Lógica de pruebas para certificación de computación móvil
Tesina de grado

Federico Feller
Facultad de Informática
UNLP

Índice general

Capítulo 1

Introducción

1.1. Introducción

Hoy en día el uso intensivo de redes de comunicación permite contar con sistemas de información en donde los cálculos se distribuyen entre varios nodos de una red. Una manera de implementar sistemas distribuidos es mediante el uso de código móvil.

La idea básica de código móvil es sencilla: un agente A en un sector de la red escribe un componente de software en algún lenguaje, lo compila y lo transmite para su ejecución a otro agente B ubicado en un punto diferente de la red. Al agente que envía al programa se lo conoce como productor de código, mientras que al agente que lo recibe y ejecuta se lo denomina consumidor de código. Para que un programa pueda ser enviado a través de la red se requiere que el código no dependa de ningún recurso local. El esquema de uso de código móvil es flexible y poderoso, pero si no se tienen ciertos recaudos pueden ocurrir problemas relacionados con la seguridad del sistema. Por ejemplo, un programa que se envía por la red puede ser capturado y alterado por un tercero con intenciones de afectar a quien lo ejecuta. Incluso el productor mismo puede generar, por error, código que ponga en riesgo la seguridad del consumidor. Por lo tanto, para que este esquema funcione, es de vital importancia que el consumidor pueda confiar en que el código enviado por el productor cumple con los requisitos necesarios para garantizar la seguridad. Se pueden pensar varias alternativas para manejar este tipo de situaciones, como ser el uso de criptografía para identificar al consumidor del código. En particular, la alternativa que se considera en este trabajo es PCC o Proof-Carrying-Code.

PCC es un mecanismo introducido en [Nec00] para garantizar la seguridad en un entorno donde se utiliza código móvil. En este esquema, el consumidor define formalmente una política de seguridad mediante un conjunto de reglas que los programas deben cumplir para ser considerados seguros. Conceptualmente, las acciones que realizan tanto el productor como el consumidor de código están divididas en tres etapas. En la primer etapa, denominada certificación, el productor crea una prueba formal o certificado que asegura que el código que va a ser enviado respeta la política de seguridad previamente definida por el consumidor. Este certificado se adjunta al código y se envía al consumidor. En la segunda etapa, denominada validación, el

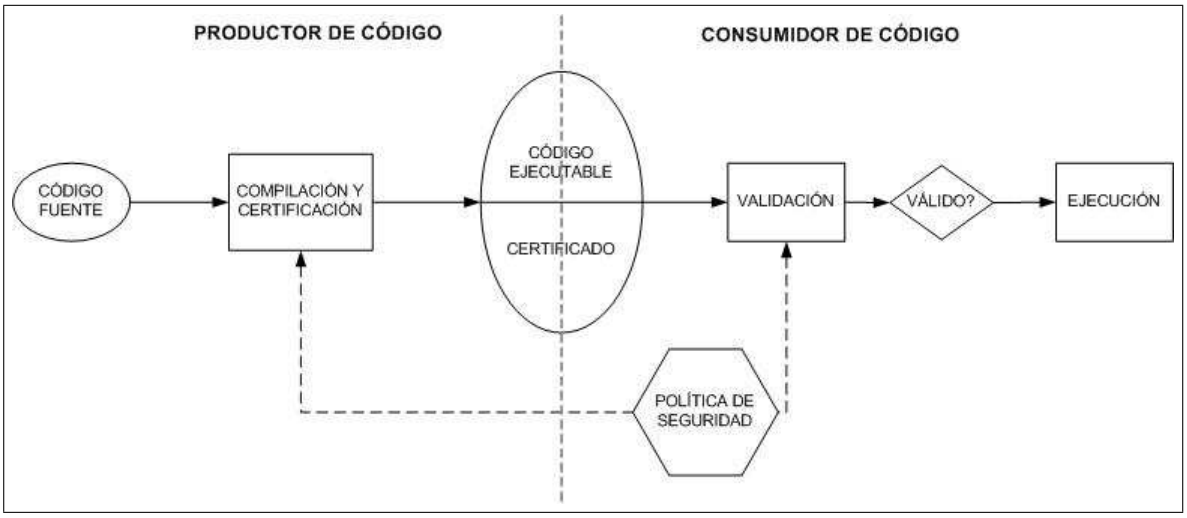

Figura 1.1: Resumen de PCC

consumidor valida el certificado que acompaña al código. La validación se realiza en forma rápida y sólo una vez para un programa dado. Finalmente, en la última etapa el consumidor ejecuta efectivamente el código. Lo puede realizar tantas veces como sea necesario sin la necesidad de volver a realizar la validación. Además, esta etapa procede sin ningún tipo de verificaciones adicionales en tiempo de ejecución ya que la etapa anterior asegura que el código cumple con las reglas de seguridad.

Una característica de PCC es que el código y su certificado se generan de manera separada. Este trabajo propone estudiar un modelo de lenguaje de programación para el productor en PCC en el que la generación de código y certificado se realiza en un marco unificado.

El modelo estudiado supone un conjunto finito de nodos denominados mundos posibles sobre los cuales se ejecutan los programas (ver figura 1.2). Todos los mundos se suponen que están conectados entre sí. En un momento determinado, la ejecución de un programa sucede en un mundo particular (mundo actual), pero la misma puede trasladarse hacia otro mundo, siempre y cuando se trate de código móvil con un certificado válido.

La definición del modelo consiste en tres partes: un lenguaje de programación recortado al estilo lambda cálculo denominado $\lambda_{\Box}^{\mathsf{Cert}}$, un sistema de tipos y una semántica operacional para definir el comportamiento de los programas. Un ejemplo de programa en $\lambda_{\Box}^{\mathsf{Cert}}$ es el siguiente:

$$(\lambda a.\mathit{unpack}\ a\ \mathit{to}\ \langle v^{\bullet}, v^{\circ}\rangle\ \mathit{in}\ v^{\bullet})(\mathit{fetch}[w_1]\ \mathit{box}_s\, M)$$

Este término es una aplicación de los dos términos encerrados entre paréntesis y que se puede ejecutar, por ejemplo, en el mundo w_0. El término de la izquierda es una función a la cual se le aplica el término de la derecha. En este último se encuentra la construcción $\mathit{box}_s\, M$ que se denomina unidad móvil y que representa la conjunción de código móvil M junto a su certificado s que garantiza que M no depende de recursos locales. Como se trata de código móvil, es posible ejecutarlo en

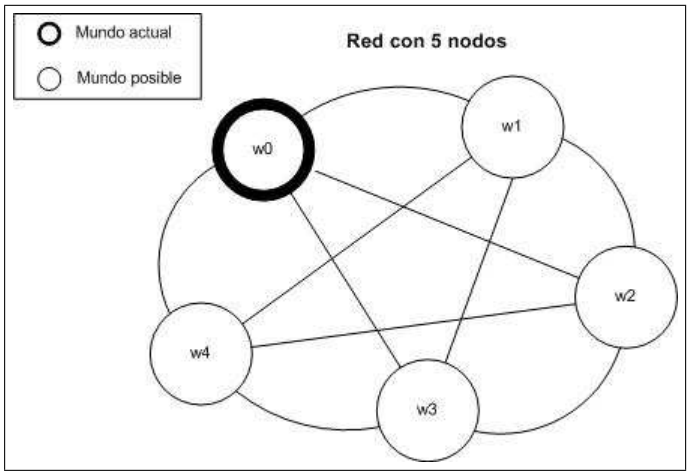

Figura 1.2: Ejemplo de red de nodos

cualquier nodo, en particular, con la instrucción $fetch[w_1]$ se indica que el mismo debe ser ejecutado en w_1. En el cuerpo de la función, se encuentra una instrucción *unpack*, que permite inspeccionar las partes de una unidad móvil (en este caso, el valor ligado a la variable a), pudiendo utilizar v° para referirse al certificado y $v^\bullet$ para referirse al código. Los detalles sobre estas construcciones son explicados en el capítulo 3. (Notar que se utilizó un esquema de término M y no un término en concreto, para simplificar el ejemplo).

Para poder definir $\lambda_{\Box}^{\mathsf{Cert}}$ y su sistema de tipos, se ha utilizado una técnica inspirada en el isomorfismo de Curry-De Bruijn-Howard, en donde las proposiciones y pruebas de una lógica se interpretan como los tipos y términos de un lenguaje. La lógica elegida en este caso es LP o lógica de pruebas y, en particular, su versión intuicionista ILPnd. La lógica de pruebas es una lógica modal que tiene la característica que el operador $[]A$ se reemplaza por $[s]A$ con la siguiente interpretación: "s es una prueba de A". Aquí s es una codificación de la prueba de A, por lo que se dice que LP internaliza sus propias pruebas. Esta propiedad de LP es la que la propone como candidata natural para derivar un modelo que incluya la generación de certificados. Cuando se traslada la lógica al mundo computacional a través del isomorfismo, la interpretación que se le da a $[s]A$ es la del tipo de los programas que representan código móvil y tienen certificado s. Los certificados en el nuevo modelo son generados a partir de las reglas definidas por el sistema de tipos.

Una vez definido el cálculo es necesario darle una semántica que defina el comportamiento de los programas. Para ello se ha definido una máquina abstracta con estados y transiciones capaz de simular la ejecución de código móvil.

En base a todas las definiciones realizadas, se estudiaron formalmente las propiedades que cumple el modelo. El conjunto de propiedades que han sido demostradas puede ser dividido en dos grupos: seguridad en tipos y normalización fuerte. La seguridad de tipos o *type safety* consiste en garantizar que un programa bien tipado no puede fallar al momento de ser ejecutado. Esto incluye el motivo de falla resultante de la generación de código móvil junto a un certificado que no se corresponde con el código. Para demostrar seguridad de tipos se han demostrado dos propieda-

des. La primera de ellas es progreso o *progress*, que consiste en demostrar que la máquina abstracta nunca se quede trabada en un estado bien tipado. La segunda de ellas es preservación de tipos ó *subject reduction*, que consiste en demostrar que si se realiza una reducción desde un estado bien tipado a otro estado, entonces este último también es bien tipado. La normalización fuerte consiste en demostrar que no existan secuencias de reducción de la máquina de estados bien tipados que no terminen nunca.

Además de la definición teórica del cálculo, se ha realizado una implementación de un prototipo del lenguaje en el lenguaje funcional Haskell, que ha servido para poner a prueba los conceptos introducidos. Para ello, antes fue necesario introducir una serie de extensiones a la definición original del cálculo que le han permitido darle más expresividad, como ser valores booleanos y números naturales.

Los resultados del trabajo realizado han sido publicados en el simposio internacional LFCS 2009. Para más detalles, referirse a [BF09]

1.2. Estructura del trabajo

El trabajo se organiza de la siguiente manera. En el capítulo 2 se hace una recapitulación las lógicas de pruebas, incluyendo LP y su versión intuisionista ILPnd. En el capítulo 3 se introduce la definición básica de $\lambda_{\square}^{Cert}$ incluyendo la sintaxis, el sistema de tipos y la semántica operacional. En el capítulo 4 se detallan las demostraciones de las propiedades más importantes del cálculo: seguridad de tipos y normalización fuerte. En el capítulo 5 se presentan algunas extensiones a la definición original del $\lambda_{\square}^{Cert}$ que permiten darle más expresividad al lenguaje, incluyendo valores booleanos y números naturales. En el capítulo 6 se analizan algunos aspectos de la implementación del prototipo del cálculo en Haskell. Finalmente, en el capítulo 7, se presentan algunas conclusiones, trabajo relacionado y sugerencias para el trabajo a futuro. Adicionalmente, se incluye un apéndice con el detalle de las demostraciones de todas las propiedades auxiliares.

Capítulo 2

Lógica de pruebas

En este capítulo se introducen una serie de lógicas de pruebas que son lógicas modales capaces de internalizar sus derivaciones y que, por lo tanto, proveen un entorno natural para la generación de código certificado. En particular, se estudia LP, su versión intuicionista ILP y ILPnd, una representación en deducción natural de esta última.

2.1. Lógicas modales y LP

Como ya se ha mencionado, el objetivo principal del trabajo es definir un modelo que capture la noción de computaciones móviles con certificados. El modelo debe incluir un lenguaje de programación, un sistema de tipos y una semántica operacional. No se trata de un lenguaje de programación completo, sino un cálculo recortado definido formalmente el cual sirva para dar garantías rigurosas de las propiedades estudiadas. Para definir los primeros dos aspectos del modelo se utilizará una técnica basada en el isomorfismo de Curry-DeBrujiun-Howard [GTL89], que consiste en interpretar las proposiciones y las pruebas de una lógica como los tipos y programas de un lenguaje. Es decir que se tomará una lógica y a partir de ella se obtendrá un cálculo junto a su sistema de tipos. Por lo tanto, el primer paso para definir el modelo consiste en elegir una lógica adecuada.

En trabajos anteriores ([Mur08], [Moo04]) ya se han utilizado lógicas modales para representar computaciones móviles. Las lógicas modales permiten razonar sobre el valor de verdad de una proposición desde diferentes perspectivas o mundos posibles. Estos mundos se relacionan entre sí mediante una relación de accesibilidad. Las lógicas modales que son de interés para este trabajo se caracterizan por incluir el operador modal $\Box$. Dada una proposición cualquiera A, es posible construir la proposición $\Box A$ que se interpreta como "A es verdadero en todo mundo accesible desde el mundo actual". Una interpretación computacional posible de estas lógicas consiste en ver a los mundos posibles de la lógica como los nodos de una red y a una proposición de la forma $\Box A$ como el tipo de un programa que representa código móvil y que calcula un valor de tipo A. Como en general se desea que todos los nodos de la red estén comunicados entre sí, se considera una relación de accesibilidad que sea reflexiva, simétrica y transitiva. La prueba de que $\Box A$ es verdadero en la lógica

se traduce a un programa que representa el código móvil.

Por ejemplo, la proposición $\Box P \supset (\Box(P \supset Q) \supset \Box Q)$ se puede interpretar como el tipo de un programa que, dado código móvil que calcula un valor de tipo P y código móvil que representa una función de P en Q, retorna código móvil que calcula un valor de tipo Q.

Hasta aquí se sabe que una lógica modal puede ser útil para modelar computaciones móviles. Para definir el cálculo buscado es necesario también considerar la construcción de los certificados. Para ello, se ha optado por utilizar una lógica modal denominada lógica de pruebas o LP [Art94]. En esta lógica el operador modal $\Box A$ es reemplazado por uno de la forma $[s]A$ que se lee como 's es una prueba de A'. A s se lo denomina polinomio de prueba y consiste en un término que codifica la prueba de que A es verdadero. La idea propuesta es hacer una interpretación computacional de $[s]A$ como el tipo de un programa que representa código móvil y que calcula un valor de tipo A, siendo s la prueba o certificado de esta afirmación. A continuación se presenta la definición de LP.

Definición 2.1 (Lenguaje LP)**.** *Consta del lenguaje usual de la lógica proposicional clásica junto a: variables de prueba $x_0, \ldots, x_n, \ldots$,; constantes de prueba $a_0, \ldots, a_n, \ldots$,; símbolos de función ! (check), $\cdot$ (aplicación) y + (suma).*

Los términos o polinomios de pruebas (denominados $p, s, t, \ldots$) se construyen a partir de variables y constantes de pruebas utilizando los operadores $\cdot, +, !$. Las fórmulas en LP se construyen a partir de átomos proposicionales utilizando los conectivos booleanos usuales (al igual que en la lógica clásica) y aplicando una nueva regla de formación que dice: si t es un polinomio de pruebas y A es una fórmula, entonces $[t]A$ es una fórmula.

