente precisa como para lograr el objetivo buscado, ya que es justamente esta noción la que dificulta la demostración de propiedades auxiliares requeridas para la prueba de la propiedad de invarianza. Entendemos que esta idea puede refinarse si se interpreta al constructor funcional como la implicación lógica, es decir:

$$\sigma \rightarrow \tau = \neg\sigma \vee \tau$$

Esto daría lugar a que variables que en la interpretación original resultaban no ser positivas ahora lo sean, pues $\neg(\neg\alpha) = \alpha$.

Por otro lado, buscamos hacer extensivo el análisis a los *Pure Type Systems*, más precisamente a los sistemas del λ-cubo, concluyendo en que no es posible tal planteo, pues los distintos PTS's se formalizan sobre una axiomatización uniforme que no distingue entre términos y tipos, siendo la única diferencia entre cada sistema sus conjuntos de sorts, axiomas y reglas. Estos últimos sólo involucran constantes de los sistemas, por lo que no es posible distinguir entre un PTS y otro desde el punto de vista de las variables, mostrando que el método no resulta aplicable, pues se sabe que existen PTS's inconsistentes como λ^*.

Estos resultados nos llevan a concluir que el método propuesto resulta interesante por su simplicidad y facilidad de aplicación en sistemas de bajo orden, pero al mismo tiempo es limitado al considerar sistemas de alto orden. De aquí resulta que no sea suficiente con analizar sintácticamente las variables de un sistema de tipos para demostrar su consistencia, pues muchas veces la semántica esconde un poder expresivo y propiedades que escapan al alcance del método propuesto.

Una posible línea de trabajo futuro es la de plantear un método que incorpore a las constantes en el análisis, al mismo tiempo que refine la noción utilizada, teniendo en cuenta también la cantidad de apariciones de cada variable, o incluso introduciendo la noción de *variables negadas*. De este modo puede incorporarse al análisis la polaridad de las variables dentro del tipo, como comentamos anteriormente. Todas estas propuestas fueron surgiendo a lo largo del estudio del método, si bien aún no hay una idea clara de cómo pueden contribuir a la mejora del mismo.

Como segundo tema propuesto en el presente trabajo, se estudió la posibilidad de extender el sistema de tipos simples de CL para incorporar polimorfismo paramétrico. De este modo se buscó formalizar un sistema de tipos de segundo orden que resultase equivalente a **λ2** bajo la noción propuesta.

Vimos que el principal problema al intentar definir dicho sistema es que la noción de extensionalidad no es capturada por CL, al mismo tiempo que ésta juega un papel importante en λ-cálculo. Para subsanar este inconveniente, buscamos agregar a **λ2** las condiciones necesarias para que la η-reducción no resulte un problema, pues el sistema no posee la propiedad de *η-Subject Reduction*. Como consecuencia, la equivalencia fue finalmente planteada entre nuestro sistema CL**2** y System F_η, el cual resulta de agregar a **λ2** la regla de tipado para η-reductos.

Una vez definido formalmente el sistema y probada la equivalencia, se analizaron sus propiedades principales. Este último estudio permitió demostrar que

CL**2** cumple con la propiedad de *Subject Reduction* y *Strong Normalization* de términos tipables. También se pudo ver que el problema de *type-checking* resulta indecidible, pues la misma relación de subtipado introducida para el sistema lo es.

Otro resultado negativo es el hecho de que existen términos tipables en CL**2** que no poseen un tipo principal, al igual que sucede en System F_η. Planteos como los expuestos en [25, 23] sobre λ-cálculo buscan mejorar esto último incorporando tipos intersección al sistema de segundo orden. De aquí surge la idea de extender el sistema propuesto del mismo modo. Esto resulta en una línea de trabajo futuro sobre sistemas de tipos para CL, a partir del desarrollado en esta tesis.

Por otro lado, queda abierto el análisis de otras propiedades sobre CL**2**, así como el planteo de una versión completa del lema de generación para el sistema, que contemple los casos de los combinadores **S** y **K**.

Una consecuencia interesante de la equivalencia conseguida entre CL**2** y System F_η es que, contrariamente a lo que podría indicar la intuición, el hecho de que el sistema de reescritura CL es estrictamente más débil que el λ-cálculo, no impide que los términos tipables sean, módulo traducción, los mismos. Más aún, con los mismo tipos. Esta diferencia de poderes expresivos entre los sistemas de reescritura, no se manifiesta, por lo visto, en la equivalencia entre los respectivos sistemas de tipos. Esto pasa tanto para tipado simple, como para tipado de segundo orden.

El presente trabajo no pretende ser exhaustivo, y puede extenderse el planteo a otras versiones de CL, por ejemplo aquellas que utilizan otros combinadores distintos de los que hemos utilizado aquí. Resulta interesante analizar el sistema CL**2** producto de las conocidas reformulaciones de CL, correspondientes a la lógica de relevancia, la lógica de hipótesis descartables y la lógica lineal [17]. Para estas restricciones de la lógica intuicionista, se definen otros conjuntos de combinadores, los cuales codifican las pruebas de las fórmulas demostrables en cada una de esas lógicas.

La intuición dice que el sistema CL**2** se puede reformular para estas tres restricciones y, si bien no lo hemos desarrollado en esta tesis, entendemos que el camino es directo: los combinadores presentes tendrán reglas de tipado análogas a las de **S** y **K** (derivadas del tipo de dichos combinadores en λ-cálculo), y se mantendrán las reglas $(\forall I)$ y $(\leq)$, resultando así en un sistema del mismo estilo que CL**2** y que goza de buenas propiedades de traducción.

Apéndice A

Subsistemas de $\boldsymbol{\lambda 2}$

En el presente apéndice estudiamos los subsistemas de $\boldsymbol{\lambda 2}$ propuestos en la sección 2.3. El propósito es demostrar para cada uno de ellos que cumple con la propiedad principal de clausura que es deseable en todo sistema de tipos.

En las secciones subsiguientes presentamos las reglas de cada sistema a analizar y desarrollamos los distintos resultados preliminares de los que haremos uso para terminar probando la propiedad SR.

A.1. El sistema $\boldsymbol{\lambda 2}^i$

Este sistema surge de considerar el sistema de tipos de segundo orden $\boldsymbol{\lambda 2}$ sin la regla $(\forall E)$.

$$\frac{}{\Gamma, x : \sigma \vdash_i x : \sigma}\ (axiom) \qquad \frac{\Gamma \vdash_i M : \sigma \quad \alpha \notin \mathcal{FV}(\Gamma)}{\Gamma \vdash_i M : \forall \alpha.\sigma}\ (\forall I)$$

$$\frac{\Gamma, x : \sigma \vdash_i M : \tau}{\Gamma \vdash_i \lambda x.M : \sigma \to \tau}\ (\to I) \qquad \frac{\Gamma \vdash_i M : \sigma \to \tau \quad \Gamma \vdash_i N : \sigma}{\Gamma \vdash_i MN : \tau}\ (\to E)$$

Puede verse que la expresividad respecto a CL$\mathbf{2}$ es menor, dada la imposibilidad de eliminar los cuantificadores introducidos con la regla $(\forall I)$ o provenientes del contexto. Para ilustrar esto, veamos un ejemplo, del cual puede demostrarse su validez mediante el Lema A.1.3 (1) presentado en la siguiente sección:

$$x : \forall \alpha.\alpha \not\vdash_i x : \alpha \to \alpha \qquad\qquad x : \forall \alpha.\alpha \vdash_{\boldsymbol{\lambda 2}} x : \alpha \to \alpha$$

Sin embargo, el poder expresivo del sistema es claramente mayor al de $\boldsymbol{\lambda}\!\to$, y pueden lograrse aún juicios de tipado que despiertan cierto interés, como por ejemplo:

$$\begin{aligned} x : \alpha &\vdash_i x : \alpha \\ &\vdash_i \lambda x.x : \alpha \to \alpha \\ &\vdash_i \lambda x.x : \forall \alpha.(\alpha \to \alpha) \end{aligned}$$

Esto último motiva el planteo del sistema como una restricción apropiada de **λ2**, y el análisis de sus propiedades principales.

A.1.1. Definiciones y lemas auxiliares

Antes de pasar a la demostración de la propiedad de clausura, presentamos una serie de lemas a los cuales recurriremos al momento de formalizar dicha prueba.

El siguiente lema permite trabajar con las hipótesis incluidas en el contexto de tipado, pudiendo agregar o eliminar las mismas bajo determinadas condiciones:

Lema A.1.1 (Base Lemma)**.** *Sea* Γ *una base.*

1. *Si* $\Delta \supseteq \Gamma$ *es otra base, entonces* $\Gamma \vdash_i M : \delta \Rightarrow \Delta \vdash_i M : \delta$.
2. $\Gamma \vdash_i M : \delta \Rightarrow \mathsf{FV}(M) \subseteq \mathrm{dom}(\Gamma)$.
3. $\Gamma \vdash_i M : \delta \Rightarrow \Gamma\restriction_{\mathsf{FV}(M)} \vdash_i M : \delta$.

Demostración. Por inducción en la derivación de $\Gamma \vdash_i M : \delta$, analizando la última regla aplicada. El análisis por regla es análogo al presentado en la prueba del Lema 3.4.1 para los casos $(axiom)$, $(\to E)$ y $(\forall I)$, por lo que sólo se presentan a continuación los casos de aplicación de $(\to I)$ para cada ítem del lema. Luego, se tiene $M \equiv \lambda x.M'$ con $\delta \equiv \sigma \to \tau$ y por hipótesis de la regla vale:

$$\Gamma, x : \sigma \vdash_i M' : \tau$$

1. Por convención de variables libres, podemos asumir $x \notin \mathrm{dom}(\Delta)$ de modo que $\Delta, x : \sigma$ resulta una base y, por hipótesis inductiva

   $$\Delta, x : \sigma \vdash_i M' : \tau$$

   Luego, por $(\to I)$ se concluye

   $$\Delta \vdash_i \lambda x.M' : \sigma \to \tau$$

2. Por definición

   $$\mathsf{FV}(\lambda x.M') = \mathsf{FV}(M') - \{x\}$$

   Por hipótesis inductiva vale

   $$\mathsf{FV}(M') \subseteq \mathrm{dom}(\Gamma, x : \sigma) = \mathrm{dom}(\Gamma) \cup \{x\}$$

   Luego, restando $\{x\}$ a ambos lados de la desigualdad

   $$\mathsf{FV}(\lambda x.M') = \mathsf{FV}(M') - \{x\} \subseteq \big(\mathrm{dom}(\Gamma) \cup \{x\}\big) - \{x\} = \mathrm{dom}(\Gamma)$$

3. Por hipótesis inductiva

$$(\Gamma, x:\sigma)\restriction_{\mathsf{FV}(M')} \vdash_i M' : \tau$$

Si $x \notin \mathsf{FV}(M')$, entonces $(\Gamma, x:\sigma)\restriction_{\mathsf{FV}(M')} = \Gamma\restriction_{\mathsf{FV}(M')}$ y por (1) vale

$$\Gamma\restriction_{\mathsf{FV}(M')}, x:\sigma \vdash_i M' : \tau$$

Además, se tiene $\mathsf{FV}(M') = \mathsf{FV}(\lambda x.M')$. Luego, por $(\to I)$

$$\Gamma\restriction_{\mathsf{FV}(\lambda x.M')} \vdash_i \lambda x.M' : \sigma \to \tau$$

Si $x \in \mathsf{FV}(M')$, se tiene $(\Gamma, x:\sigma)\restriction_{\mathsf{FV}(M')} = (\Gamma\restriction_{\mathsf{FV}(M')}, x:\sigma)$ Además

$$\Gamma\restriction_{\mathsf{FV}(M')} = \Gamma\restriction_{\mathsf{FV}(\lambda x.M')}$$

pues $\mathsf{FV}(\lambda x.M') = \mathsf{FV}(M') - \{x\}$ y $x \notin \mathrm{dom}(\Gamma)$. Luego, por $(\to I)$

$$\Gamma\restriction_{\mathsf{FV}(\lambda x.M')} \vdash_i \lambda x.M' : \sigma \to \tau$$

□

El siguiente lema permite analizar como afectan las sustituciones, tanto de términos como de tipos, sobre los juicios de tipado. El fin es poder demostrar que la operación de substitución introducida por la β-reducción preserva el tipo del término en cuestión.