Definición 2.2 (Axiomas y reglas de inferencia de LP)**.**

$A0$.	*Reglas de inferencia de la lógica proposicional clásica*	
$A1$.	$[t]A \supset A$	*"verificación"*
$A2$.	$[t](A \supset B) \supset ([s]A \supset [t \cdot s]B)$	*"aplicación"*
$A3$.	$[t]A \supset [!t][t]A$	*"chequeo de prueba"*
$A4$.	$[s]A \supset [s+t]A, [t]A \supset [s+t]A$	*"selección"*
$R1$.	$\vdash A \supset B$ *y* $\vdash A$ *entonces* $\vdash B$	*"modus ponens"*
$R2$.	$\vdash [c]A$, *si* A *es un axioma* $A0 - A4$ *y* c *es una cte de prueba*	*"necessitation"*

Una lectura de las reglas de inferencia es la siguiente. Para el caso de la verificación se lee "si t es una prueba de A entonces A vale". La aplicación se lee "si t es una prueba de $A \supset B$ y s es una prueba de B, entonces $t \cdot s$ es una prueba de B". Es decir, "$\cdot$" representa la composición de pruebas. Para el caso de selección se lee "si s es una prueba de A entonces $s + t$ también es una prueba de A." El operador '+' representa la concatenación de pruebas. Para el caso de chequeo de prueba se lee "si t es una prueba de A, entonces $!t$ es una prueba de 't es una prueba de A ". Es decir que el operador '!' puede verse como una computación que verifica $[t]A$.

En LP es admisible escribir la regla de *necessitation* como:

$$\text{Si } \vdash A \text{ entonces } \vdash [p]A, \text{ para algún polinomio de prueba } p$$

Aquí, en realidad, p no es otra cosa que la codificación de alguna derivación de F en LP. Por lo tanto, se dice que LP es un sistema capaz de internalizar sus propias pruebas.

Una propiedad muy importante de LP es que es completa con respecto a demostrabilidad en PA (Aritmética de Peano). Es decir, las fórmulas que son derivables en LP se corresponden exactamente con las proposiciones que son demostrables en PA (consultar [Art94] para una formulación precisa). Además, LP tiene una correspondencia con otra lógica modal denominada S4. Esta lógica cuenta con el operador modal usual $\Box A$. Una derivación en LP puede ser fácilmente transformada en una derivación en S4 sustituyendo todas las ocurrencias de $[p]A$ por $\Box A$. Más interesante aún resulta la propiedad inversa, que establece que toda derivación en S4 puede transformarse en una derivación en LP. En este caso es necesario asignar polinomios de pruebas a cada ocurrencia de $\Box A$. Es decir que LP captura todos los teoremas de S4 lo que permite dar una semántica de probabilidad para S4, problema que estuvo abierto durante un tiempo largo hasta la aparición de LP.

La capacidad de LP de internalizar sus derivaciones lo convierte en un candidato natural para la generación de certificados de código. Para definir el cálculo de certificados móviles no se utilizará LP tal cual fue definido, sino una versión intuicionista del mismo que se denomina ILP. Esta visión de utilizar una versión intuicionista de LP es análoga al uso de lógica intuicionista (IPC o Int) para obtener el lambda cálculo simplemente tipado. Más precisamente, se considerará LP tomando como base el fragmento proposicional mínimo en lugar de la lógica proposicional clásica. Con el propósito de obtener una representación más directa del cálculo de certificados se utilizará una representación en deducción natural de ILP que se denomina ILPnd que se explica a continuación.

2.2. ILPnd

ILPnd es una representación en deducción natural de ILP que fue introducida en [AB07] considerando dos conjuntos de hipótesis, de verdad y de validez, y analizando el significado del siguiente juicio hipotético con evidencia explícita:

$$\Delta; \Gamma \rhd A \mid s$$

En este juicio, Δ es una secuencia de hipótesis de validez, Γ es una secuencia de hipótesis de verdad, A es una proposición y s es un término de prueba. Una hipótesis de validez en Δ tiene la forma $v_i : A_i$ $(i \in 1 \ldots n)$, donde v_i pertenece a un conjunto infinito de variables de validez y A_i es una proposición que se asume que vale en todos los mundos accesibles. De manera similar, una hipótesis de verdad en Γ tiene la forma $a_j : B_j$ $(j \in 1 \ldots m)$, donde a_j pertenece a un conjunto infinito de variables de de verdad y B_j es una proposición que se asume que vale en el mundo actual. Se requiere que tanto las v_i en Δ como las a_j en Γ sean diferentes entre sí y que sean frescas, es decir, que no aparezcan ni en A_i ni en B_j. El juicio se lee como "A es verdadero con evidencia s bajo las hipótesis de validez Δ y las hipótesis

de verdad Γ". Notar que el término de prueba s es parte constituyente del juicio y que sin él este último carece de sentido. La siguiente es una definición formal de las proposiciones, los términos de prueba y de los contextos de validez y verdad que forman el juicio.

Definición 2.3.

$$\begin{array}{lrcl} \textit{Términos de prueba} & s,t & ::= & x \mid s \cdot t \mid \lambda a : A.s \mid !s \mid \textsc{letc}\, s\, \textsc{be}\, v : A\, \textsc{in}\, t \\ \textit{Proposiciones} & A,B & ::= & P \mid A \supset B \mid [s]A \\ \textit{Contextos de verdad} & \Gamma & ::= & \cdot \mid \Gamma, a : A \\ \textit{Contextos de validez} & \Delta & ::= & \cdot \mid \Delta, v : A \end{array}$$

Todas las ocurrencias libres de a están ligadas en $\lambda a : A.s$. De manera similar, todas las ocurrencias libres de v están ligadas en $\textsc{letc}\, s\, \textsc{be}\, v : A\, \textsc{in}\, t$. Una proposición puede ser una variable P, una implicación $A \supset B$ ó una proposición de validez $[s]A$. A un contexto vacío se lo denota $\cdot$. Se escribe $s\{x/t\}$ para denotar el resultado de sustituir todas las ocurrencias libres de x en s por t; de manera similar para $A\{x/t\}$.

El significado del juicio $\Delta; \Gamma \rhd A \mid s$ está dado por los axiomas y reglas de inferencia que se definen a continuación.

Definición 2.4.

Fragmento de la lógica proposicional minimal

$$\frac{}{\Delta; \Gamma, a : A, \Gamma' \rhd A \mid a}\ \mathsf{oVar}$$

$$\frac{\Delta; \Gamma, a : A \rhd B \mid s}{\Delta; \Gamma \rhd A \supset B \mid \lambda a : A.s}\ \supset \mathsf{I} \qquad \frac{\Delta; \Gamma \rhd A \supset B \mid s \quad \Delta; \Gamma \rhd A \mid t}{\Delta; \Gamma \rhd B \mid s \cdot t}\ \supset \mathsf{E}$$

Fragmento de demostrabilidad

$$\frac{}{\Delta, v : A, \Delta'; \Gamma \rhd A \mid v}\ \mathsf{mVar}$$

$$\frac{\Delta; \cdot \rhd A \mid s}{\Delta; \Gamma \rhd [s]A \mid !s}\ \Box\mathsf{I} \qquad \frac{\Delta; \Gamma \rhd [r]A \mid s \quad \Delta, v : A; \Gamma \rhd C \mid t}{\Delta; \Gamma \rhd C\{v/r\} \mid \textsc{letc}\, s\, \textsc{be}\, v : A\, \textsc{in}\, t}\ \Box\mathsf{E}$$

$$\frac{\Delta; \Gamma \rhd A \mid s \quad \Delta; \Gamma \vdash s \equiv t : A}{\Delta; \Gamma \rhd A \mid t}\ \mathsf{EqEvid}$$

Una explicación informal de estas reglas es la siguiente. El axioma oVar establece que el juicio $\Delta; \Gamma, a : A, \Gamma' \rhd A \mid a$ es evidente en sí mismo. De hecho, si se asume que a es evidencia de que la proposición A es verdadera, entonces se puede concluir inmediatamente que A es verdadera con evidencia a. De manera similar, la regla mVar establece que si se asume que v es evidencia de que la proposición A es válida,

entonces se puede utilizar esta evidencia para inmediatamente concluir de que A es verdadera. Los esquemas de inferencia $\supset$ I y $\supset$ E son los estándar, con el agregado de los términos de prueba como evidencia. El esquema de introducción para el operador modal $[s]$ (regla $\Box$I) internaliza evidencia de metanivel dentro la lógica objeto. Esta regla establece que, si s es evidencia incondicional de que A es verdadera, entonces A es de hecho válida con evidencia s, es decir, $[s]A$ es verdadera. La evidencia de que $[s]A$ es verdadera se construye a partir de la evidencia (verificada) que A es incondicionalmente verdadera agregándole como prefijo el constructor '!'. Finalmente, la regla $\Box$E permite descargar hipótesis de validez. Para poder descargar una hipótesis de validez de la forma $v : A$ es necesita una prueba de que A es válida. En este sistema, esto significa que se requiere una prueba de que $[r]A$ es verdadera con evidencia s, para ciertos términos de prueba r y s. Notar que r es evidencia de que A es incondicionalmente verdadera (es decir, válida), mientras que s es evidencia de que $[r]A$ es verdadera. Esta última evidencia es sustituida en lugar de todas las ocurrencias libres de v en la proposición C. Esta construcción es registrada con la evidencia $\text{LETC}\, s\, \text{BE}\, v : A\, \text{IN}\, t$ en la conclusión.

La inclusión de la regla EqEvid está relacionada con el proceso de normalización de pruebas en ILPnd y la capacidad de este sistema de internalizar sus propias pruebas. Por ejemplo, supongamos que se tiene la siguiente derivación:

$$\dfrac{\dfrac{\dfrac{v : A; a : A \rhd A \mid a}{v : A; \cdot \rhd A \supset A \mid \lambda a : A.a} \supset I \qquad v : A; \cdot \rhd A \mid v}{v : A; \cdot \rhd A \mid (\lambda a : A.a) \cdot v} \supset E}{v : A; \cdot \rhd [(\lambda a : a.A) \cdot v]A \mid !(\lambda a : A.a) \cdot v} \Box\mathsf{I}$$

Una definición naïve de paso de normalización podría ser la siguiente:

$$\dfrac{\dfrac{\Delta; \Gamma, a : A \rhd B \mid s}{\Delta; \Gamma \rhd A \supset B \mid \lambda a : A.s} \supset I \qquad \Delta; \Gamma \rhd A \mid t}{\Delta; \Gamma \rhd B \mid (\lambda a : A.s) \cdot t} \supset E \quad \rightsquigarrow \quad \Delta; \Gamma \rhd B \mid s^a_t$$

Sin embargo, si se aplica el paso de normalización en el ejemplo a la subderivación terminada en el juicio $v : A; \cdot \rhd A \mid (\lambda a : A.a) \cdot v$, la derivación resultante no sería válida. El problema aquí es que la normalización de pruebas introduce en un metanivel de la lógica una relación de identidad entre derivaciones. Estas derivaciones a su vez internalizan sus propias pruebas. Como consecuencia, para que el proceso de normalización funcione adecuadamente el mismo debe estar reflejado también en la lógica objeto. Para ello se introduce un juicio de igualdad entre evidencias, $\Delta; \Gamma \vdash s \equiv t : A$. En el ejemplo, se puede usar que $\Delta; \Gamma \vdash s^a_t \equiv (\lambda a : A.s) \cdot t : B$ para escribir una normalización que funcione. Para más detalles sobre las reglas para este tipo de juicios se puede consultar [AB07].

Es importante remarcar que no se requiere de la regla EqEvid para ver que la definición de ILPnd es correcta con respecto a ILP. La misma sólo es necesaria para que el proceso de normalización sea cerrado respecto al conjunto de derivaciones.

Dado que no se utilizará el proceso de normalización para darle semántica al cálculo a introducir, para el propósito del trabajo no es necesario entrar en más detalle sobre esta regla. A continuación se presentan ejemplos de derivaciones en ILPnd.

Ejemplo 2.1. El siguiente es un ejemplo de la derivación de $[s]A \supset [!s][s]A$

$$\dfrac{\dfrac{}{\cdot; a : [s]A \rhd [s]A \mid a}\,\mathsf{oVar} \qquad \dfrac{\dfrac{\dfrac{}{w : A; \cdot \rhd A \mid w}\,\mathsf{mVar}}{w : A; \cdot \rhd [w]A \mid !w}\,\Box\mathsf{I}}{w : A; a : [s]A \rhd [!w][w]A \mid !!w}\,\Box\mathsf{I}}{\dfrac{\cdot; a : [s]A \rhd [!s][s]A \mid \text{LETC}\, a\, \text{BE}\, w : A\, \text{IN}\, !!w}{\cdot; \cdot \rhd [s]A \supset [!s][s]A \mid \lambda a : [s]A.\text{LETC}\, a\, \text{BE}\, w : A\, \text{IN}\, !!w}\,\supset\mathsf{I}}\,\Box\mathsf{E}$$

Ejemplo 2.2. El siguiente es un ejemplo de derivación de la proposición

$$[s](A \supset B) \supset [t]A \supset [s \cdot t]B$$

donde $\Gamma = a : [s](A \supset B), b : [t]A$ y $\Delta = u : A \supset B, v : A$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\cdot; a : [s](A \supset B) \rhd [t]A \supset [s \cdot t]B \mid \lambda b : [s]A.\text{LETC}\, a\, \text{BE}\, u : A \supset B\, \text{IN}\, \text{LETC}\, b\, \text{BE}\, v : A\, \text{IN}\, !u \cdot v}\,\supset\mathsf{I}}{\cdot; \cdot \rhd [s](A \supset B) \supset [t]A \supset [s \cdot t]B \mid \lambda a : [s](A \supset B).\lambda b : [t]A.\text{LETC}\, a\, \text{BE}\, u : A \supset B\, \text{IN}\, \text{LETC}\, b\, \text{BE}\, v : A\, \text{IN}\, !(u \cdot v)}\,\supset\mathsf{I}}{\dfrac{\cdot; \Gamma \rhd [s](A \supset B) \mid a \qquad \dfrac{u : (A \supset B); \Gamma \rhd [t]A \mid b \qquad \dfrac{\dfrac{\Delta; \cdot \rhd A \supset B \mid u \quad \Delta; \cdot \rhd A \mid v}{\Delta; \cdot \rhd B \mid u \cdot v}}{\Delta; \Gamma \rhd [u \cdot v]B \mid !(u \cdot v)}\,\Box\mathsf{I}}{u : (A \supset B); \Gamma \rhd [u \cdot t]B \mid \text{LETC}\, b\, \text{BE}\, v : A\, \text{IN}\, !(u \cdot v)}\,\Box\mathsf{E}}{\cdot; \Gamma \rhd [s \cdot t]B \mid \text{LETC}\, a\, \text{BE}\, u : A \supset B\, \text{IN}\, \text{LETC}\, b\, \text{BE}\, v : A\, \text{IN}\, !u \cdot v}\,\Box\mathsf{E}}$$

En este capítulo se presentó un panorama general de la lógica de pruebas LP y su versión intuicionista ILPnd. Es en base a esta última lógica que se define el cálculo $\lambda^{\mathsf{Cert}}_{\Box}$, tal como se verá en el próximo capítulo.

Capítulo 3

Cálculo para certificados de código móvil

En este capítulo se introduce $\lambda_{\Box}^{\mathsf{Cert}}$, un cálculo para modelar computaciones móviles con certificados. El objetivo no es presentar un lenguaje de programación completo, sino un cálculo al estilo λ-cálculo que permita demostrar propiedades a partir de razonamientos formales. La definición de $\lambda_{\Box}^{\mathsf{Cert}}$ surge de una interpretación computacional de ILPnd introducido en el capítulo anterior. La misma consiste en: una sintaxis para construir los términos y los certificados, un sistema de tipos definido a partir de la asignación de términos al sistema ILPnd y una semántica basada en la definición de una máquina abstracta. En las diferentes secciones de este capítulo se definen cada una de las partes mencionadas.

3.1. Términos y certificados

Definición 3.1.

mundos	Σ	::=	$\{w_1, \ldots w_n\}$
contexto móvil	Δ	::=	$\cdot \mid \Delta, v : A@w$
contexto local	Γ	::=	$\cdot \mid \Gamma, a : A@w$
tipos	A	::=	$P \mid A \supset B \mid [s]A$
certificados	s, t	::=	$a \mid v^{\circ} \mid s \cdot t \mid \lambda a : A.s \mid !s \mid letc\ s\ be\ v^{\circ} : A\ in\ t \mid fetch(s)$
valores	V	::=	$box_s\, M \mid \lambda a.M$
términos	M, N	::=	$a \mid v^{\bullet} \mid V \mid M\ N$
		$\mid$	$unpack\ M\ to\ \langle v^{\bullet}, v^{\circ}\rangle\ in\ N \mid fetch[w]\, M$

Σ es un conjunto finito de mundos que representan a cada uno de los nodos de la red. Cada $v : A@w$ en Δ es una hipótesis de validez, donde v es una variable que representa código móvil con certificado que calcula un valor de tipo A; el w indica que el código se utiliza en ese mundo. Cada $a : A@w$ en Γ es una hipótesis de verdad, dónde a es una variable que representa código que calcula un valor de tipo A en w. Es decir que Δ es un contexto de variables de código móvil, mientras que Γ es un contexto de variables de código local. Las variables en Δ y Γ se asumen

diferentes entre sí y frescas, es decir, que no aparecen en el tipo ubicado a su derecha. Además, los operadores $\supset$ y [ ] tienen más precedencia que el operador @, por lo que se abrevia $a : [s]A \supset [t]B@w$ para denotar la expresión $a : ([s]A \supset [t]B)@w$.