Lema A.1.2 (Substitution Lemma)**.**

1. $\Gamma \vdash_i M : \sigma \Rightarrow \Gamma[\alpha := \tau] \vdash_i M : \sigma[\alpha := \tau]$.
2. *Si* $\Gamma, x:\sigma \vdash_i M : \tau$ *y* $\Gamma \vdash_i N : \sigma$*, entonces* $\Gamma \vdash_e M[N/x] : \tau$.

Demostración.

1. Por inducción en la derivación de $\Gamma \vdash_i M : \sigma$.

   Caso 1: Se aplicó (*axiom*), por lo que $M \equiv x$ y $\Gamma = \Gamma', x:\sigma$. Al aplicar la sustitución al contexto queda $\Gamma[\alpha := \tau] = \big(\Gamma'[\alpha := \tau], x:\sigma[\alpha := \tau]\big)$, de lo que se deduce por (*axiom*)

   $$\Gamma'[\alpha := \tau], x:\sigma[\alpha := \tau] \vdash_i M : \sigma[\alpha := \tau]$$

   Caso 2: Si se aplicó $(\to E)$ se tiene $M \equiv M_1 M_2$ tal que

   $$\Gamma \vdash_i M_1 : \rho \to \sigma \quad \text{y} \quad \Gamma \vdash_i M_2 : \rho$$

   Por hipótesis inductiva

   $$\Gamma[\alpha := \tau] \vdash_i M_1 : (\rho \to \sigma)[\alpha := \tau] \quad \text{y} \quad \Gamma[\alpha := \tau] \vdash_i M_2 : \rho[\alpha := \tau]$$

   Como $(\rho \to \sigma)[\alpha := \tau] = \rho[\alpha := \tau] \to \sigma[\alpha := \tau]$, por $(\to E)$

   $$\Gamma[\alpha := \tau] \vdash_i M_1 M_2 : \sigma[\alpha := \tau]$$

Caso 3: La última regla usada fue $(\to I)$. Entonces $M \equiv \lambda x.M'$, $\sigma \equiv \rho \to \delta$ y vale

$$\Gamma, x : \rho \vdash_i M' : \delta$$

Por hipótesis inductiva

$$\Gamma[\alpha := \tau], x : \rho[\alpha := \tau] \vdash_i M' : \delta[\alpha := \tau]$$

Luego, aplicando $(\to I)$

$$\Gamma[\alpha := \tau] \vdash_i \lambda x.M' : (\rho \to \delta)[\alpha := \tau]$$

Caso 4: Si la última regla aplicada fue $(\forall I)$, se tiene $\sigma \equiv \forall \beta.\sigma'$ con $\beta \notin \mathcal{FV}(\Gamma)$ y vale

$$\Gamma \vdash_i M : \sigma'$$

Por convención de variables podemos asumir $\beta \neq \alpha$ y $\beta \notin \mathcal{FV}(\tau)$. Por hipótesis inductiva

$$\Gamma[\alpha := \tau] \vdash_i M : \sigma'[\alpha := \tau]$$

Luego, por $(\forall I)$ se tiene

$$\Gamma[\alpha := \tau] \vdash_i M : (\forall \beta.\sigma')[\alpha := \tau]$$

pues $(\forall \beta.\sigma')[\alpha := \tau] = \forall \beta.\sigma'[\alpha := \tau]$.

2. Por inducción en la derivación de $\Gamma, x : \sigma \vdash_i M : \tau$.

Caso 1: Si se aplicó $(axiom)$, pueden darse dos casos. Si $M \equiv x$, entonces $x[N/x] = N$, $\sigma = \tau$ y, por hipótesis, vale

$$\Gamma \vdash_i N : \sigma$$

Si $M \equiv y \neq x$, luego $y[N/x] = y$ y la implicación vale trivialmente pues, por hipótesis, se tiene

$$\Gamma, y : \sigma \vdash_i y : \sigma$$

Caso 2: Se aplicó $(\to E)$. Entonces $M \equiv M_1 M_2$ y por hipótesis de la regla valen

$$\Gamma, x : \sigma \vdash_i M_1 : \rho \to \sigma \quad \text{y} \quad \Gamma, x : \sigma \vdash_i M_2 : \rho$$

Por hipótesis inductiva

$$\Gamma \vdash_i M_1[N/x] : \rho \to \sigma \quad \text{y} \quad \Gamma \vdash_i M_2[N/x] : \rho$$

Luego por $(\to E)$ se concluye

$$\Gamma \vdash_i (M_1 M_2)[N/x] : \sigma$$

Caso 3: La última regla usada fue $(\to I)$. Se tiene $M \equiv \lambda y.M'$ con $\sigma \equiv \rho \to \delta$. Por convención de variables asumimos $y \neq x$ e $y \notin \mathsf{FV}(N)$. Por precondición de la regla vale

$$\Gamma, x : \sigma, y : \rho \vdash_i M' : \delta$$

Luego, por hipótesis inductiva

$$\Gamma, y : \rho \vdash_i M'[N/x] : \delta$$

Como $y \notin \mathsf{FV}(N)$ vale $\lambda y.M'[N/x] = (\lambda y.M')[N/x]$. Luego, aplicando $(\to I)$ se tiene

$$\Gamma \vdash_i (\lambda y.M')[N/x] : \rho \to \delta$$

Caso 4: Si se usó $(\forall I)$, se tiene $\sigma \equiv \forall\beta.\sigma'$ con $\beta \notin \mathcal{FV}(\Gamma, x : \sigma)$ y vale

$$\Gamma, x : \sigma \vdash_i M : \sigma'$$

Por hipótesis inductiva

$$\Gamma \vdash_i M[N/x] : \sigma'$$

Como $\beta \notin \mathcal{FV}(\Gamma, x : \sigma) = \mathcal{FV}(\Gamma) \cup \mathcal{FV}(\sigma)$, en particular vale $\beta \notin \mathcal{FV}(\Gamma)$. Luego, por $(\forall I)$

$$\Gamma \vdash_i M[N/x] : \forall\beta.\sigma'$$

□

A continuación nos proponemos enunciar y demostrar un lema de generación apropiado para el sistema. Para ello introducimos la siguiente relación de orden entre tipos:

Definición A.1.1.

1. $\sigma > \tau$ *sii* $\tau \equiv \forall\alpha.\sigma$ *para algún* $\alpha \in \mathcal{V}$.
2. $\geq$ *es la clausura reflexiva y transitiva de* $>$.

Una vez introducida esta noción de orden entre los tipos, presentamos el lema de generación de la siguiente manera:

Lema A.1.3 (Generation Lemma)**.**

1. $\Gamma \vdash_i x : \sigma \Rightarrow \exists\, \sigma' \geq \sigma \;\; \big[(x : \sigma') \in \Gamma\big]$.
2. $\Gamma \vdash_i MN : \tau \Rightarrow \exists\sigma \exists\, \tau' \geq \tau \;\; [\Gamma \vdash_i M : \sigma \to \tau' \quad y \quad \Gamma \vdash_i N : \sigma]$.
3. $\Gamma \vdash_i \lambda x.M : \rho \Rightarrow \exists\, \sigma, \tau \;\; [\Gamma, x : \sigma \vdash_i M : \tau \quad y \quad \sigma \to \tau \geq \rho]$.

Demostración. Por inducción en la derivación del juicio de tipado, analizando las reglas que aplican en cada caso.

1. Para el tipado de variables sólo aplican las reglas ($axiom$) y ($\forall I$).

   Caso 1: Se usó ($axiom$). Luego vale $\Gamma, x : \sigma \vdash_i x : \sigma$, por lo que basta tomar $\sigma' = \sigma$.

   Caso 2: Se aplicó ($\forall I$). Entonces $\sigma \equiv \forall\alpha.\sigma_1$ con $\alpha \notin \mathcal{FV}(\Gamma)$ y vale

   $$\Gamma \vdash_i x : \sigma_1$$

   Por hipótesis inductiva

   $$\exists\, \sigma' \geq \sigma_1 \;\; \big[(x : \sigma') \in \Gamma\big]$$

   Como $\sigma_1 > \forall\alpha.\sigma_1 = \sigma$, por transitividad $\sigma' \geq \sigma$.

2. En el caso de las aplicaciones, sólo intervienen las reglas ($\to E$) y ($\forall I$).

   Caso 1: Si se usó ($\to E$), entonces valen

   $$\Gamma \vdash_i M : \sigma \to \tau \quad \text{y} \quad \Gamma \vdash_i N : \sigma$$

   Luego basta tomar $\tau' = \tau$.

   Caso 2: Se aplicó ($\forall I$), el razonamiento es análogo al ilustrado en el **Caso 2** del ítem (1), resultando la existencia el tipo τ' buscado por transitividad a partir de la hipótesis inductiva.

3. El tipado de abstracciones solamente involucra las reglas ($\to I$) y ($\forall I$). El caso de esta última es análogo al desarrollado en el ítem (1), mientras que para ($\to I$) se tiene $\rho \equiv \sigma \to \tau$ tal que $\Gamma, x : \sigma \vdash_i M : \tau$.

□

Demostramos un último lema auxiliar sobre la relación de orden $\geq$ que nos será de utilidad para la prueba que presentaremos en la sección siguiente:

Lema A.1.4.

1. $\sigma \geq \tau \Rightarrow \tau \equiv \forall\alpha_n \ldots \alpha_1.\sigma$ *con* $n \geq 0$.
2. $\sigma \to \tau \geq \sigma' \to \tau' \Rightarrow \sigma \to \tau \equiv \sigma' \to \tau'$.

Demostración.

1. Por inducción en la prueba de $\sigma \geq \tau$.

   Caso base: $\sigma = \tau$. Tomamos $n = 0$.

Paso inductivo: $\sigma \geq \sigma' > \tau$. Por hipótesis inductiva

$$\sigma' \equiv \forall \alpha_n \ldots \alpha_1 . \sigma$$

Pero por definición de $>$

$$\tau \equiv \forall \alpha_{n+1} . \sigma' \equiv \forall \alpha_{n+1} \ldots \alpha_1 . \sigma$$

2. Sean $\sigma \to \tau \geq \sigma' \to \tau'$. Por (1),

$$\sigma' \to \tau' \equiv \forall \alpha_n \ldots \alpha_1 . (\sigma \to \tau)$$

Necesariamente se tiene $n = 0$, pues $\sigma' \to \tau'$ es un tipo funcional.

□

A.1.2. Propiedad de clausura

Ahora estamos en condiciones de demostrar la propiedad de clausura bajo reescritura del sistema $\boldsymbol{\lambda 2}^i$, haciendo uso de los resultados presentados en la sección anterior.

Proposición A.1.1 (Subject Reduction)**.** *Sean $M, M' \in \Lambda$ tal que $M \to_\beta M'$. Entonces*

$$\Gamma \vdash_i M : \sigma \Rightarrow \Gamma \vdash_i M' : \sigma$$

Demostración. Por inducción en el término M, analizando el lugar donde se produce la reducción.

Caso 1: $M \equiv x$. Si el término es una variable no existe M' tal que $x \to_\beta M'$ por lo que la implicación vale trivialmente.