Los tipos pueden ser una variable de tipo P, el tipo de una abstracción $A \supset B$ que toma un valor de tipo A y devuelve un valor de tipo B ó el tipo de código móvil con certificado $[s]A$ que calcula un valor de tipo A con un certificado s.

Los certificados tienen dos tipos de variables. Las variables locales a son usadas para abstraer asunciones locales cuando se construyen certificados. Las variables de certificado v° representan certificados desconocidos. El certificado de la forma $s \cdot t$ representa la composición de los certificados s y t, mientras que el certificado $!s$ representa la aprobación del certificado s. A *unpack* M *to* $\langle v^\bullet, v^\circ \rangle$ *in* N se lo denomina certificado de validación y es la operación inversa a la aprobación. Finalmente, $fetch(s)$ es un certificado de traslado de código como se verá más adelante.

Los valores son un subconjunto de los términos que representan el resultado de computaciones. El sistema presentado distingue dos tipos de valores. Un valor de la forma $\lambda a.M$ es una abstracción o función. Como es usual en este tipo de definiciones, las ocurrencias libres de a en el término M están ligadas en $\lambda a.M$. El otro tipo de valor posible es el de la forma $box_s\, M$ el cual representa una *unidad móvil*, compuesta del código móvil M y del certificado s. Notar que esta es la única construcción que permite combinar los términos y los certificados.

Al igual que los certificados, los términos en $\lambda^{\mathsf{Cert}}_{\Box}$ tienen dos tipos de variables. Una variable de término a representa código local, mientras que una variable de término $v^\bullet$ representa código móvil. Como ya se mencionó antes, un valor V es un término. El término $M\, N$ representa la aplicación de funciones definida de la manera usual. Los términos de la forma *unpack* M *to* $\langle v^\bullet, v^\circ \rangle$ *in* N se denominan términos de desempaquetamiento y permiten extraer pares código-certificado de las unidades móviles. Aquí, M es el argumento del término, mientras que N es el cuerpo. De manera análoga al caso de las funciones, todas las ocurrencias libres de v° y de $v^\bullet$ en M se consideran ligadas en el término *unpack* M *to* $\langle v^\bullet, v^\circ \rangle$ *in* N. Finalmente, el término $fetch[w]\, M$ representa la operación de traslado de código, dónde el cuerpo M se traslada hacia el destino w. La definición precisa de la semántica de estas construcciones se describe más adelante en la sección 3.3. A continuación se presentan algunos ejemplos de términos en $\lambda^{\mathsf{Cert}}_{\Box}$:

Ejemplo 3.1. El siguiente ejemplo muestra un término que hace uso de varias de las construcciones definidas:

$$\lambda a.\lambda b.unpack\ a\ to\ \langle u^\bullet, u^\circ \rangle\ in\ (unpack\ b\ to\ \langle v^\bullet, v^\circ \rangle\ in\ (box_{u^\circ \cdot v^\circ}\, u^\bullet v^\bullet))$$

El mismo representa una función que, dadas una unidad móvil a y una unidad móvil b, extrae el código $v^\bullet$ y el certificado v° de b y extrae el código $u^\bullet$ y el certificado u° de a. Luego genera nuevo código $u^\bullet\ v^\bullet$ aplicando $u^\bullet$ a $v^\bullet$ y un nuevo certificado para este código: $u^\circ\ v^\circ$. Finalmente junta el código y el certificado en una nueva unidad móvil $box_{u^\circ \cdot v^\circ}\, u^\bullet v^\bullet$. La nueva unidad móvil es creada en el mundo actual w, el mismo mundo donde se asume que residen tanto a como b.

Ejemplo 3.2. El siguiente ejemplo introduce un término M en el cual se asume que las unidades móviles a y b residen en los mundos w_a y w_b, diferentes al mundo actual w:

$$\mathit{unpack}\ \mathit{fetch}[w_a]\, a\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}\ (\mathit{unpack}\ \mathit{fetch}[w_b]\, b\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ (\mathit{box}_{u^\circ \cdot v^\circ}\, u^\bullet v^\bullet))$$

Aquí, la expresión $\mathit{fetch}[w_a]\, a$ se interpreta como una llamada remota para computar el valor de a (una unidad móvil) en w_a y luego retornar al mundo actual. Notar que tanto a como b ocurren libres en esta expresión. Como b es un recurso que no es local (no reside en el mundo actual) el mismo no puede ser ligado de manera directa utilizando λb. Por el contrario, el código debe ser antes movido del mundo actual w al mundo w_b. Algo similar ocurre para a. Teniendo en cuenta que M es el término anterior, el termino final sin ocurrencias libres queda así:

$$\lambda a.\mathit{fetch}[w_b]\,(\lambda b.\mathit{fetch}[w]\, M)$$

Teniendo en cuenta que M reside en w, $\mathit{fetch}[w]\, M$ puede residir en cualquier mundo, en particular en w_b. Por lo tanto, $\lambda b.\mathit{fetch}[w]\, M$ reside en w_b. De manera análoga, $\mathit{fetch}[w_b]\,(\lambda b.\mathit{fetch}[w]\, M)$ puede residir en w_a, por lo que todo el término final reside en w_a

Antes de continuar con la presentación del cálculo $\lambda_{\Box}^{\mathsf{Cert}}$ es necesario introducir algunas definiciones. La primera de ellas es la de sustitución de variables locales en términos: $M\{a/N\}$.

Definición 3.2.

$$\begin{array}{rcl}
a\{a/N\} & = & N \\
b\{a/N\} & = & b \\
v^\bullet\{a/N\} & = & v^\bullet \\
(PQ)\{a/N\} & = & P\{a/N\}Q\{a/N\} \\
(\lambda a.P)\{a/N\} & = & \lambda a.P\{a/N\} \\
(\mathit{box}_t\, P)\{a/N\} & = & \mathit{box}_t\, P\{a/N\} \\
(\mathit{fetch}[w]\, P)\{a/N\} & = & \mathit{fetch}[w]\, P\{a/N\} \\
(\mathit{unpack}\ P\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}\ Q)\{a/N\} & = & \mathit{unpack}\ P\{a/N\}\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}\ Q\{a/N\}
\end{array}$$

De manera similar, se define la sustitución de variables móviles en términos: $M\{v^\bullet/N\}$

Definición 3.3.

$$\begin{array}{rcl}
a\{v^\bullet/N\} & = & a \\
v^\bullet\{v^\bullet/N\} & = & N \\
u^\bullet\{v^\bullet/N\} & = & u^\bullet \\
(PQ)\{v^\bullet/N\} & = & P\{v^\bullet/N\}Q\{v^\bullet/N\} \\
(\lambda a.P)\{v^\bullet/N\} & = & \lambda a.P\{v^\bullet/N\} \\
(\mathit{box}_t\, P)\{v^\bullet/N\} & = & \mathit{box}_t\, P\{v^\bullet/N\} \\
(\mathit{fetch}[w]\, P)\{v^\bullet/N\} & = & \mathit{fetch}[w]\, P\{v^\bullet/N\} \\
(\mathit{unpack}\ P\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}\ Q)\{v^\bullet/N\} & = & \mathit{unpack}\ P\{v^\bullet/N\}\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}\ Q\{v^\bullet/N\}
\end{array}$$

La noción de sustitución también se aplica sobre los certificados. La siguiente es la definición de la sustitución de variables de certificados en certificados: $t\{v^\circ/s\}$

Definición 3.4.

$$\begin{array}{rcl} a\{v^\circ/s\} & = & a \\ v^\circ\{v^\circ/s\} & = & s \\ u^\circ\{v^\circ/s\} & = & u^\circ \\ (t_1 \cdot t_2)\{v^\circ/s\} & = & t_1\{v^\circ/s\} \cdot t_2\{v^\circ/s\} \\ (\lambda a : A.t)\{v^\circ/s\} & = & \lambda a.t\{v^\circ/s\} \\ (!t)\{v^\circ/s\} & = & !t\{v^\circ/s\} \\ fetch(t)\{v^\circ/s\} & = & fetch(t\{v^\circ/s\}) \\ (letc\ t_1\ be\ u^\circ : A\ in\ t_2)\{v^\circ/s\} & = & letc\ t_1\{v^\circ/s\}\ be\ u^\circ : A\ in\ t_2\{v^\circ/s\} \end{array}$$

La siguiente definición establece la sustitución de certificados en términos: $M\{v^\circ/s\}$. Notar que el caso más interesante ocurre en el término $box_t\, P$.

Definición 3.5.

$$\begin{array}{rcl} a\{v^\circ/s\} & = & a \\ u^\bullet\{v^\circ/s\} & = & u^\bullet \\ (PQ)\{v^\circ/s\} & = & P\{v^\circ/s\}Q\{v^\circ/s\} \\ (\lambda a.P)\{v^\circ/s\} & = & \lambda a.P\{v^\circ/s\} \\ (box_t\, P)\{v^\circ/s\} & = & box_{t\{v^\circ/s\}}\, P\{v^\circ/s\} \\ (fetch[w]\, P)\{v^\circ/s\} & = & fetch[w]\, P\{v^\circ/s\} \\ (unpack\ P\ to\ \langle u^\bullet, u^\circ\rangle\ in\ Q)\{v^\circ/s\} & = & unpack\ P\{v^\circ/s\}\ to\ \langle u^\bullet, u^\circ\rangle\ in\ Q\{v^\circ/s\} \end{array}$$

Finalmente, esta última definición establece la sustitución de mundos en términos: $M\{w/w'\}$. Notar que aquí los casos más interesantes ocurren sobre el término $fetch[w]\, P$.

Definición 3.6.

$$\begin{array}{rcl} a\{w/w'\} & = & a \\ u^\bullet\{w/w'\} & = & u^\bullet \\ (PQ)\{w/w'\} & = & P\{w/w'\}Q\{w/w'\} \\ (\lambda a.P)\{w/w'\} & = & \lambda a.P\{w/w'\} \\ (box_t\, P)\{w/w'\} & = & box_t\, P\{w/w'\} \\ (fetch[w]\, P)\{w/w'\} & = & fetch[w']\, P\{w/w'\} \\ (fetch[w'']\, P)\{w/w'\} & = & fetch[w'']\, P\{w/w'\} \\ (unpack\ P\ to\ \langle u^\bullet, u^\circ\rangle\ in\ Q)\{w/w'\} & = & unpack\ P\{w/w'\}\ to\ \langle u^\bullet, u^\circ\rangle\ in\ Q\{w/w'\} \end{array}$$

3.2. Asignación de términos

Ahora que ha sido definida la sintaxis de los términos es posible dar una asignación de los mismos al sistema de ILPnd para obtener un sistema de tipos para el cálculo $\lambda^{\mathsf{Cert}}_{\Box}$. Esta asignación resulta básicamente de los esquemas en la definición 2.4 con términos codificando las derivaciones, localizando las hipótesis de Δ y Γ en mundos específicos y agregando una referencia al mundo actual. En el nuevo sistema, los juicios de tipado tienen la siguiente forma:

$$\Sigma; \Delta; \Gamma \rhd M : A@w \,|\, s$$

El mismo se lee: "Bajo las asunciones de validez en Δ y las asunciones de verdad en Γ, M tiene tipo A en w con prueba s". Además, un nuevo tipo de juicio es necesario, $\Sigma \vdash w$, que significa que el mundo w pertenece al Σ. A continuación se definen los esquemas de asignación de tipos para $\lambda^{\mathsf{Cert}}_{\Box}$:

Definición 3.7.

$$\frac{\Sigma \vdash w}{\Sigma; \Delta; \Gamma, a : A@w, \Gamma' \rhd a : A@w \,|\, a}\ \text{VarT}$$

$$\frac{\Sigma; \Delta; \Gamma, a : A@w \rhd M : B@w \,|\, s}{\Sigma; \Delta; \Gamma \rhd \lambda a.M : A \supset B@w \,|\, \lambda a : A.s} \supset I$$

$$\frac{\Sigma; \Delta; \Gamma \rhd M : A \supset B@w \,|\, s \quad \Sigma; \Delta; \Gamma \rhd N : A@w \,|\, t}{\Sigma; \Delta; \Gamma \rhd M\, N : B@w \,|\, s \cdot t} \supset E$$

$$\frac{\Sigma \vdash w}{\Sigma; \Delta, v : A@w, \Delta'; \Gamma \rhd v^{\bullet} : A@w \,|\, v^{\circ}}\ \text{VarV}$$

$$\frac{\Sigma; \Delta; \cdot \rhd M : A@w \,|\, s}{\Sigma; \Delta; \Gamma \rhd box_s\, M : [s]A@w \,|\, !s} \Box I \qquad \frac{\Sigma; \Delta; \Gamma \rhd M : [s]A@w' \,|\, t \quad \Sigma \vdash w}{\Sigma; \Delta; \Gamma \rhd fetch[w']\, M : [s]A@w \,|\, fetch(t)}\ \text{Fetch}$$

$$\frac{\Sigma; \Delta; \Gamma \rhd M : [r]A@w \,|\, s \quad \Sigma; \Delta, v : A@w; \Gamma \rhd N : C@w \,|\, t}{\Sigma; \Delta; \Gamma \rhd unpack\ M\ to\ \langle v^{\bullet}, v^{\circ} \rangle\ in\ N : C\{v^{\circ}/r\}@w \,|\, letc\ s\ be\ v : A\ in\ t} \Box E$$

Si se compara esta definición de las reglas de tipado con el sistema ILPnd se puede observar que la regla EqEvid ha sido eliminada. El motivo está relacionado con la forma en que se definirá la semántica para $\lambda^{\mathsf{Cert}}_{\Box}$. A diferencia de lo expuesto por el isomorfismo de Curry-Howard-DeBruijn, la semántica del nuevo cálculo no se basará en el proceso de normalización de pruebas en la lógica. Como ya se ha mencionado, la regla EqEvid es necesaria en la lógica para que el proceso de normalización de derivaciones sea una operación cerrada. Pero como la normalización no será tenida en cuenta, la regla no es necesaria.

El axioma VarT surge de la regla oVar de ILPnd asignando la variable de término a. La misma establece que si a es una variable local de tipo A en w, entonces a tiene tipo A en w con prueba a. El axioma VarV requiere una explicación más detallada. Ya se ha definido el concepto de unidades móviles para hacer énfasis en el hecho de que el código móvil siempre es acompañado por un certificado. Ya que las unidades móviles son expresiones de tipo modal y las variables de validez v representan valores de tipo modal, entonces las variables de validez pueden ser vistas como pares de la forma $\langle v^{\bullet}, v^{\circ} \rangle$, dónde $v^{\bullet}$ es el componente de código móvil y v° es el componente de certificado de la unidad móvil. Luego, el axioma modal mVar de ILPnd toma la siguiente forma:

$$\frac{\Sigma \vdash w}{\Sigma; \Delta, v : A@w, \Delta'; \Gamma \rhd v^\bullet : A@w \,|\, v^\circ}\ \text{VarV}$$

Los esquemas $\supset I$ y $\supset E$ forman abstracciones y aplicaciones en el mundo actual w. La aplicación de estos esquemas se ve reflejada en sus correspondientes certificados. El esquema $\Box I$ permite tipar unidades móviles. El mismo establece que si se cuenta con una derivación de M con tipo A en w que no depende de ninguna hipótesis local (es decir, Γ es vacío) y s es una prueba de esta afirmación, entonces M puede ser ejecutado en cualquier mundo. Por lo tanto, se introduce la unidad móvil $box_s\, M$ de tipo $[s]A$ y con certificado $!s$. La regla $\Box E$ permite eliminar hipótesis de validez utilizando un término *unpack*. Por un lado se tiene un término N que computa un valor de tipo C en el mundo w y que depende de una hipótesis de validez $v : A@w$. Por otro lado, se tiene un término M que computa una unidad móvil de tipo $[r]A$ en el mismo mundo w. Entonces el término *unpack* M *to* $\langle v^\bullet, v^\circ \rangle$ *in* N tiene tipo $C\{v^\circ/r\}$ en w con la hipótesis $v : A@w$ descartada. El certificado *letc* s *be* $v : A$ *in* t codifica la aplicación de este esquema.