Caso 2: $M \equiv \lambda x.N$. La reducción se produce en N, de modo que $M' \equiv \lambda x.N'$ con $N \to_\beta N'$. Luego, $\Gamma \vdash_i M : \sigma$ implica, por Lema A.1.3 (3)

$$\exists \sigma', \tau' \ [\Gamma, x : \sigma' \vdash_i N : \tau' \quad \text{y} \quad \sigma' \to \tau' \geq \sigma]$$

Por hipótesis inductiva

$$\Gamma, x : \sigma' \vdash_i N' : \tau'$$

Luego, por $(\to I)$

$$\Gamma \vdash_i \lambda x.N' : \sigma' \to \tau'$$

Además, por Lema A.1.4 (1) se tiene $\sigma \equiv \forall \alpha_n \ldots \alpha_1 . (\sigma' \to \tau')$ para $n \geq 0$. Por convención de variables podemos asumir que $\alpha_i \notin \mathcal{FV}(\Gamma)$ para cada $1 \geq i \geq n$. Luego, aplicando n veces la regla $(\forall I)$ se concluye

$$\Gamma \vdash_i \lambda x.N' : \sigma$$

Caso 3: Consideramos el caso en que $M \equiv M_1 M_2$ y la reducción se produce internamente en M_1, quedando $M' \equiv M_1' M_2$ con $M_1 \to_\beta M_1'$. Por Lema A.1.3 (2)

$$\exists \sigma' \exists \tau' \geq \sigma \;\; [\Gamma \vdash_i M_1 : \sigma' \to \tau' \quad \text{y} \quad \Gamma \vdash_i M_2 : \sigma']$$

Por hipótesis inductiva

$$\Gamma \vdash_i M_1' : \sigma' \to \tau'$$

Luego, por $(\to E)$

$$\Gamma \vdash_i M_1' M_2 : \tau'$$

Nuevamente, por el argumento presentado en el caso anterior, podemos aplicar $(\forall I)$ la cantidad de veces necesaria para terminar concluyendo

$$\Gamma \vdash_i M_1' M_2 : \sigma$$

El caso en que la reducción se produce en M_2 es completamente análogo a este último.

Caso 4: Falta analizar el caso en que M es una aplicación con $M_1 \equiv \lambda x.P$ produciéndose la reducción en la raíz del término. Es decir que $M' \equiv P[M_2/x]$. Por Lema A.1.3 (2) sobre el juicio de tipado de M se tiene

$$\exists \sigma' \exists \tau' \geq \sigma \;\; [\Gamma \vdash_i \lambda x.P : \sigma' \to \tau' \quad \text{y} \quad \Gamma \vdash_i M_2 : \sigma']$$

Ahora por Lema A.1.3 (3) vale

$$\exists \, \sigma'', \tau'' \;\; [\Gamma, x : \sigma'' \vdash_i P : \tau'' \quad \text{y} \quad \sigma'' \to \tau'' \geq \sigma' \to \tau']$$

Por Lema A.1.4 (2) $\sigma' \to \tau' \equiv \sigma'' \to \tau''$, lo que implica $\sigma' \equiv \sigma''$ y $\tau' \equiv \tau''$. Entonces

$$\Gamma, x : \sigma' \vdash_i P : \tau' \quad \text{y} \quad \Gamma \vdash_i M_2 : \sigma'$$

Luego, por Lema A.1.2 (2)

$$\Gamma \vdash_i P[M_2/x] : \tau'$$

Nuevamente estamos ante las hipótesis del argumento usado en los casos anteriores para concluir

$$\Gamma \vdash_i P[M_2/x] : \sigma$$

□

Como corolario de la preservación del tipo en un paso de reducción, se tiene la propiedad de clausura para varios pasos:

Corolario A.1.1. *Sean $M, M' \in CL$ tal que $M \twoheadrightarrow_\beta M'$. Entonces*

$$\Gamma \vdash_i M : \sigma \Rightarrow \Gamma \vdash_i M' : \sigma$$

Demostración. Por inducción en la longitud de la derivación $M \twoheadrightarrow_\beta M'$.

□

Este resultado nos muestra que $\boldsymbol{\lambda 2}^i$ cumple con la propiedad principal que se pide que tenga todo sistema de tipos para ser considerado apropiado. Además, como se ilustró anteriormente, el sistema tiene un poder expresivo que se encuentra entre $\boldsymbol{\lambda}\!\!\rightarrow$ y $\boldsymbol{\lambda 2}$. Estos dos argumentos justifican su estudio.

A.2. El sistema $\boldsymbol{\lambda 2}^e$

Analizamos ahora la otra restricción planteada sobre $\boldsymbol{\lambda 2}$, obtenida al eliminar del sistema la regla $(\forall I)$.

$$\frac{}{\Gamma, x:\sigma \vdash_e x:\sigma}\ (axiom) \qquad \frac{\Gamma \vdash_e M:\forall\alpha.\sigma}{\Gamma \vdash_e M:\sigma[\alpha := \tau]}\ (\forall E)$$

$$\frac{\Gamma, x:\sigma \vdash_e M:\tau}{\Gamma \vdash_e \lambda x.M:\sigma\to\tau}\ (\to I) \qquad \frac{\Gamma \vdash_e M:\sigma\to\tau \quad \Gamma \vdash_e N:\sigma}{\Gamma \vdash_e MN:\tau}\ (\to E)$$

De este modo, la expresividad respecto a $\boldsymbol{\lambda 2}$ se ve claramente disminuida, pues ya no es derivable, por ejemplo, el juicio que asigna a la identidad el tipo $\forall\alpha.(\alpha\to\alpha)$. Más aún, veremos que todo tipo derivable para un término cuyo constructor principal sea un abstracción debe necesariamente ser un tipo funcional.

Sin embargo, la expresividad respecto de $\boldsymbol{\lambda}\!\!\rightarrow$ es trivialmente mayor, pues el conjunto $\mathcal{T}$ posee más tipos posibles y al mismo tiempo pueden inferirse los siguientes juicios:

$$\begin{aligned} x:\forall\alpha.\alpha &\vdash_e x:\sigma \\ &\vdash_e \lambda x.x:\forall\alpha.\alpha\to\sigma \end{aligned}$$

para cualquier $\sigma\in\mathcal{T}$, en particular con $\sigma=\forall\alpha.\alpha$ se tiene:

$$\vdash_e \lambda x.x:\forall\alpha.\alpha\to\forall\alpha.\alpha$$

Nuevamente buscamos probar la propiedad de clausura bajo reducción, para lo cual haremos uso de los lemas presentados a continuación.

A.2.1. Definiciones y lemas auxiliares

Los próximos dos lemas son equivalentes a los introducidos al analizar el subsistema $\boldsymbol{\lambda 2}^i$, y sus demostraciones son completamente análogas para los casos de las reglas comunes a ambos sistemas. Por esta razón, enunciamos los lemas y sólo analizamos los casos en que la regla que se aplica es $(\forall E)$.

Lema A.2.1 (Base Lemma)**.** *Sea* Γ *una base.*

1. *Si* $\Delta \supseteq \Gamma$ *es otra base, entonces* $\Gamma \vdash_e M : \delta \Rightarrow \Delta \vdash_e M : \delta$.
2. $\Gamma \vdash_e M : \delta \Rightarrow \mathsf{FV}(M) \subseteq \mathrm{dom}(\Gamma)$.
3. $\Gamma \vdash_e M : \delta \Rightarrow \Gamma\restriction_{\mathsf{FV}(M)} \vdash_e M : \delta$.

Demostración. Por inducción en la derivación de $\Gamma \vdash_e M : \delta$.

1. Se tiene $\delta \equiv \delta'[\alpha := \tau]$ y por hipótesis inductiva vale
$$\Delta \vdash_e M : \forall\alpha.\delta'$$
Luego, aplicando $(\forall E)$
$$\Delta \vdash_e M : \delta'[\alpha := \tau]$$
2. Directamente de la hipótesis inductiva se tiene $\mathsf{FV}(M) \subseteq \mathrm{dom}(\Gamma)$, pues la regla no modifica el término.
3. Por hipótesis inductiva se vale
$$\Gamma\restriction_{\mathsf{FV}(M)} \vdash_e M : \forall\alpha.\delta'$$
por lo que, aplicando $(\forall E)$ se deduce
$$\Gamma\restriction_{\mathsf{FV}(M)} \vdash_e M : \delta'[\alpha := \tau]$$
con $\delta \equiv \delta'[\alpha := \tau]$.

□

Lema A.2.2 (Substitution Lemma)**.**

1. $\Gamma \vdash_e M : \sigma \Rightarrow \Gamma[\alpha := \tau] \vdash_e M : \sigma[\alpha := \tau]$.
2. *Si* $\Gamma, x : \sigma \vdash_e M : \tau$ *y* $\Gamma \vdash_e N : \sigma$*, entonces* $\Gamma \vdash_e M[N/x] : \tau$.

Demostración.

1. Por inducción en la derivación de $\Gamma \vdash_e M : \sigma$.

   Para el caso en que se aplicó la regla $(\forall E)$ se tiene $\sigma \equiv \sigma'[\beta := \rho]$ y por hipótesis inductiva vale
   $$\Gamma[\alpha := \tau] \vdash_e M : \forall\beta.\sigma'[\alpha := \tau]$$
   pues podemos asumir $\beta \neq \alpha$ y $\beta \notin \mathcal{FV}(\tau)$ por convención de variables. Luego, por $(\forall E)$ se infiere
   $$\Gamma[\alpha := \tau] \vdash_e M : \sigma'[\alpha := \tau]\big[\beta := \rho[\alpha := \tau]\big]$$
   y finalmente por Lema 1.2.2 se tiene
   $$\Gamma[\alpha := \tau] \vdash_e M : \sigma'[\beta := \rho][\alpha := \tau]$$
   pues $\beta \notin \mathcal{FV}(\tau)$.

2. Por inducción en la derivación de $\Gamma, x : \sigma \vdash_e M : \tau$.

   Nuevamente se tiene $\tau \equiv \tau'[\beta := \rho]$ y por hipótesis inductiva vale

   $$\Gamma \vdash_e M[N/x] : \forall\beta.\tau'$$

   Luego, aplicando $(\forall E)$ se deduce

   $$\Gamma \vdash_e M[N/x] : \tau'[\beta := \rho]$$

□

Presentamos ahora una relación de orden sobre los tipos con el fin de introducir el lema de generación para el sistema.

Definición A.2.1.

1. $\sigma > \tau$ *sii* $\sigma \equiv \forall\alpha.\sigma_1$ *y* $\tau \equiv \sigma_1[\beta := \rho]$ *para algún* $\rho \in \mathcal{T}$.
2. $\geq$ *es la clausura reflexiva y transitiva de* $>$.

De este modo, planteamos el siguiente lema de generación. Es en esencia similar al presentado para $\boldsymbol{\lambda 2}$ en la sección 2.3 y para $\boldsymbol{\lambda 2}^i$ en este mismo apéndice, pero sufre una modificación en el caso del tipado de abstracciones.

Lema A.2.3 (Generation Lemma)**.**

1. $\Gamma \vdash_e x : \sigma \Rightarrow \exists\, \sigma' \geq \sigma \;\; \big[(x : \sigma') \in \Gamma\big]$.
2. $\Gamma \vdash_e MN : \tau \Rightarrow \exists\sigma \exists\, \tau' \geq \tau \;\; [\Gamma \vdash_e M : \sigma \to \tau' \quad y \quad \Gamma \vdash_e N : \sigma]$.
3. $\Gamma \vdash_e \lambda x.M : \rho \Rightarrow \exists\, \sigma, \tau \;\; [\Gamma, x : \sigma \vdash_e M : \tau \quad y \quad \rho \equiv \sigma \to \tau]$.

Demostración. Por inducción en la derivación del juicio de tipado, analizando las reglas que aplican en cada caso.

1. Para el tipado de variables sólo aplican las reglas $(axiom)$ y $(\forall E)$.

   Caso 1: Se usó $(axiom)$. Luego vale $\Gamma, x : \sigma \vdash_e x : \sigma$, por lo que basta tomar $\sigma' = \sigma$.

   Caso 2: Se aplicó $(\forall E)$. Entonces $\sigma \equiv \sigma_1[\alpha := \tau]$ y vale

   $$\Gamma \vdash_e x : \forall\alpha.\sigma_1$$

   Por hipótesis inductiva

   $$\exists\, \sigma' \geq \sigma_1 \;\; \big[(x : \sigma') \in \Gamma\big]$$

   Por definición $\forall\alpha.\sigma_1 > \sigma_1[\alpha := \tau] = \sigma$, luego, por transitividad, $\sigma' \geq \sigma$.