A continuación se muestran algunos ejemplos de derivaciones de tipos en $\lambda_\Box^{\mathsf{Cert}}$:

Ejemplo 3.3. En este ejemplo se demuestra que el término del ejemplo 3.1

$$M = \lambda a.\lambda b.unpack\ a\ to\ \langle u^\bullet, u^\circ \rangle\ in\ (unpack\ b\ to\ \langle v^\bullet, v^\circ \rangle\ in\ (box_{u^\circ \cdot v^\circ}\, u^\bullet v^\bullet))$$

tiene tipo

$$C = [s](A \supset B) \supset [t]A \supset [s \cdot t]B$$

donde A y B son dos tipos cualquiera y s y t son los certificados de movilidad de los parámetros a y b respectivamente. Para ello se debe encontrar una derivación de

$$\Sigma; \cdot\,; \cdot \rhd M : C@w \,|\, s'$$

con $\Sigma = \{w\}$ y s' un certificado válido. Debido a la extensión de la derivación se la presentará por partes. En primer lugar se muestra la derivación del término *box* más interno de M:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Sigma \vdash w}{\Sigma; \Delta; \cdot \rhd u^\bullet : A \supset B@w \,|\, u^\circ}\ \text{VarV} \qquad \dfrac{\Sigma \vdash w}{\Sigma; \Delta; \cdot \rhd v^\bullet : A@w \,|\, v^\circ}\ \text{VarV}}{\Sigma; \Delta; \cdot \rhd u^\bullet\, v^\bullet : B@w \,|\, u^\circ \cdot v^\circ}\ \supset E}{\Sigma; \Delta; \Gamma \rhd box_{u^\circ \cdot v^\circ}\, u^\bullet\, v^\bullet : [u^\circ \cdot v^\circ]B@w \,|\, !(u^\circ \cdot v^\circ)}\ \Box I}{(\mathbf{1})}$$

donde $\Delta = \{u : A \supset B@w, v : A@w\}$ y $\Gamma = \{a : [s](A \supset B)@w, b : [t]A@w\}$. El segundo paso consiste en descartar la hipótesis v en Δ utilizando la regla $\Box E$:

$$\dfrac{(\mathbf{1}) \quad \dfrac{\Sigma \vdash w}{\Sigma; \Delta_u; \Gamma \rhd b : [t]A@w \,|\, b}\,\mathrm{VarT}}{\Sigma; \Delta_u; \Gamma \rhd \mathit{unpack}\ b\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ \mathit{box}_{u^\circ \cdot v^\circ}\, u^\bullet\, v^\bullet : [s \cdot v^\circ]B@w \,|\, \mathit{letc}\ b\ \mathit{be}\ v : A\ \mathit{in}\ !(u^\circ \cdot v^\circ)}\,\Box E$$

$$(\mathbf{2})$$

donde $\Delta_u = \{u : A \supset B@w\}$. Análogamente se descarta la hipótesis u en Δ:

$$\dfrac{(\mathbf{2}) \quad \dfrac{\Sigma \vdash w}{\Sigma; \cdot\,; \Gamma \rhd a : [s]A \supset B@w \,|\, a}\,\mathrm{VarT}}{\Sigma; \cdot\,; \Gamma \rhd \mathit{unpack}\ a\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}\ (\mathit{unpack}\ b\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ (\mathit{box}_{u^\circ \cdot v^\circ}\, u^\bullet v^\bullet)) : [s \cdot t]B@w \,|\, t'}\,\Box E$$

$$(\mathbf{3})$$

donde $t' = \mathit{letc}\ a\ \mathit{be}\ u : A \supset B\ \mathit{in}\ \mathit{letc}\ b\ \mathit{be}\ v : A\ \mathit{in}\ !(u^\circ v^\circ)$. Finalmente se descartan las variables a y b de Γ con sucesivas aplicaciones de la regla $\supset I$:

$$\dfrac{\dfrac{(\mathbf{3})}{\Sigma; \cdot\,; \Gamma_{-b} \rhd \lambda b.P : [t]A \supset [s \cdot t]B@w \,|\, \lambda b : [t]A@.t'}\supset I}{\Sigma; \cdot\,; \cdot \rhd M : C@w \,|\, s'}\supset I$$

donde
$\Gamma_{-b} = \{a : [s]A \supset B@w\}$,
$P = \mathit{unpack}\ a\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}\ (\mathit{unpack}\ b\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ (\mathit{box}_{u^\circ \cdot v^\circ}\, u^\bullet v^\bullet))$ y
$s' = \lambda a : [s]A \supset B.\lambda b : [t]A.t'$ (este último es el certificado de la derivación).

El siguiente ejemplo muestra como no es posible tipar unidades móviles con un certificado inválido, es decir, con un certificado que no se corresponde con el código.

Ejemplo 3.4. Para este ejemplo se considera el término $M = \lambda a.\lambda b.a\ b$ y el certificado $s = a \cdot b$. El objetivo es ver que para un mundo w, el término $\mathit{box}_s\, M$ no se puede tipar, es decir, no existe ni un tipo B ni un certificado t tales que el siguiente juicio sea demostrable:

$$\Sigma; \cdot; \cdot \rhd \mathit{box}_s\, M : A@w \,|\, t$$

Para justificar lo antes dicho, se razona inspeccionando las reglas de tipado. Para que el juicio de arriba sea verdadero, por la regla $\Box I$, debe existir un C tal que $B = [s]C$, $t = !s$ y además valga que

$$\Sigma; \cdot; \cdot \rhd M : C@w \,|\, s$$

Lo que equivale a

$$\Sigma; \cdot; \cdot \rhd \lambda a.\lambda b.a\ b : C@w \,|\, a \cdot b$$

Pero este juicio nunca puede ser verdadero, ya que por la regla $\supset I$ el certificado $a \cdot b$ debería ser de la forma $\lambda a : A.r$. Por lo tanto, el término $box_s\, M$ no se puede tipar.

El ejemplo anterior es sencillo, pero sirve para entender cómo la generación de certificados acompaña a la generación de código en $\lambda^{\mathsf{Cert}}_{\Box}$. No es posible generar unidades móviles bien tipadas que tengan certificados inválidos. Es el propio sistema de tipos el que asegura la correcta construcción de certificados. Desde el punto de vista de PCC, se puede ver al sistema de tipos como el conjunto de reglas de seguridad que establece el consumidor. De esta manera, bastará con saber que una unidad móvil está bien tipada para poder ejecutar el código asociado sin riesgos en la seguridad. Este esquema se diferencia de la formulación original de PCC en donde los certificados deben ser construidos por el productor aparte y ser verificados por el consumidor antes de ejecutar el código. En $\lambda^{\mathsf{Cert}}_{\Box}$ la verificación del certificado no es necesaria.

3.3. Semántica operacional

En esta sección se describe la semántica operacional para el cálculo $\lambda^{\mathsf{Cert}}_{\Box}$. Como ya se mencionó anteriormente, no se utilizará la normalización de pruebas de la lógica para definir la semántica sino que se utilizará otro enfoque siguiendo las ideas en [VCHP04]. Para definir la semántica se deben modelar los pasos de computación de un programa distribuido sobre una red de nodos. Para ello se define una máquina abstracta que se ejecuta sobre un conjunto de mundos w_i que representan a cada uno de los nodos de la red. La máquina realiza una computación secuencial del programa, donde cada paso tiene lugar en un mundo determinado.

Notar que no se están modelando ejecuciones concurrentes de programas, sino computaciones secuenciales que tienen en cuenta la existencia de varios nodos. La especificación de la semántica consiste entonces en una definición para los estados de la máquina y una definición de transición entre estos estados. La definición de los estados es la siguiente:

Definición 3.8 (Sintaxis de la máquina abstracta)**.**

$\mathbb{N}$	::=	$\mathbb{W}\,;w : [k, M]$	*estado de la máquina*
$\mathbb{W}$	::=	$\{w_1 : C_1, \ldots w_n : C_n\}$	*ambiente*
k	::=	return $w \mid$ finish $\mid k \triangleleft l$	*contexto*
l	::=	$\circ\ N \mid V\ \circ \mid$ *unpack* $\circ$ *to* $\langle v^{\bullet}, v^{\circ}\rangle$ *in* N	*capa*
C	::=	$\epsilon \mid C::k$	*pila de contextos*

Un estado de la máquina abstracta es una expresión de la forma $\mathbb{W}\,;w : [k, M]$. El mundo w indica el nodo donde se está efectuando la computación. El término M es el código que se está ejecutando bajo el contexto local k y se lo denomina foco de la computación. El contexto k es una pila de términos con agujeros (representados con un "$\circ$") que representan las capas de términos que se van quitando para acceder al redex. La necesidad del uso de contextos está relacionada con la forma en que

la máquina realiza la evaluación de los términos. Por ejemplo, la evaluación del término $M\,N$ introducirá en el contexto la capa $\circ\, N$. El contexto ahora espera la evaluación de un término lambda y, cuando esto suceda, se empezará a evaluar N. Cuando se termine de evaluar N se realizará la sustitución por lo que la estrategia de sustitución utilizada es call-by-value. Volviendo a la definición de contextos, los mismos pueden finalizar con un return w o con un finish. El contexto return w es introducido al evaluar un término de la forma $fetch[w]\,M$ e indica que una vez que el término que se encuentra en el foco de la computación es evaluado a un valor V, este último debe ser retornado al mundo w. Cuando k es de la forma finish significa que el valor del término actualmente en foco es el resultado final de la computación. Finalmente, $k \lhd l$ significa que la última capa que se ha extraído es l. Esta última puede tener las siguientes formas: $\circ\ N$ indica un argumento pendiente; $V\ \circ$ indica una abstracción pendiente (el hecho de que V es una abstracción y no una unidad móvil es asegurado por el sistema de tipos); $unpack\ \circ\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N$ indica el cuerpo de un término de desempaquetamiento pendiente.

Finalmente, $\mathbb{W}$ es el ambiente de la red y codifica el estado actual de ejecución de todos los nodos. El dominio de $\mathbb{W}$ es el conjunto de nodos al que se refiere. Por cada w del dominio se tiene una pila de contextos C. Cada vez que la computación del programa requiere un cambio de mundo, el contexto del mundo actual se apila para ser retomado cuando se retorne del mundo invocado. El uso de pilas de contextos está justificado en el hecho de que la evaluación es reentrante en el sentido de que el código que se invoca en un mundo puede a su vez invocar código en el mundo original.

Dado un conjunto de mundos $\Sigma = \{w_1, \ldots, w_n\}$, el estado inicial de una máquina es $\mathbb{W}\,; w : [finish, M]$, donde $\mathbb{W} = \{w_1 : \epsilon, \cdots, w_n : \epsilon\}$, w es cualquier mundo de Σ y M es cualquier término. Un estado de la máquina es *terminal* si es de la forma $\mathbb{W}\,; w : [finish, V]$. Notar que en un estado final el foco de la computación es un término evaluado en su totalidad, es decir, un valor.

La semántica operacional se termina de definir con una relación de reducción entre estados de la máquina que se presentan a continuación:

Definición 3.9 (Pasos de reducción de la máquina abstracta)**.**

$$
\begin{array}{lrcl}
(1) & \mathbb{W}; w : [k, M\,N] & \longrightarrow & \mathbb{W}; w : [k \lhd \circ\ N, M] \\
(2) & \mathbb{W}; w : [k \lhd \circ\ N, V] & \longrightarrow & \mathbb{W}; w : [k \lhd V\ \circ, N] \\
(3) & \mathbb{W}; w : [k \lhd (\lambda a.M)\ \circ, V] & \longrightarrow & \mathbb{W}; w : [k, M\{a/V\}] \\
(4) & \mathbb{W}; w : [k, unpack\ M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N] & \longrightarrow & \mathbb{W}; w : [k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N, M] \\
(5) & \mathbb{W}; w : [k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N, box_s\,M] & \longrightarrow & \mathbb{W}; w : [k, N\{v^\circ/s\}\{v^\bullet/M\}] \\
(6) & \{w : C; w_s\}; w : [k, fetch[w']\,M] & \longrightarrow & \{w : C :: k; w_s\}; w' : [\text{return } w, M] \\
(7) & \{w : C :: k; w_s\}; w' : [\text{return } w, V] & \longrightarrow & \{w : C; w_s\}; w : [k, V\{w'/w\}]
\end{array}
$$

La explicación de las reglas de reducción es la siguiente. Los esquemas (1), (2) y (3) permiten la evaluación de funciones. En primer lugar, el esquema (1) permite reducir un término de aplicación $M\,N$. Este esquema selecciona para seguir reduciendo el término de más a la izquierda de la aplicación (M) y apila en el contexto el argumento como término pendiente a evaluar ($\circ\, N$). El esquema (2) se puede aplicar una vez que se obtiene un valor V de la reducción de la parte más a la izquierda

de una aplicación. El sistema de tipos asegurará que este valor sea una abstracción. En ese caso el argumento pendiente se quita de la pila para ser evaluado y a su vez se introduce la abstracción en el contexto. Finalmente, cuando el argumento de la aplicación se reduce a un valor, se puede aplicar el esquema (3) donde la abstracción se desapila del contexto y se aplica una beta-reducción cuyo resultado se posiciona como foco del próximo paso de reducción.

El esquema (4) permite reducir términos de la forma *unpack* M *to* $\langle v^\bullet, v^\circ \rangle$ *in* N. En este caso, el argumento M se evalúa primero ubicándolo como foco de la computación, mientras que el resto del término se apila en el contexto. Cuando la reducción del argumento de un término de desempaquetamiento computa un valor que es una unidad móvil, se puede aplicar el esquema (5). Este esquema desapila del contexto el cuerpo del término de desempaquetamiento N, extrae el certificado s y el cuerpo M de la unidad móvil y los sustituye en N. El término resultado de la sustitución se ubica como foco de la computación.

Hasta aquí los esquemas vistos realizan todas las computaciones localmente, es decir, los pasos de reducción mantienen entre los estados de la máquina el mundo w donde se realiza la computación. Los esquemas (6) y (7) permiten operar reducciones entre dos mundos diferentes. El primero de estos esquemas supone como foco de la computación un término de la forma $fetch[w']\, M$ y un ambiente de ejecución que contiene una pila de contextos C en el mundo actual w. En este caso, el contexto de ejecución actual k se apila en C y el control se transfiere al mundo w'. El foco de la aplicación pasa a ser M y se establece como contexto a return w. El esquema (7) se puede aplicar una vez que computa un valor V a partir del cuerpo de un fetch. El control se transfiere de vuelta al mundo que realizó la invocación remota, desapilando su contexto del ambiente. El nuevo foco de computación pasa a ser el valor V dónde las ocurrencias de w' son sustituidas por w.