2. En el caso de las aplicaciones, sólo intervienen las reglas $(\to E)$ y $(\forall E)$.

Caso 1: Si se usó $(\rightarrow E)$, entonces valen

$$\Gamma \vdash_e M : \sigma \rightarrow \tau \quad \text{y} \quad \Gamma \vdash_e N : \sigma$$

Luego basta tomar $\tau' = \tau$.

Caso 2: Si la última regla aplicada fue $(\forall E)$, entonces $\tau \equiv \tau_1[\alpha := \rho]$ y por hipótesis inductiva se tiene

$$\exists \sigma \exists\, \tau' \geq \forall \alpha.\tau_1 \;\; [\Gamma \vdash_e M : \sigma \rightarrow \tau' \quad \text{y} \quad \Gamma \vdash_e N : \sigma]$$

pero nuevamente, por definición de $>$ y por transitividad, vale $\tau' \geq \tau_1[\alpha := \rho]$.

3. Veamos que el tipado de abstracciones sólo involucra la regla $(\rightarrow I)$. En primer lugar es inmediato ver que $(axiom)$ y $(\rightarrow E)$ no aplican, pues los términos que resultan de utilizar estas reglas son variables o aplicaciones respectivamente. Por otro lado, las abstracciones son introducidas mediante la aplicación de $(\rightarrow I)$, resultando en un tipo funcional. De aquí, y del hecho que no se pueden introducir cuantificadores por delante de un constructor de tipo flecha ya establecido, por no contar con la regla $(\forall I)$, resulta que $(\forall E)$ no interviene en el tipado de una abstracción, pues ésta asume tipo $\forall \alpha.\sigma$ para el término en cuestión.
   Luego, por hipótesis de la regla se tiene $\Gamma, x : \sigma \vdash_e M : \tau$ y resulta $\rho \equiv \sigma \rightarrow \tau$.

□

Es aquí donde se ve lo que comentamos al introducir el sistema sobre que un término cuyo constructor principal es una abstracción tiene necesariamente tipo funcional.

Como último resultado preliminar, veamos que si puede derivarse un tipo σ para un término dado, también pude derivarse τ, para todo tipo τ tal que $\sigma \geq \tau$.

Lema A.2.4. *Sean $\sigma, \tau \in \mathcal{T}$ tal que $\sigma \geq \tau$. Luego,*

$$\Gamma \vdash_e M : \sigma \Rightarrow \Gamma \vdash_e M : \tau$$

Demostración. Por inducción en la prueba de $\sigma \geq \tau$.

Caso base: $\sigma = \tau$, por lo que la implicación es trivialmente verdadera.

Paso inductivo: $\sigma \geq \sigma' > \tau$. Por hipótesis inductiva

$$\Gamma \vdash_e M : \sigma \Rightarrow \Gamma \vdash_e M : \sigma'$$

Pero por definición de $>$

$$\sigma' \equiv \forall \alpha.\sigma'_1 \quad \text{y} \quad \tau \equiv \sigma'_1[\alpha := \rho]$$

Luego, por $(\forall E)$ puede inferirse $\Gamma \vdash_e M : \tau$, por lo que vale

$$\Gamma \vdash_e M : \sigma \Rightarrow \Gamma \vdash_e M : \tau$$

□

Con estos resultado auxiliares ya demostrados, estamos en condiciones de probar la propiedad de clausura para el sistema.

A.2.2. Propiedad de clausura

Proposición A.2.1 (Subject Reduction)**.** *Sean $M, M' \in \Lambda$ tal que $M \to_\beta M'$. Entonces*

$$\Gamma \vdash_e M : \sigma \Rightarrow \Gamma \vdash_e M' : \sigma$$

Demostración. Por inducción en el término M, analizando el lugar donde se produce la reducción. Presentamos a continuación el caso en que $M \equiv (\lambda x.P)Q$ y $M' \equiv P[Q/x]$. Los casos de reducción interna se deducen directamente de la hipótesis inductiva, aplicando lema de generación, de manera análoga a la demostración presentada para la Proposición A.1.1.

Se tiene $\Gamma \vdash_e (\lambda x.P)Q : \sigma$. Por Lema A.2.3 (2)

$$\exists \sigma' \exists \tau' \geq \sigma \;\; [\Gamma \vdash_e \lambda x.P : \sigma' \to \tau' \quad \text{y} \quad \Gamma \vdash_e Q : \sigma']$$

Ahora por Lema A.2.3 (3) vale

$$\exists\, \sigma'', \tau'' \;\; [\Gamma, x : \sigma'' \vdash_e P : \tau'' \quad \text{y} \quad \sigma' \to \tau' \equiv \sigma'' \to \tau'']$$

Luego, se tiene

$$\Gamma, x : \sigma' \vdash_e P : \tau' \quad \text{y} \quad \Gamma \vdash_e Q : \sigma'$$

Entonces, por Lema A.2.2 (2)

$$\Gamma \vdash_e P[Q/x] : \tau'$$

Finalmente, como $\tau' \geq \sigma$, por Lema A.2.4 vale

$$\Gamma \vdash_e P[Q/x] : \sigma$$

□

Nuevamente la propiedad es extensible a muchos pasos de reducción:

Corolario A.2.1. *Sean $M, M' \in CL$ tal que $M \twoheadrightarrow_\beta M'$. Entonces*

$$\Gamma \vdash_e M : \sigma \Rightarrow \Gamma \vdash_e M' : \sigma$$

Demostración. Por inducción en la longitud de la derivación $M \twoheadrightarrow_\beta M'$.

□

Con este resultado puede verse que $\boldsymbol{\lambda 2}^e$, al igual que $\boldsymbol{\lambda 2}^i$, cumple con la propiedad principal que se desea tenga todo sistema de tipos.

Más aún, es inmediato verificar, para los dos subsistemas estudiados, que todo juicio derivable en ellos es también derivable en $\boldsymbol{\lambda 2}$. Lo que implica de manera directa la propiedad de SN en ambos formalismos, pues ésta vale en el sistema completo [4]. De este modo pueden estudiarse distintas propiedades sobre los subsistemas, algunas de las cuales se "heredan" de $\boldsymbol{\lambda 2}$.

Bibliografía

[1] Arbiser, A.: *Consistencia a través de variables positivas.* En *LVIII Reunión Anual de la Unión Matemática Argentina*, Mendoza, 2008. UMA.

[2] Baader, F. y T. Nipkow: *Term Rewriting and All That.* Cambridge University Press, New York, NY, USA, 1998, ISBN 0-521-45520-0.

[3] Barendregt, H. P.: *The Lambda Calculus: Its Syntax and Semantics, Revised Edition.* North Holland, 1984.

[4] Barendregt, H. P.: *Lambda Calculi With Types.* En *Handbook of Logic in Computer Science (Vol. 2): Background: Computational Structures*, págs. 117–309, New York, NY, USA, 1992. Oxford University Press, Inc., ISBN 0-19-853761-1.

[5] Bezem, M., J. W. Klop y R. de Vrijer: *Term Rewriting Seminar – TeReSe.* Cambridge University Press, New York, NY, USA, 2003.

[6] Bruijn, N. G. de: *The Mathematical Language AUTOMATH, Its Usage, and Some of Its Extensions.* En Laudet, M. (ed.): *Proceedings of the Symposium on Automatic Demonstration*, págs. 29–61, Versailles, France, 1968. Springer-Verlag LNM 125.

[7] Bunder, M. W.: *The inhabitation problem for intersection types.* En *CATS '08: Proceedings of the Fourteenth Symposium on Computing: the Australasian Theory*, págs. 7–14, Darlinghurst, Australia, 2008. Australian Computer Society, Inc., ISBN 978-1-920682-58-3.

[8] Church, A.: *A Formulation of the Simple Theory of Types.* Journal of Symbolic Logic, 5(2):56–68, 1940.

[9] Church, A.: *The Calculi of Lambda Conversion (Annals of Mathematics Studies Vol. 6).* Princeton University Press, Princeton, NJ, USA, 1941, ISBN 0691083940.

[10] Coppo, M. y M. Dezani-Ciancaglini: *A New Type-Assignment for Lambda Term.* Archiv für Mathematische Logik, 19:139–156, 1978.

[11] Coquand, T. y G. P. Huet: *The Calculus of Constructions.* Information and Computation, 76(2-3):95–120, 1988, ISSN 0890-5401.

[12] Curry, H. B.: *Functionality in Combinatory Logic.* Proceedings of the National Academy of Science USA, 20:584–590, 1934.

[13] Curry, H. B. y R. Feys: *Combinatory Logic, Vol. 1.* North-Holland Publishing Co., Amsterdam, 1958.

[14] Curry, H. B., J. R. Hindley y J. P. Seldin: *Combinatory Logic, Vol. 2.* North-Holland Publishing Co., Amsterdam, 1972.

[15] Dezani-Ciancaglini, M. y J. R. Hindley: *Intersection Types for Combinatory Logic.* Theoretical Computer Science, 100(2):303–324, 1992, ISSN 0304-3975.

[16] Enderton, H. B.: *A Mathematical Introduction to Logic.* Academic Press, New York, 1972.

[17] Gabbay, D. M. y R. J. G. B. de Queiroz: *Extending the Curry–Howard Interpretation to Linear, Relevant and Other Resource Logics.* The Journal of Symbolic Logic, 57(4):1319–1365, 1992.

[18] Girard, J. Y.: *Interprétation Fonctionnelle et Élimination des Coupures de l'Arithmétique d'Ordre Supérieur.* Thèse de doctorat d'état, Université Paris VII, Paris, France, 1972.

[19] Howard, W. A.: *The Formulae-as-Types Notion of Construction.* En Seldin, J. P. y J. R. Hindley (eds.): *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, págs. 479–490. Academic Press, Inc., New York, N.Y., 1980.

[20] Lavalette, G. R. Renardel de: *Strictness Analysis Via Abstract Interpretation for Recursively Defined Types.* Information and Computation, 99(2):154–177, 1992.

[21] Longo, G. y E. Moggi: *Constructive Natural Deduction and Its Modest Interpretation.* Mathematical Structures in Computer Science, 1, 1991.

[22] Mendelson, E.: *Introduction to Mathematical Logic, 3rd Edition.* Wadsworth and Brooks/Cole Advanced Books & Software, Monterey, CA, USA, 1987, ISBN 0-534-06624-0.

[23] Miquel, A.: *System F.* En *Types Summer School 2005*, Göteborg, 2008.

[24] Mitchell, J. C.: *Polymorphic Type Inference and Containment.* Information and Computation, 76(2-3):211–249, 1988, ISSN 0890-5401.

[25] Pierce, B. C.: *Intersection Types and Bounded Polymorphism.* En Bezem, M. y J. F. Groote (eds.): *Proc. of 1st Int. Conf. on Typed Lambda Calculi and Applications, TLCA'93, Utrecht, The Netherlands, 16–18 March 1993*, vol. 664, págs. 346–360. Springer-Verlag, Berlin, 1993.

[26] Reeves, S. y M. Clarke: *Logic for Computer Science, 2nd Edition.* Addison-Wesley, Reading, MA, England, 2003.

[27] Reynolds, J. C.: *Towards a Theory of Type Structure.* En *Symposium on Programming*, págs. 408–423, 1974.

[28] Schönfinkel, M.: *Über die Bausteinen der Matematische Logik.* Mathematische Annalen, 92:305–316, 1924.

[29] Sørensen, M. H. y P. Urzyczyn: *Lectures on The Curry–Howard Isomorphism, Vol. 149 (Studies in Logic and the Foundations of Mathematics).* Elsevier Science Inc., New York, NY, USA, 2006, ISBN 0444520775.

[30] Tiuryn, J. y P. Urzyczyn: *The Subtyping Problem for Second-Order Types is Undecidable.* Information and Computation, 176(1), 2002.

[31] Wells, J. B.: *Typability and Type Checking in System F Are Equivalent and Undecidable.* Annals of Pure and Applied Logic, 98:111–156, 1998.