Para terminar de definir formalmente la semántica operacional es necesario introducir un sistema de tipos que establezca las reglas para construcción correcta de estados de la máquina Para ello se introducen dos nuevos juicios, uno para estados de máquina otro para ambiente de la red:

- $\Sigma \vdash \mathbb{W}\,; w_j : [k, M]$
- $\Sigma \vdash \mathbb{W}\,; k : A@w_j$

El primero de los juicios establece que $\mathbb{W}\,; w_j : [k, M]$ es un estado de máquina bien tipado bajo el conjunto de mundos Σ. El segundo establece que el ambiente de red $\mathbb{W}$ junto con el contexto local k está bien tipado bajo el conjunto de mundos Σ. La lectura intuitiva de este juicio es que el contexto k (junto al ambiente $\mathbb{W}$) espera un valor de tipo A en el mundo w_j. A continuación se definen los esquemas de tipado para estos juicios:

Definición 3.10 (Reglas de tipado para estados de la máquina)**.**

$$\frac{}{\Sigma \vdash \mathbb{W}\,;\mathrm{finish} : A@w}\ C.Finish$$

$$\frac{\Sigma \vdash \mathbb{W}\,; k : B@w \quad \Sigma;\cdot\,;\cdot \rhd N : A@w \mid s}{\Sigma \vdash \mathbb{W}\,; k \lhd \circ\ N : A \supset B@w}\ C.Abs$$

$$\frac{\Sigma \vdash \mathbb{W}\,; k : B@w \quad \Sigma;\cdot\,;\cdot \rhd V : A \supset B@w \mid s}{\Sigma \vdash \mathbb{W}\,; k \lhd V\ \circ : A@w}\ C.App$$

$$\frac{\Sigma \vdash \mathbb{W}\,; k : B\{v^\circ/t\}@w \quad \Sigma; v : A@w;\cdot \rhd N : B@w \mid s}{\Sigma \vdash \mathbb{W}\,; k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N : [t]A@w}\ C.Box$$

$$\frac{\Sigma \vdash \{w' : C; w_s\}\,; k : A@w'}{\Sigma \vdash \{w' : C :: k; w_s\}\,;\mathrm{return}\ w' : A@w}\ C.Return$$

$$\frac{\begin{array}{c}\Sigma = \{w_1, \ldots, w_n\} \qquad \mathbb{W} = \{w_1 : C_1, \ldots w_n : C_n\} \\ \Sigma;\cdot\,;\cdot \rhd M : A@w_j \mid s \qquad \Sigma \vdash \mathbb{W}\,; k : A@w_j\end{array}}{\Sigma \vdash \mathbb{W}\,; w_j : [k, M]}\ \mathit{MState}$$

Una explicación de las reglas es la siguiente. La regla *MState* establece cuándo un estado de máquina $\Sigma \vdash \mathbb{W}\,; w_j : [k, M]$ está bien tipado. Para ello se deben cumplir tres requerimientos. El primero de ellos es que $\mathbb{W}$ sea un ambiente de red cuyo conjunto dominio coincida con Σ. En segundo lugar, deben existir un tipo A y un certificado s tal que M sea un término cerrado bien tipado en w_j con certificado s de tipo A. Por último, el ambiente de red $\Sigma \vdash \mathbb{W}\,; k : A@w_j$ debe estar bien tipado. Es decir, M tiene el tipo que espera el contexto k.

El axioma *C.Finish* establece que el contexto finish puede ser tipado con cualquier tipo A y mundo w. O sea que el contexto vacío puede esperar valores de cualquier tipo. La regla *C.Abs* permite tipar ambientes de red con un contexto de la forma $k \lhd \circ\ N$. Puesto que N es un argumento pendiente de evaluar de una aplicación, este contexto espera un tipo función de la forma $A \supset B$ en el mundo w, siempre y cuando N esté bien tipado con tipo A en w y B sea el tipo del ambiente con contexto k (es decir, k espera un valor de tipo B). La regla *C.App* establece que el contexto $k \lhd V \circ$ espera un valor de tipo A en w siempre y cuando el contexto k espere un valor de tipo B en w y el valor V sea una abstracción de tipo $A \supset B$ en w. La regla *C.Box* dice que un contexto de la forma $k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N$ siempre espera una unidad móvil con un tipo de la forma $[t]A$. Para ello el término N debe estar bien tipado con tipo B con la hipótesis de validez $v : A@w$ y el contexto k debe esperar un valor de tipo $B\{v^\circ/t\}$. La necesidad de esta última sustitución surge a partir de la definición de la regla $\Box E$. Finalmente, la regla *C.Return* permite tipar ambientes con contexto return w'. Este contexto espera un valor de tipo A en w siempre y cuando el último contexto apilado del mundo w' en el ambiente, k, también espere un valor de tipo A, pero en w'.

A continuación se presenta un ejemplo donde se muestra el funcionamiento de la máquina abstracta:

Ejemplo 3.5. Para el ejemplo se considerará la siguiente función

$$P = \lambda f.\lambda x.\lambda y.unpack\ x\ to\ \langle v^\bullet, v^\circ\rangle\ in\ (unpack\ y\ to\ \langle u^\bullet, u^\circ\rangle\ in\ f\ v^\bullet\ u^\bullet)$$

Esta función toma tres valores como parámetro: una función f y dos unidades móviles x e y. Luego extrae el código de las unidades móviles y los aplica a la función f. El tipo de la función es el siguiente:

$$(A \supset B \supset C) \supset [s]A \supset [t]B \supset C@w$$

La idea para el ejemplo es encontrar parámetros para P que permitan ver reducciones de la máquina que impliquen traslados de código. Para simplificar el ejemplo se asume la existencia de tres términos a, b y c con tipos A, B y C (y certificados s_a, s_b y s_c) respectivamente. Como primer parámetro se considerará el término $Q = \lambda m.\lambda n.c$ el cual tiene tipo $A \supset B \supset C$ en w (con certificado $\lambda m : A.\lambda n : B.s_c$). Como unidades móviles se considerarán dos términos los cuales computan sus valores en mundos diferentes al actual: el término $R = fetch[w_1]\ box_{s_a}\ a$ que tiene tipo $[s_a]A$ en w (con certificado $fetch(!s_a)$) y el término $S = fetch[w_2]\ box_{s_b}\ b$ que tiene tipo $[s_b]B$ en w (con certificado $fetch(!s_b)$). Luego, el término $M = P\ Q\ R\ S$ tiene tipo C en w y, como se demostrará a continuación, reduce al término c:

Para simplificar la notación se consideran las siguientes igualdades:

$\mathbb{W} = \{w : \epsilon, w_1 : \epsilon, w_2 : \epsilon\}$
$P_{-f} = \lambda x.\lambda y.unpack\ x\ to\ \langle v^\bullet, v^\circ\rangle\ in\ (unpack\ y\ to\ \langle u^\bullet, u^\circ\rangle\ in\ Q\ v^\bullet u^\bullet)$
$P_{-fx} = \lambda y.unpack\ (box_{s_a}\ a)\ to\ \langle v^\bullet, v^\circ\rangle\ in\ (unpack\ y\ to\ \langle u^\bullet, u^\circ\rangle\ in\ Q\ v^\bullet u^\bullet)$
$P_{-fxy} = unpack\ (box_{s_a}\ a)\ to\ \langle v^\bullet, v^\circ\rangle\ in\ (unpack\ (box_{s_b}\ b)\ to\ \langle u^\bullet, u^\circ\rangle\ in\ Q\ v^\bullet u^\bullet)$
$k = unpack\ \circ\ to\ \langle v^\bullet, v^\circ\rangle\ in\ (unpack\ (box_{s_b}\ b)\ to\ \langle u^\bullet, u^\circ\rangle\ in\ Q\ v^\bullet u^\bullet)$

La ejecución de la máquina procede de la siguiente manera:

1.	$\mathbb{W}; w : [\text{finish}, M]$	$\longrightarrow$	(1)
2.	$\mathbb{W}; w : [\text{finish} \lhd \circ\ S, PQR]$	$\longrightarrow$	(1)
3.	$\mathbb{W}; w : [\text{finish} \lhd \circ\ S \lhd \circ\ R, PQ]$	$\longrightarrow$	(1)
4.	$\mathbb{W}; w : [\text{finish} \lhd \circ\ S \lhd \circ\ R \lhd \circ\ Q, P]$	$\longrightarrow$	(2)
5.	$\mathbb{W}; w : [\text{finish} \lhd \circ\ S \lhd \circ\ R \lhd P\ \circ, Q]$	$\longrightarrow$	(3)
6.	$\mathbb{W}; w : [\text{finish} \lhd \circ\ S \lhd \circ\ R, P_{-f}]$	$\longrightarrow$	(2)
7.	$\mathbb{W}; w : [\text{finish} \lhd \circ\ S \lhd P_{-f}\ \circ, fetch[w_1]\ box_{s_a}\ a]$	$\longrightarrow$	(6)
8.	$\{w : \text{finish} \lhd \circ\ S \lhd P_{-f}\ \circ, w_1 : \epsilon, w_2 : \epsilon\}; w_1 : [\text{return}\ w, box_{s_a}\ a]$	$\longrightarrow$	(7)
9.	$\mathbb{W}; w : [\text{finish} \lhd \circ\ S \lhd P_{-f}\ \circ, box_{s_a}\ a]$	$\longrightarrow$	(3)
10.	$\mathbb{W}; w : [\text{finish} \lhd \circ\ S, P_{-fx}]$	$\longrightarrow$	(2)
11.	$\mathbb{W}; w : [\text{finish} \lhd P_{-fx}\ \circ, fetch[w_2]\ box_{s_b}\ b]$	$\longrightarrow$	(6)
12.	$\{w : \text{finish} \lhd P_{-fx}\ \circ, w_1 : \epsilon, w_2 : \epsilon\}; w_2 : [\text{return}\ w, box_{s_b}\ b]$	$\longrightarrow$	(7)
13.	$\mathbb{W}; w : [\text{finish} \lhd P_{-fx}\ \circ, box_{s_b}\ b]$	$\longrightarrow$	(3)
14.	$\mathbb{W}; w : [\text{finish}, P_{-fxy}]$	$\longrightarrow$	(4)
15.	$\mathbb{W}; w : [\text{finish} \lhd k, box_{s_a}\ a]$	$\longrightarrow$	(5)
16.	$\mathbb{W}; w : [\text{finish}, unpack\ (box_{s_b}\ b)\ to\ \langle u^\bullet, u^\circ\rangle\ in\ Q\ a\ u^\bullet]$	$\longrightarrow$	(4)

17. $\mathbb{W}; w : [\text{finish} \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle u^{\bullet}, u^{\circ}\rangle\ \mathit{in}\ Q\ a\ u^{\bullet}, \mathit{box}_{s_b}\, b]$ $\longrightarrow$ (5)
18. $\mathbb{W}; w : [\text{finish}, Q\ a\ b]$ $\longrightarrow$ (1)
19. $\mathbb{W}; w : [\text{finish} \lhd \circ\ b, Q\ a\}]$ $\longrightarrow$ (1)
20. $\mathbb{W}; w : [\text{finish} \lhd \circ\ b \lhd \circ\ a, Q]$ $\longrightarrow$ (2)
21. $\mathbb{W}; w : [\text{finish} \lhd \circ\ b \lhd Q\ \circ, a]$ $\longrightarrow$ (3)
22. $\mathbb{W}; w : [\text{finish} \lhd \circ\ b, \lambda y.c]$ $\longrightarrow$ (2)
23. $\mathbb{W}; w : [\text{finish} \lhd \lambda y.c\ \circ, b]$ $\longrightarrow$ (3)
24. $\mathbb{W}; w : [\text{finish}, c]$

En este capítulo se introdujo $\lambda_{\Box}^{\mathsf{Cert}}$ a partir de la interpretación computacional de ILPnd. A partir de la definición de la sintaxis del lenguaje se dió una asignación de términos a las reglas de la lógica con el objetivo de obtener un sistema de tipos que incluye la generación de certificados. Además, se presentó la semántica operacional del lenguaje. La definición precisa del cálculo permite estudiar formalmente propiedades sobre el mismo, tal cual como se detalla en el próximo capítulo.

Capítulo 4

Propiedades

En este capítulo se estudian las propiedades principales de $\lambda_{\Box}^{\mathsf{Cert}}$. En primer lugar se estudian tres principios de sustitución que se cumplen dentro del cálculo y que sirven para demostrar otras proposiciones importantes. Luego se se estudia la seguridad de tipos, consistente en demostrar progreso y preservación de tipos (subject reduction). Finalmente, se demuestra normalización fuerte de la reducción de la máquina abstracta. A lo largo del capítulo se muestran detalles de las demostraciones más trascendentes, dejando para el apéndice aquellas menos importantes.

4.1. Principios de sustitución

Antes de poder enunciar las propiedades respecto a seguridad de tipos es necesario presentar algunas propiedades importantes denominadas principios de sustitución.

El primer principio de sustitución que se considera es el de sustitución de mundos, que es una propiedad que permite, a partir de una derivación de tipos, obtener otra simplemente sustituyendo un mundo por otro. La segunda parte de la proposición considera un caso particular, donde w'' es igual a w.

Proposición 1 (Principio de sustitución de mundos)**.** *Si* $\Sigma; \Delta; \Gamma \rhd M : A@w'' \mid s$ *es derivable y* $\Sigma \vdash w, w'$*, entonces* $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd M\{w/w'\} : A@w''\{w/w'\} \mid s$ *es derivable.*

En particular, vale que si $\Sigma; \Delta; \Gamma \rhd M : A@w \mid s$ *es derivable y* $\Sigma \vdash w'$*, entonces* $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd M\{w/w'\} : A@w' \mid s$ *también es derivable.*

Antes de enunciar el segundo de los principios de sustitución es necesario el siguiente lema auxiliar que establece que si M es tipable, entonces todas sus variables libres aparecen en el dominio de Γ:

Lema 1. *Si* $\Sigma; \Delta; \Gamma \rhd M : A@w \mid s$*, entonces se cumple que si* $a \in fv(M)$ *(*$a \in fv(s)$*) entonces* $a \in dom(\Gamma)$

El segundo principio de sustitución parte de la suposición de que un término N tiene tipo B en un mundo w' con una hipótesis de verdad de que a tiene tipo A

en w. Si, por otro lado, se tiene un término M con tipo a en w, ese término M puede ser sustituido por a en N en la derivación original para obtener una nueva derivación que ya no depende de la hipótesis sobre a. También se realiza sustitución de los certificados correspondientes:

Proposición 2 (Principio de sustitución para hipótesis verdaderas)**.** *Si* $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd M : A@w \mid s$ *y* $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd N : B@w' \mid t$ *son derivables, entonces también lo es* $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd N\{a/M\} : B@w' \mid t\{a/s\}$.

Para demostrar el último de los principios de sustitución, hace falta otro lema auxiliar que permite agregar hipótesis de verdad (weakening):

Lema 2. *Si* $\Sigma; \Delta; \Gamma \rhd M : A@w \mid s$ *es derivable,* $x \notin Dom(\Gamma)$ *y* $\Sigma \vdash w$ *, entonces* $\Sigma; \Delta; \Gamma, x : A@w \rhd M : A@w \mid s$ *también es derivable.*

El tercer y último principio de sustitución es análogo al anterior, considerando una hipótesis de validez en lugar de una hipótesis de verdad. Notar que la derivación del término con que se sustituye (M) no debe tener asunciones de verdad:

Proposición 3 (Principio de sustitución para hipótesis válidas)**.** *Si* $\Sigma; \Delta_1, \Delta_2; \cdot \rhd M : A@w \mid s$ *y* $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd N : B@w' \mid t$ *son derivables, entonces también lo es* $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd N\{v^\circ/s\}\{v^\bullet/M\} : B\{v^\circ/s\}@w' \mid t\{v^\circ/s\}$.

4.2. Seguridad de tipos o *type safety*

La seguridad de tipos o *type safety* consiste en garantizar que un programa que ha sido bien tipado no puede fallar al momento de ser ejecutado, es decir, no genera excepciones en tiempo de ejecución. El motivo de falla que se interesa obviar, además del caso estándar en que una función recibe un parámetro de un tipo que no esperaba, es la generación de código móvil junto a un certificado de manera tal que ese certificado no se corresponde con el código. Para demostrar seguridad en tipos se probarán las siguientes dos propiedades: progreso y preservación de tipos.

4.2.1. Progreso

La propiedad de progreso o *progress* consiste en que la máquina abstracta nunca se quede trabada en ningún estado que esté bien tipado. Es decir, que la máquina nunca alcance un estado no terminal desde el cual no sea posible realizar ninguna reducción. La siguiente proposición enuncia formalmente la propiedad:

Proposición 4 (Progreso)**.** *Si* $\Sigma \vdash \mathbb{N}$ *es derivable y* $\mathbb{N}$ *no es terminal, entonces existe* $\mathbb{N}'$ *tal que* $\mathbb{N} \longrightarrow \mathbb{N}'$.

Demostración. La demostración se realiza por análisis de casos sobre k y M.

Si $\Sigma \vdash \mathbb{N}$, entonces existen A, s tales que:

(a) $\Sigma; \cdot; \cdot \rhd M : A@w \mid s$

(b) $\Sigma \vdash \mathbb{W}\,; k : A@w$

De (a) $M \neq a, v^\bullet$ ya que Γ, Δ son vacíos. Por lo tanto se consideran los casos restantes de M y k.

- Caso 1: M es un valor $V = box_t\, P$ o $V = \lambda x.P$.
  - Subcaso 1.1: $k =$ finish. $\mathbb{N}$ es terminal y por lo tanto el resultado vale.
  - Subcaso 1.2: $k = k' \lhd \circ\ N$. Por el esquema de reducción de la máquina (2), $\mathbb{N} \to \mathbb{W}\,; w : [k' \lhd V\ \circ, N]$.
  - Subcaso 1.3: $k = k' \lhd V'\ \circ$. Por la regla de tipos $C.App$ existen B, t' tales que $\Sigma; \cdot; \cdot \rhd V' : A \supset B@w \,|\, t'$. Por lo tanto, de $\supset I$, $V' = \lambda b.N$. Luego, por el esquema de reducción (3), $\mathbb{N} \to \mathbb{W}\,; w : [k', N\{a/V\}]$.
  - Subcaso 1.4: $k = k' \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N$. Por la regla $C.Box$ existen t', A' tales que $A = [t']A'$. Por lo tanto $V = box_t\, P$. Luego, por el esquema de reducción (5), $\mathbb{N} \to \mathbb{W}\,; w : [k', N\{v^\circ/s\}\{v^\bullet/M\}]$.
  - Subcaso 1.5: $k =$ return w'. Por la regla $C.Return$ $\mathbb{W} = \{w' : C :: k'; w_s\}$. Luego, por el esquema de reducción (7) $\mathbb{N} \to \{w' : C; w_s\}\,; w' : [k', V]$.
- Case 2: $M = PQ$. Por el esquema de reducción (1), $\mathbb{N} \to \mathbb{W}\,; w : [k \lhd \circ\ Q, P]$.
- Case 3: $M = unpack\ P\ to\ \langle v^\bullet, v^\circ \rangle\ in\ Q$. Por el esquema de reducción (5), $\mathbb{N} \to \mathbb{W}\,; w : [k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ Q, P]$.
- Case 4: $M = fetch[w']\, P$. Por el esquema de reducción (6), $\mathbb{N} \to \{w : C :: k; w_s\}\,; w' : [\text{return } w, P]$.

□

4.2.2. Preservación de tipos

La propiedad de preservación de tipos o *subject reduction* consiste en que cualquier estado bien tipado de la máquina sea cerrado con respecto a la reducción. Es decir, que si se realiza una reducción desde un estado bien tipado a otro estado, entonces este último también es bien tipado. La siguiente proposición describe formalmente la propiedad:

Proposición 5 (Preservación de tipos)**.** *Si* $\Sigma \vdash \mathbb{N}$ *es derivable y* $\mathbb{N} \longrightarrow \mathbb{N}'$*, entonces* $\Sigma \vdash \mathbb{N}'$ *es derivable.*

Demostración. La demostración se realiza por análisis de casos sobre el paso de reducción aplicado. La misma hace uso de las 3 propiedades de sustitución (de hipótesis verdaderas, de hipótesis válidas y de sustitución de mundos).

- Caso 1:
  - $\mathbb{N} = \mathbb{W}\,; w : [k, MN]$

- $\mathbb{N}' = \mathbb{W}\,; w : [k \lhd \circ N, M]$

Supongamos que $\Sigma \vdash \mathbb{W}\,; w : [k, MN]$. Entonces por la regla *MState*, existen B, s' tales que:

(1.1) $\Sigma; \cdot\,; \cdot \rhd MN : B@w \,|\, s'$

(1.2) $\Sigma \vdash \mathbb{W}\,; k : B@w$

De (1.1) y por la regla $\supset E$, existen s, t, A tales que $s' = s.t$ y

(1.3) $\Sigma; \cdot\,; \cdot \rhd M : A \supset B@w \,|\, s$

(1.4) $\Sigma; \cdot\,; \cdot \rhd N : A@w \,|\, t$

De (1.2) y (1.4), por $C.Abs$:

(1.5) $\Sigma \vdash \mathbb{W}\,; k \lhd \circ N : A \supset B@w$

De (1.3) y (1.5) por *MState*

$$\Sigma \vdash \mathbb{W}\,; w : [k \lhd \circ N, M]$$

- Caso 2:

  - $\mathbb{N} = \mathbb{W}\,; w : [k \lhd \circ N, V]$
  - $\mathbb{N}' = \mathbb{W}\,; w : [k \lhd V \circ, N]$

Supongamos que $\Sigma \vdash \mathbb{W}\,; w : [k \lhd \circ N, V]$. Entonces por *MState* existen A', s' tales que:

(2.1) $\Sigma; \cdot\,; \cdot \rhd V : A'@w \,|\, s'$

(2.2) $\Sigma \vdash \mathbb{W}\,; k \lhd \circ N : A'@w$

De (2.2) y $C.Abs$ existen A, B, s tales que $A' = A \supset B$ y, además,

(2.3) $\Sigma \vdash \mathbb{W}\,; k : B@w$

(2.4) $\Sigma; \cdot\,; \cdot \rhd N : A@w \,|\, s$

De (2.1) y (2.3) por $C.App$:

(2.5) $\Sigma \vdash \mathbb{W}\,; k \lhd \circ V : A@w$

De (2.4) y (2.5) por *MState*

$$\Sigma \vdash \mathbb{W}\,; w : [k \lhd \circ V, N]$$

- Caso 3:

  - $\mathbb{N} = \mathbb{W}\,; w : [k \lhd (\lambda a.M) \circ, V]$

- $\mathbb{N}' = \mathbb{W}\,; w : [k, M\{a/V\}]$

Supongamos que $\Sigma \vdash \mathbb{W}\,; w : [k \triangleleft (\lambda a.M)\circ, V]$. Entonces por la regla *MState* existen A, s' tales que:

(3.1) $\Sigma; \cdot\,; \cdot \rhd V : A@w \,|\, s'$

(3.2) $\Sigma \vdash \mathbb{W}\,; k \triangleleft (\lambda a.M)\circ : A@w$

De (3.2) y $C.App$ existen B, t tales que:

(3.3) $\Sigma \vdash \mathbb{W}\,; k : B@w$

(3.4) $\Sigma; \cdot\,; \cdot \rhd (\lambda a.M) : A \supset B@w \,|\, t$

De (3.4) y por $\supset I$ existe s tal que $t = \lambda a : A.s$ y, además,

(3.5) $\Sigma; \cdot\,; a : A@w \rhd M : B@w \,|\, s$

De (3.1), (3.5) y por el principio de sustitución para hipótesis verdaderas (Lema 2)

(3.6) $\Sigma; \cdot\,; \cdot \rhd M\{a/V\} : B@w \,|\, s\{a/s'\}$

De (3.3) y (3.6) por *MState*

$$\Sigma \vdash \mathbb{W}\,; w : [k, M\{a/V\}]$$

- Caso 4:

  - $\mathbb{N} = \mathbb{W}\,; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N]$
  - $\mathbb{N}' = \mathbb{W}\,; w : [k \triangleleft \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, M]$

Supongampos que $\Sigma \vdash \mathbb{W}\,; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N]$. Entonces por la regla *MState* existen B, s' tales que:

(4.1) $\Sigma; \cdot\,; \cdot \rhd \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N : B@w \,|\, s'$

(4.2) $\Sigma \vdash \mathbb{W}\,; k : B@w$

De (4.1) y por $\Box E$ existen A, B', r, s, t tales que $s' = \mathit{letc}\ s\ \mathit{be}\ v : A\ \mathit{in}\ t$, $B = B'\{v^\circ/r\}$ y, además,

(4.3) $\Sigma; \cdot\,; \cdot \rhd M : [r]A@w \,|\, s$

(4.4) $\Sigma; v : A@w; \cdot \rhd N : B'@w \,|\, t$

De (4.2), (4.4) y la regla $C.Box$:

(4.5) $\Sigma \vdash \mathbb{W}\,; k \triangleleft \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N : [r]A@w$

De (4.3) y (4.5) por la regla *MState*

$$\Sigma \vdash \mathbb{W}\,;w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, M]$$

- Caso 5:
  - $\mathbb{N} = \mathbb{W}\,;w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, \mathit{box}_s\, M]$
  - $\mathbb{N}' = \mathbb{W}\,;w : [k, N\{v^\circ/s\}\{v^\bullet/M\}]$

  Supongamos que $\Sigma \vdash \mathbb{W}\,;w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, \mathit{box}_s\, M]$. Entonces por la regla *MState* existen A', s' tales que:

  (5.1) $\Sigma; \cdot\,; \cdot \rhd \mathit{box}_s\, M : A'@w \,|\, s'$

  (5.2) $\Sigma \vdash \mathbb{W}\,; k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N : A'@w$

  De (5.1) y por $\Box I$ existen s, A tales que $A' = [s]A$, $s' = !s$ y, además,

  (5.3) $\Sigma; \cdot\,; \cdot \rhd M : A@w \,|\, s$

  De (5.2) y por la regla $C.\mathit{Box}$ existen B, t tales que:

  (5.4) $\Sigma \vdash \mathbb{W}\,; k : B\{v^\circ/s\}@w$

  (5.5) $\Sigma; v : A@w; \cdot \rhd N : B@w \,|\, t$

  De (5.3) y (5.5) por el principio de sustitución para hipótesis válidas (Lema 3)

  (5.6) $\Sigma; \cdot\,; \cdot \rhd N\{v^\circ/s\}\{v^\bullet/M\} : B\{v^\circ/s\}@w \,|\, t\{v^\circ/s\}$.

  De (5.4) y (5.6) por la regla *MState* vale que

  $$\Sigma \vdash \mathbb{W}\,;w : [k, N\{v^\circ/s\}\{v^\bullet/M\}]$$

- Caso 6:
  - $\mathbb{N} = \{w : C; w_s\}\,;w : [k, \mathit{fetch}[w']\, M]$
  - $\mathbb{N}' = \{w : C :: k; \mathrm{w}_s\}\,;w' : [\text{return } w, M]$

  Supongamos que $\Sigma \vdash \{w : C; w_s\}\,;w : [k, \mathit{fetch}[w']\, M]$. Entonces por la regla *MState* existen A', s' tales que:

  (6.1) $\Sigma \vdash \{w : C; w_s\}\,; k : A'@w$

  (6.2) $\Sigma; \cdot\,; \cdot \rhd \mathit{fetch}[w']\, M : A'@w \,|\, s'$

  De (6.2) y por Fetch existen s, t, A tales que $A' = [s]A$, $s' = \mathit{fetch}(t)$ y, además,

  (6.3) $\Sigma; \cdot\,; \cdot \rhd M : [s]A@w' \,|\, t$

  De (6.1) y por $C.\mathit{Return}$:

  (6.4) $\Sigma \vdash \{w : C :: k; \mathrm{w}_s\}\,; \text{return } w : [s]A@w'$

De (6.3) y (6.4) por *MState*

$$\Sigma \vdash \{w : C :: k; \mathrm{w}_s\}\,; w' : [\text{return } w, M]$$

- Caso 7:

  - $\mathbb{N} = \{w : C :: k; \mathrm{w}_s\}\,; w' : [\text{return } w, V]$
  - $\mathbb{N}' = \{w : C; \mathrm{w}_s\}\,; w : [k, V\{w'/w\}]$

  Supongamos que $\Sigma \vdash \{w : C :: k; \mathrm{w}_s\}\,; w' : [\text{return } w, V]$. Entonces por la regla *MState* existen A', s' tales que:

  (7.1) $\Sigma \vdash \{w : C :: k; \mathrm{w}_s\}\,; \text{return } w : A'@w'$

  (7.2) $\Sigma; \cdot\,; \cdot \rhd V\{w'/w\} : A'@w' \mid s'$

  De (7.1) y por *C.Return*:

  (7.3) $\Sigma \vdash \{w : C; \mathrm{w}_s\}\,; k : A'@w$

  De (7.2) y por el principio de sustitución de mundos (Lema 1),

  (7.4) $\Sigma; \cdot\,; \cdot \rhd V\{w'/w\} : A'@w \mid s'$

  De (7.3) y (7.4) por *MState*

$$\Sigma \vdash \{w : C; \mathrm{w}_s\}\,; w : [k, V\{w'/w\}]$$

□

Un resultado inmediato de la propiedad de preservación de tipos es el siguiente corolario, que indica que no es posible generar una unidad móvil con un certificado que no se corresponda con su código:

Corolario 2.1. *Si* $\Sigma \vdash \mathrm{N}$ *es derivable y* $\mathrm{N} \longrightarrow^* \mathrm{W}\,; w : [k, box_s\, M]$*, entonces* $\Sigma; \cdot; \cdot \rhd M : A@w \mid s$*.*

4.3. Normalización fuerte

Un sistema de reescritura, en su forma más básica, consiste en un conjunto de términos y relaciones entre los términos que indican cómo transformar o reescribir los mismos. Un término se considera en forma normal si el mismo no puede ser transformado a otro término según las relaciones definidas. Un sistema de reescritura tiene la propiedad de normalización fuerte si, para cada término, todas las secuencias de reescritura a partir de ese término finalizan en algún momento en un término en forma normal. En otras palabras, no existen secuencias de reducción infinitas. Una propiedad más débil es justamente la de normalización débil, en la que solo se pide

que para cada término exista al menos una secuencia que finalice en un término en forma normal.

Para el sistema $\lambda_{\square}^{\mathsf{Cert}}$ que se está presentando se probará normalización fuerte respecto a la reducción de la máquina abstracta. Es decir, que cualquier secuencia de pasos de reducción de la máquina siempre finaliza en un estado terminal. Para poder demostrar esta propiedad se tomará como base el sistema de lambda cálculo simplemente tipado con el tipo unitario ($\lambda^{\mathbf{1},\rightarrow}$) el cual ya se sabe que cumple con la propiedad de normalización fuerte. Los estados de la máquina y pasos de reducción en $\lambda_{\square}^{\mathsf{Cert}}$ serán traducidos en términos y reducciones de $\lambda^{\mathbf{1},\rightarrow}$. Con dar una traducción precisa alcanzará, puesto que si existiese una reducción infinita en $\lambda_{\square}^{\mathsf{Cert}}$ al traducirla en una reducción en $\lambda^{\mathbf{1},\rightarrow}$ daría como resultado una reducción infinita en este último sistema, lo cual contradeciría la normalización fuerte de $\lambda^{\mathbf{1},\rightarrow}$.

Por razones técnicas que serán expuestas en breve, se considerará una modificación en la definición de la semántica de la máquina abstracta de $\lambda_{\square}^{\mathsf{Cert}}$ que consiste en reemplazar la regla

$$(2) \quad \mathbb{W}; w : [k \lhd \circ\ N, V] \quad \longrightarrow \quad \mathbb{W}; w : [k \lhd V\ \circ, N]$$

por las siguientes dos reglas

$$(2.1) \quad \mathbb{W}; w : [k \lhd \circ\ N, V] \quad \longrightarrow \quad \mathbb{W}; w : [k \lhd V\ \circ, N],\ N \text{ no es un valor}$$
$$(2.2) \quad \mathbb{W}; w : [k \lhd \circ\ V, \lambda a.M] \quad \longrightarrow \quad \mathbb{W}; w : [k, M\{a/V\}]$$

Estos dos últimos esquemas surgen de refinar el paso de reducción (2) a partir de inspeccionar su comportamiento en una secuencia de reducción infinita. Si N es un valor, entonces cada aplicación del paso (2) es seguida inmediatamente por una aplicación del paso (3). Si se juntan estos dos pasos en uno se obtiene precisamente la regla (2.2). Por otro lado, el esquema de reducción (2.2) equivale al esquema (2) cuando N no es un valor. Esta claro que cualquier secuencia de reducción infinita en la formulación original de la semántica de $\lambda_{\square}^{\mathsf{Cert}}$ puede ser imitada por una secuencia de reducción infinita en la semántica modificada de manera tal que cada paso de la regla (2)

- no es seguido por un paso de la regla (3) y por lo tanto se transforma en un paso de la regla (2.1)
- sí es seguido por un paso (3) por lo que el paso (2) seguido del paso (3) se transforma en un paso (2.2).

Luego, alcanza con probar la propiedad de normalización fuerte en el sistema modificado para deducir que la misma propiedad se cumple para la formulación original.

La prueba de normalización fuerte procede en dos partes. En primer lugar se relacionan las reducciones de la máquina abstracta con una nueva noción de reducción que opera directamente sobre términos de $\lambda_{\square}^{\mathsf{Cert}}$ a través de un mapeo $F(\cdot)$. Luego se relacionará esta última con la reducción en $\lambda^{\mathbf{1},\rightarrow}$ a través de un mapeo $T(\cdot)$. La siguiente figura sintetiza lo expuesto:

$$\text{Reducción de la máquina } (\lambda^{\mathsf{Cert}}_{\Box}) \xrightarrow[F(\cdot)]{} \text{Reducción lambda } (\lambda^{\mathsf{Cert}}_{\Box}) \xrightarrow[T(\cdot)]{} \text{Lambda cálculo tipado } (\lambda^{\mathbf{1},\to})$$

Antes de definir el mapeo $F(\cdot)$, es necesaria una definición auxiliar de $\overline{M}$ que consiste simplemente en sustituir cada ocurrencia de un mundo w por $\bullet$:

Definición 4.1.

$$\begin{array}{rcl}
\overline{a} & =_{def} & a \\
\overline{v^\bullet} & =_{def} & v^\bullet \\
\overline{M\,N} & =_{def} & \overline{M}\,\overline{N} \\
\overline{\lambda a.M} & =_{def} & \lambda a.\overline{M} \\
\overline{unpack\ M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N} & =_{def} & unpack\ \overline{M}\ to\ \langle v^\bullet, v^\circ\rangle\ in\ \overline{N} \\
\overline{box_s\, M} & =_{def} & box_s\, \overline{M} \\
\overline{fetch[w]\, M} & =_{def} & fetch[\bullet]\, \overline{M}
\end{array}$$

El mapeo $F(\cdot)$ consiste en transformar estados de la máquina en términos de $\lambda^{\mathsf{Cert}}_{\Box}$, reconstruyendo un término a partir de la información del contexto y reemplazando los mundos utilizando la definición anterior:

Definición 4.2 (Mapeo $F(\cdot)$)**.**

$$\begin{array}{rcl}
F(\mathbb{W}; w : [\text{finish}, M]) & =_{def} & \overline{M} \\
F(\mathbb{W}; w : [k \lhd \circ\ N, M]) & =_{def} & F(\mathbb{W}; w : [k, M\,N]) \\
F(\mathbb{W}; w : [k \lhd V\ \circ, N]) & =_{def} & F(\mathbb{W}; w : [k, V\,N]) \\
F(\mathbb{W}; w : [k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N, M]) & =_{def} & F(\mathbb{W}; w : [k, unpack\ M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N]) \\
F(\{w : C :: k; w_s\}; w' : [\text{return } w, M]) & =_{def} & F(\{w : C; w_s\}; w : [k, M])
\end{array}$$

Si el contexto es vacío (finish), $F(\cdot)$ simplemente sustituye los mundos del término foco de la computación. Sino, va reconstruyendo el término de manera inversa a los pasos de reducción. Notar que en el último item de la definición no se tiene en cuenta la sustitución que sí aparece en la regla semántica (7). Esto se debe a que en el resultado final solo aparecerá el mundo $\bullet$ por lo que no es necesario considerar mundos en particular.

El siguiente lema establece que el mapeo $F(\cdot)$ preserva el tipo:

Lema 3. *Sea $\mathbb{N}$ igual a $\mathbb{W}; w : [k, M]$. Si $\Sigma \vdash \mathbb{N}$ es derivable y $\Sigma \vdash \bullet$, entonces existen A y s tales que $\Sigma; \cdot\,; \cdot \rhd F(\mathbb{N}) : A@\bullet \mid s$ es derivable.*

Para poder relacionar la reducción de máquina abstracta en $\lambda^{\mathsf{Cert}}_{\Box}$ y la reducción en $\lambda^{\mathbf{1},\to}$ se introduce la noción reducción lambda para $\lambda^{\mathsf{Cert}}_{\Box}$.

Definición 4.3 (Reducción lambda para $\lambda^{\mathsf{Cert}}_{\Box}$)**.**

$$\begin{array}{rcl}
(\lambda a.M)\, N & \longrightarrow_{\beta} & M\{a/N\} \\
unpack\ box_s\, M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N & \longrightarrow_{\beta_\Box} & N\{v^\bullet/M\}\{v^\circ/s\} \\
fetch[w]\, M & \longrightarrow_{ftch} & M
\end{array}$$

Los dos primeros esquemas son estándar. El último establece que los términos fetch no tienen efecto computacional a nivel de lambda términos. Se debe mencionar en este punto que se considera reducción bajo todos los constructores de término.

A partir de esta definición se puede establecer el siguiente lema, que dice que el mapeo $F(\cdot)$ preserva la reducción:

Lema 4. *Si* $\mathbb{N} \longrightarrow_{1,2.1,4,7} \mathbb{N}'$*, entonces* $F(\mathbb{N}) = F(\mathbb{N}')$.
Si $\mathbb{N} \longrightarrow_{2.2,3,5,6} \mathbb{N}'$*, entonces* $F(\mathbb{N}) \longrightarrow_{\beta,\beta_\Box,ftch} F(\mathbb{N}')$.

Notar que el mapeo $F(\cdot)$ se mantiene invariante en los pasos de reducción de la máquina que solo involucran el manejo del contexto, mientras que cambia en los pasos de reducción que involucran una sustitución o cambio de mundo.

La segunda parte de la prueba consiste en relacionar la reducción lambda en $\lambda_\Box^{\mathsf{Cert}}$ con la reducción $\lambda^{\mathbf{1},\rightarrow}$. Con tal propósito se introduce el mapeo $T(\cdot)$. que consta de dos partes. Por un lado, un mapeo de los tipos de $\lambda_\Box^{\mathsf{Cert}}$ con los tipos de $\lambda^{\mathbf{1},\rightarrow}$ y, por otro lado, un mapeo de los términos de $\lambda_\Box^{\mathsf{Cert}}$ con los términos de $\lambda^{\mathbf{1},\rightarrow}$.

Definición 4.4 (Mapeo $T(\cdot)$)**.**

$$\begin{aligned} T(P) &=_{def} P \\ T(A \supset B) &=_{def} T(A) \supset T(B) \\ T([s]A) &=_{def} \mathbf{1} \supset T(A) \end{aligned}$$

$$\begin{aligned} T(a) &=_{def} a \\ T(v^\bullet) &=_{def} v\ unit \\ T(\lambda a.M) &=_{def} \lambda a.T(M) \\ T(M\,N) &=_{def} T(M)\ T(N) \\ T(box_s\, M) &=_{def} \lambda a.T(M), a\ fresh\ of\ type\ \mathbf{1} \\ T(unpack\ M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N) &=_{def} (\lambda v.T(N))\ T(M) \\ T(fetch[w]\, M) &=_{def} (\lambda a.a)\ T(M) \end{aligned}$$

En $\lambda^{\mathbf{1},\rightarrow}$, el tipo $\mathbf{1}$ representa al (único) valor *unit*. Respecto al mapeo de tipos, el tipo función se mantiene, mientras que el tipo modal $[s]A$ se traduce en un tipo función cuyo dominio es el tipo unidad $\mathbf{1}$ y cuyo codominio es la traducción de A. La traducción de los términos surge de la traducción de los tipos. El término $box_s\, M$ representa código móvil, por lo que se traduce a una función que toma una variable fresca de tipo $\mathbf{1}$ y retorna la traducción M. Es decir que el código móvil se decora con una función que solamente necesita el valor *unit* para retornar el código original. Esto se ve reflejado en la traducción de $v^\bullet$ en donde el código móvil representado por la variable v se obtiene aplicando *unit*. En el término *unpack* M *to* $\langle v^\bullet, v^\circ\rangle$ *in* N el código móvil v se abstrae en la traducción del cuerpo N y se le aplica la traducción de M. El caso $fetch$ garantiza que cada paso $\longrightarrow_{ftch}$ se mapea con un paso no vacío en $\lambda^{\mathbf{1},\rightarrow}$.

El siguiente lema establece que el mapeo $T(\cdot)$ preserva los tipos:

Lema 5. *Si* $\Sigma; \Delta; \Gamma \rhd M : A@w \,|\, s$ *es derivable en* $\lambda_\Box^{\mathsf{Cert}}$*, entonces* $\Delta', \Gamma' \rhd T(M) : T(A)$ *es derivable en* $\lambda^{\mathbf{1},\rightarrow}$*, donde*

1. Γ' *resulta de reemplazar cada hipótesis de la forma* $a : A@w$ *por* $a : T(A)$ *y*

2. Δ' *resulta de reemplazar cada hipótesis de la forma* $v : A@w$ *por* $v : \mathbf{1} \supset T(A)$.

Además de preservar los tipos, el mapeo $T(\cdot)$ preserva también la reducción. Para demostrarlo, hacen falta demostrar dos lemas previos. El primero de ellos establece que $T(\cdot)$ conmuta con la sustitución de variables locales:

Lema 6. $T(M)\{a/T(N)\} = T(M\{a/N\})$.

El segundo lema auxiliar que se necesita involucra variables de validez. A diferencia de las variables locales, $T(\cdot)$ no conmuta con la sustitución de variables de validez. Es decir que $T(M)\{v/T(N)\} \neq T(M\{v/N\})$ (basta tomar el caso de $M = v^\bullet$ para verificarlo). Sin embargo, si se cumple el siguiente lema el cual alcanza para el propósito de demostrar que $T(\cdot)$ preserva la reducción:

Lema 7. $T(M)\{v/\lambda a.T(N)\} \longrightarrow_\beta^* T(M\{v^\bullet/N\}\{v^\circ/s\})$.

La relación $\longrightarrow_\beta^*$ denota 0, 1 o más pasos de reducción $\longrightarrow_\beta$, es decir la clausura reflexiva y transitiva de $\longrightarrow_\beta$.

Ahora sí se está en condiciones de formular el lema que establece que $T(\cdot)$ preserva la reducción:

Lema 8. *If* $M \longrightarrow_{\beta,\beta_\square,ftch} N$, *then* $T(M) \longrightarrow_\beta^+ T(N)$.

La relación $\longrightarrow_\beta^+$ denota 1 o más pasos de reducción $\longrightarrow_\beta$, es decir la clausura reflexiva de $\longrightarrow_\beta$.

Antes de demostrar que la relación $\longrightarrow$ tiene la propiedad de normalización fuerte se demostrará primero que la relación $\longrightarrow_{1,2.1,4,7}$ tiene en sí misma la propiedad de normalización fuerte. La prueba de este resultado es la que motivó la modificación de la semántica de reducción presentada al principio de la sección. Para ello se hará uso del siguiente resultado ya conocido que sirve para demostrar normalización fuerte sobre la combinación de relaciones binarias:

Lema 9. *Sean* $\longrightarrow_1$ *y* $\longrightarrow_2$ *relaciones binarias sobre algún conjunto* X. *Supongamos que*

1. $\longrightarrow_1$ *tiene la propiedad de normalización fuerte y*

2. $\mathcal{M}$ *es un mapeo de* X *a algún conjunto bien fundado tal que*

   a) $x \longrightarrow_1 y$ *implica* $\mathcal{M}(x) = \mathcal{M}(y)$

   b) $x \longrightarrow_2 y$ *implica* $\mathcal{M}(x) > \mathcal{M}(y)$

Entonces $\longrightarrow_1 \cup \longrightarrow_2$ *tiene la propiedad de normalización fuerte.*

Antes de poder utilizar este lema para la prueba de normalización fuerte de $\longrightarrow_{1,2.1,4,7}$, son necesarias algunas definiciones auxiliares. El tamaño de un término M, denotado por $|M|$, se define como el número de variables y constructores en M:

Definición 4.5 (Tamaño de un término M)**.**

$$\begin{array}{rcl} |a| & =_{def} & 1 \\ |v^\bullet| & =_{def} & 1 \\ |box_s\, M| & =_{def} & |M| + 1 \\ |\lambda a.M| & =_{def} & |M| + 1 \\ |M\, N| & =_{def} & |M| + |N| + 1 \\ |unpack\ M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N| & =_{def} & |M| + |N| + 1 \\ |fetch[w]\, M| & =_{def} & |M| + 1 \end{array}$$

Notar que $|M\{w'/w\}| = |M|$. El tamaño de un contexto k, denotado $|k|$, se define tomando la suma de los tamaños de los términos con agujeros, donde cada agujero cuenta como 1:

Definición 4.6 (Tamaño de un contexto k)**.**

$$\begin{array}{rcl} |\text{return } w| & =_{def} & 1 \\ |\text{finish}| & =_{def} & 1 \\ |k \triangleleft l| & =_{def} & |k| + |l| \\ & & \\ |\circ\ N| & =_{def} & |N| + 1 \\ |V\ \circ| & =_{def} & |V| + 1 \\ |unpack\ \circ\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N| & =_{def} & |N| + 1 \end{array}$$

Se usará la notación $|k, M|$ para abreviar $|k| + |M|$.

Lema 10. *La relación de reducción $\longrightarrow_{1,2.1,4,7}$ es fuertemente normalizante.*

Demostración. En primer lugar se prueba la normalización fuerte para los esquemas (1) y (4). Luego se concluye el resultado esperado utilizando el lema Lem. 9, introduciendo un mapeo $\mathcal{M}_2$ tal que:

1. $\mathbb{N} \longrightarrow_{1,4} \mathbb{N}'$ implica $\mathcal{M}_2(\mathbb{N}) = \mathcal{M}_2(\mathbb{N}')$ y
2. $\mathbb{N} \longrightarrow_{2.1,7} \mathbb{N}'$ implica $\mathcal{M}_2(\mathbb{N}) > \mathcal{M}_2(\mathbb{N}')$.

La normalización fuerte de los esquemas (1) y (4) se comprueba a partir del hecho de que el siguiente mapeo $\mathcal{M}_1$ de estados de máquina sobre pares de números naturales (ordenados lexicográficamente) decrece estrictamente cuando se aplican los esquemas (1) y (4):

$$\mathcal{M}_1(\mathbb{W}; w : [k, M]) =_{def} \langle |\mathbb{W}|, |M| \rangle$$

Notar que también decrece cuando se aplica el esquema (7). Sin embargo, no decrece cuando se aplica el esquema (2).

El mapeo $\mathcal{M}_2$ se define de la siguiente manera:

$$\mathcal{M}_2(\mathbb{W}; w : [k, M]) =_{def} \langle |\mathbb{W}|, |k, M| - len(k) - m(M) \rangle$$

donde $len(k)$ es simplemente la longitud de k, mientras que m es el siguiente mapeo desde términos cerrados a enteros positivos:

$$\begin{array}{rcl} m(V) & =_{def} & 0 \\ m(M\ N) & =_{def} & 1 + m(M) \\ m(unpack\ M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N) & =_{def} & 1 + m(M) \\ m(fetch[w]\ M) & =_{def} & 1 \end{array}$$

Este mapeo decrece estrictamente para los esquemas (2.1) y (7), mientras que se mantiene igual para los esquemas (1) y (4).

- Caso (1)

$$\begin{array}{rcl} \mathcal{M}_2(\mathbb{W}; w : [k, M\ N]) & = & \langle |\mathbb{W}|, |k, M\ N| - len(k) - m(M\ N)\rangle \\ & = & \langle |\mathbb{W}|, |k, M\ N| - len(k) - 1 - m(M)\rangle \\ & = & \langle |\mathbb{W}|, |k \triangleleft \circ\ N, M| - len(k) - 1 - m(M)\rangle \\ & = & \mathcal{M}_2(\mathbb{W}; w : [k \triangleleft \circ\ N, M]) \end{array}$$

- Caso (2.1), recordar de la definición del esquema que N no es un valor. Por lo tanto, puede ser una aplicación, un término unpack o un término fetch. Notar que para cada uno de estos casos se cumple que $m(N) > 0$. Luego, se aplica el siguiente razonamiento:

$$\begin{array}{rcl} \mathcal{M}_2(\mathbb{W}; w : [k \triangleleft \circ\ N, V]) & = & \langle |\mathbb{W}|, |k \triangleleft \circ\ N, V| - len(k) - 1 - m(V)\rangle \\ & = & \langle |\mathbb{W}|, |k, N, V| + 1 - len(k) - 1\rangle \\ & > & \langle |\mathbb{W}|, |k, V, N| + 1 - len(k) - 1 - m(N)\rangle \\ & = & \langle |\mathbb{W}|, |k \triangleleft V\ \circ, N| - len(k) - 1 - m(N)\rangle \\ & = & \mathcal{M}_2(\mathbb{W}; w : [k \triangleleft V\ \circ, N]) \end{array}$$

- Caso (4)

$$\begin{array}{rcl} & & \mathcal{M}_2(\mathbb{W}; w : [k, unpack\ M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N]) \\ & = & \langle |\mathbb{W}|, |k, unpack\ M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N| - len(k) - m(unpack\ M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N)\rangle \\ & = & \langle |\mathbb{W}|, |k, unpack\ M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N| - len(k) - 1 - m(M)\rangle \\ & = & \langle |\mathbb{W}|, |k \triangleleft unpack\ \circ\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N, M| - len(k) - 1 - m(M)\rangle \\ & = & \mathcal{M}_2(\mathbb{W}; w : [k \triangleleft unpack\ \circ\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N, M]) \end{array}$$

- Caso (7). Sea $n = |\{w : C :: k; w_s\}|$.

$$\begin{array}{rcl} & & \mathcal{M}_2(\{w : C :: k; w_s\}; w' : [\text{return } w, V]) \\ & = & \langle n, |\text{return } w| + |V| - len(\text{return } w) - m(V)\rangle \\ & = & \langle n, 1 + |V| - 1 - m(V)\rangle \\ & = & \langle n, |V|\rangle \\ & > & \langle n - 1, |k, V| - len(k)|\rangle \\ & = & \langle n - 1, |k, V| - len(k) - m(V)|\rangle \\ & = & \mathcal{M}_2(\{w : C; w_s\}; w : [k, V]) \end{array}$$

□

Hasta aquí, se han introducido los mapeos $F(\cdot)$ y $T(\cdot)$ y se ha demostrado que ambos conservan el tipado y la reducción. Además, se ha demostrado que la relación de reducción $\longrightarrow_{1,2.1,4,7}$ tiene la propiedad de normalización fuerte. Con todos estos resultados se puede demostrar el resultado deseado de que la relación $\longrightarrow$ tiene la propiedad de normalización fuerte.

Proposición 6. *La relación de reducción $\longrightarrow$ es fuertemente normalizante.*

Demostración. La prueba se realiza por el método del absurdo. Supongamos que existe una secuencia de reducción infinita que comienza en un estado de la máquina $\mathbb{N}_1$. Como $\longrightarrow_{1,2.1,4,7}$ es fuertemente normalizante, esta secuencia debe tener un número infinito de pasos de reducción $\longrightarrow_{2.2,3,5}$ intercalados:

$$\mathbb{N}_1 \longrightarrow^*_{1,2.1,4,7} \mathbb{N}_2 \longrightarrow_{2.2,3,5} \mathbb{N}_3 \longrightarrow^*_{1,2.1,4,7} \mathbb{N}_4 \longrightarrow_{2.2,3,5} \mathbb{N}_5 \longrightarrow^*_{1,2.1,4,7} \mathbb{N}_6 \longrightarrow_{2.2,3,5} \dots$$

Luego por el lema Lem. 4, se tiene la siguiente secuencia de reducción lambda sobre términos tipados (Lem. 3):

$$F(\mathbb{N}_1) = F(\mathbb{N}_2) \longrightarrow_{\beta,\beta_\square,ftch} F(\mathbb{N}_3) = F(\mathbb{N}_4) \longrightarrow_{\beta,\beta_\square,ftch} F(\mathbb{N}_5) = F(\mathbb{N}_6) \longrightarrow_{\beta,\beta_\square,ftch} \dots$$

Finalmente, por el Lem. 8 se obtiene la siguiente secuencia de reducción infinita de términos tipables (Lem. 5) en $\lambda^{\mathbf{1},\rightarrow}$, lo cual contradice la normalización fuerte de$\lambda^{\mathbf{1},\rightarrow}$:

$$T(F(\mathbb{N}_1)) = T(F(\mathbb{N}_2)) \longrightarrow^+_\beta T(F(\mathbb{N}_3)) = T(F(\mathbb{N}_4)) \longrightarrow^+_\beta T(F(\mathbb{N}_5)) = T(F(\mathbb{N}_6)) \longrightarrow^+_\beta \dots$$

□

Capítulo 5

Extensiones

En este capítulo se introducen una serie de extensiones para $\lambda_{\square}^{\mathsf{Cert}}$ las cuales permiten darle mayor expresividad al lenguaje. En primer lugar se explica cómo introducir valores booleanos al cálculo. Luego se introduce el manejo de números naturales al lenguaje. Finalmente, se analiza la necesidad de introducir la operación suma (+) o concatenación para los certificados.

Para cada una de las extensiones presentadas se explicarán los cambios necesarios en la sintaxis del lenguaje, en el sistema de tipos y en la semántica de la máquina abstracta. Si bien todas estas extensiones requieren modificaciones en las propiedades ya demostradas, las mismas no son presentadas en el trabajo ya que no aportan nada respecto a lo ya visto.

5.1. Booleanos

Se introduce el uso de valores booleanos al lenguaje $\lambda_{\square}^{\mathsf{Cert}}$ de manera tradicional, definiendo dos constantes que representen los valores de verdad posibles (verdadero y falso), introduciendo una serie de operadores booleanos que representen a la conjunción, disyunción y negación e incluyendo una sentencia del tipo if $-$ then $-$ else.

5.1.1. Sintaxis

La definición de tipos se extiende agregando un nuevo valor denominado $Bool$. Además, en la definición de valores V, se incluyen dos nuevas constantes: **true** y **false**. Los operadores lógicos junto a la sentencia if $-$ then $-$ else se incorporan a la definición de términos. Finalmente, se agregan en la definición de certificados nuevas construcciones necesarias en el sistema de tipos. Un resumen de las extensiones es el siguiente:

Definición 5.1.

tipos	A	::=	$\ldots \mid Bool$
certificados	s, t	::=	$\ldots \mid true \mid false \mid and(s,t) \mid or(s,t) \mid not(s) \mid if(s,t,r)$
valores	V	::=	$\ldots \mid$ **true** $\mid$ **false**
términos	M, N, P	::=	$\ldots \mid M \&\& N \mid M \| N \mid \sim M \mid$ if P then M else N

5.1.2. Sistema de tipos

A la definición original del sistema de tipos, se le agregan las siguientes reglas que permiten tipar expresiones booleanas:

Definición 5.2.

$$\frac{}{\Sigma;\Delta;\Gamma \rhd \textbf{true} : Bool@w \,|\, true}\ True \qquad \frac{}{\Sigma;\Delta;\Gamma \rhd \textbf{false} : Bool@w \,|\, false}\ False$$

$$\frac{\Sigma;\Delta;\Gamma \rhd M : Bool@w \,|\, s \quad \Sigma;\Delta;\Gamma \rhd N : Bool@w \,|\, t}{\Sigma;\Delta;\Gamma \rhd M \&\& N : Bool@w \,|\, and(s,t)}\ And$$

$$\frac{\Sigma;\Delta;\Gamma \rhd M : Bool@w \,|\, s \quad \Sigma;\Delta;\Gamma \rhd N : Bool@w \,|\, t}{\Sigma;\Delta;\Gamma \rhd M \| N : Bool@w \,|\, or(s,t)}\ Or$$

$$\frac{\Sigma;\Delta;\Gamma \rhd M : Bool@w \,|\, s}{\Sigma;\Delta;\Gamma \rhd \sim M : Bool@w \,|\, not(s)}\ Not$$

$$\frac{\Sigma;\Delta;\Gamma \rhd P : Bool@w \,|\, r \quad \Sigma;\Delta;\Gamma \rhd M : A@w \,|\, s \quad \Sigma;\Delta;\Gamma \rhd N : A@w \,|\, t}{\Sigma;\Delta;\Gamma \rhd \text{if } P \text{ then } M \text{ else } N : A@w \,|\, if(r,s,t)}\ If$$

Las reglas *True* y *False* indican que las constantes booleanas siempre tienen tipo *Bool*. La regla *And* establece que si se tienen dos términos M y N que tienen tipo *Bool*, entonces la expresión $M\&\&N$ también se puede tipar como *Bool* con un certificado que combina los certificados originales de las pruebas sobre M y N. Las reglas *Or* y *Not* son análogas. Finalmente, la regla *If* indica bajo que condiciones se puede tipar un término de la forma if P then M else N: si P tiene tipo *Bool* y M y N tienen el mismo tipo A, entonces toda la expresión tiene tipo A. El certificado final es una combinación de los certificados utilizados para tipar P, M y N.

5.1.3. Semántica operacional

La máquina abstracta debe ser modificada para incluir la semántica de los nuevos operadores booleanos. Para ello en primer lugar se debe extender la sintaxis de la máquina. Más específicamente, se debe modificar la definición de capas de la siguiente manera:

Definición 5.3.

$$l \ ::= \ \ldots \,|\, \text{if } \circ \text{ then } M \text{ else } N \,|\, \circ \&\& N \,|\, \circ \| N \,|\, \sim \circ$$

El siguiente paso es agregar algunos pasos de reducción:

Definición 5.4.

$$
\begin{array}{lrcl}
(8) & \mathbb{W}; w : [k, \text{if } P \text{ then } M \text{ else } N] & \longrightarrow & \mathbb{W}; w : [k \lhd \text{if } \circ \text{ then } M \text{ else } N, P] \\
(9) & \mathbb{W}; w : [k \lhd \text{if } \circ \text{ then } M \text{ else } N, \mathbf{true}] & \longrightarrow & \mathbb{W}; w : [k, M] \\
(10) & \mathbb{W}; w : [k \lhd \text{if } \circ \text{ then } M \text{ else } N, \mathbf{false}] & \longrightarrow & \mathbb{W}; w : [k, N] \\
(11) & \mathbb{W}; w : [k, M \&\& N] & \longrightarrow & \mathbb{W}; w : [k \lhd \circ \&\& N, M] \\
(12) & \mathbb{W}; w : [k \lhd \circ \&\& N, \mathbf{false}] & \longrightarrow & \mathbb{W}; w : [k, \mathbf{false}] \\
(13) & \mathbb{W}; w : [k \lhd \circ \&\& N, \mathbf{true}] & \longrightarrow & \mathbb{W}; w : [k, N] \\
(14) & \mathbb{W}; w : [k, M \| N] & \longrightarrow & \mathbb{W}; w : [k \lhd \circ \| N, M] \\
(15) & \mathbb{W}; w : [k \lhd \circ \| N, \mathbf{true}] & \longrightarrow & \mathbb{W}; w : [k, \mathbf{true}] \\
(16) & \mathbb{W}; w : [k \lhd \circ \| N, \mathbf{false}] & \longrightarrow & \mathbb{W}; w : [k, N] \\
(17) & \mathbb{W}; w : [k, \sim M] & \longrightarrow & \mathbb{W}; w : [k \lhd \sim \circ, M] \\
(18) & \mathbb{W}; w : [k \lhd \sim \circ, \mathbf{true}] & \longrightarrow & \mathbb{W}; w : [k, \mathbf{false}] \\
(19) & \mathbb{W}; w : [k \lhd \sim \circ, \mathbf{false}] & \longrightarrow & \mathbb{W}; w : [k, \mathbf{true}]
\end{array}
$$

Las reglas (8) al (10) le dan semántica a una expresión de la forma if P then M else N: la regla (8) establece que primero se debe evaluar P; si esta evaluación da verdadero, la regla (9) indica que se debe seguir evaluando M, en caso contrario, la regla (10) establece que se evalúe el término N. Las reglas (11) al (13) indican la manera de evaluar una conjunción $M \&\& M$. En primer lugar, según la regla (11), se evalúa M. En caso de que la evaluación de falso, la regla (12) indica que el resultado de toda la expresión también es falso. Sino, la regla (13) establece que el resultado de la expresión es el resultado de la evaluación de N. De manera similar, las reglas (15) a (17) y las reglas (18) y (19) indican como evaluar la disyunción y la negación respectivamente.

Para que la definición quede completa, es necesario escribir nuevas reglas de tipado para los estados de la máquina relativos a los nuevos constructores de capas introducidos:

Definición 5.5.

$$\frac{\Sigma \vdash \mathbb{W}\,; k : A@w \quad \Sigma; \cdot\,; \cdot \rhd M : A@w \,|\, s \quad \Sigma; \cdot\,; \cdot \rhd N : A@w \,|\, t}{\Sigma \vdash \mathbb{W}\,; k \lhd \text{if } \circ \text{ then } M \text{ else } N : Bool@w}\ C.If$$

$$\frac{\Sigma \vdash \mathbb{W}\,; k : Bool@w \quad \Sigma; \cdot\,; \cdot \rhd M : Bool@w \,|\, s}{\Sigma \vdash \mathbb{W}\,; k \lhd \circ \&\& M : Bool@w}\ C.And$$

$$\frac{\Sigma \vdash \mathbb{W}\,; k : Bool@w \quad \Sigma; \cdot\,; \cdot \rhd M : Bool@w \,|\, s}{\Sigma \vdash \mathbb{W}\,; k \lhd \circ \| M : Bool@w}\ C.Or$$

$$\frac{\Sigma \vdash \mathbb{W}\,; k : Bool@w \quad \Sigma; \cdot\,; \cdot \rhd M : Bool@w \,|\, s}{\Sigma \vdash \mathbb{W}\,; k \lhd \sim \circ : Bool@w}\ C.Not$$

5.2. Números naturales

El lenguaje $\lambda_{\square}^{\mathsf{Cert}}$ se extiende para que incluya los números naturales siguiendo su definición teórica:

- La constante *zero* es un número natural

- Para todo número natural n, $succ(n)$ es un número natural.

  Además, se incluyen las operaciones aritméticas de suma y multiplicación junto a una sentencia de selección o case.

5.2.1. Sintaxis

En primer lugar se modifica la definición de tipos para incluir un nuevo valor Nat. Luego se extiende la definición de valores V para incluir a todos los los números naturales : **z**, **sz**, **szz**, ... La razón por la cual se incluyen como valores y no términos queda evidenciada en la definición de las reglas semánticas. Adicionalmente, a la definición de los términos se agrega la construcción $\mathbf{s}M$, que corresponde al sucesor de una expresión natural M, el constructor de selección case y las construcciones para las operaciones aritméticas + y $*$. Finalmente, se modifica la definición de certificados para incluir las construcciones necesarias en las nuevas reglas de tipado. Las nuevas extensiones se resumen a continuación:

Definición 5.6.

$$
\begin{array}{lrcl}
\textit{tipos} & A & ::= & \ldots \mid Nat \\
\textit{certificados} & s,t & ::= & \ldots \mid zero \mid succ(s) \mid add(s,t) \mid mul(s,t) \\
\textit{valores} & V & ::= & \ldots \mid \mathbf{s}^{(n)}\mathbf{z} \ (n \geq 0) \\
\textit{términos} & M,N,P,Q & ::= & \ldots \mid \text{case } M \text{ of } \mathbf{z} \rightarrow P \boxplus \mathbf{s}a \rightarrow Q \mid \mathbf{s}M \mid M + N \mid M * N
\end{array}
$$

5.2.2. Sistema de tipos

Es necesario extender el sistema de tipos con cinco reglas adicionales:

Definición 5.7.

$$\frac{}{\Sigma;\Delta;\Gamma \rhd \mathbf{z} : Nat@w \mid zero}\ Zero \qquad \frac{\Sigma;\Delta;\Gamma \rhd M : Nat@w \mid s}{\Sigma;\Delta;\Gamma \rhd \mathbf{s}M : Nat@w \mid succ(s)}\ Succ$$

$$\frac{\Sigma;\Delta;\Gamma \rhd M : Nat@w \mid s \quad \Sigma;\Delta;\Gamma \rhd N : Nat@w \mid t}{\Sigma;\Delta;\Gamma \rhd M + N : Nat@w \mid add(s,t)}\ Add$$

$$\frac{\Sigma;\Delta;\Gamma \rhd M : Nat@w \mid s \quad \Sigma;\Delta;\Gamma \rhd N : Nat@w \mid t}{\Sigma;\Delta;\Gamma \rhd M * N : Nat@w \mid mul(s,t)}\ Mul$$

$$\frac{\Sigma;\Delta;\Gamma \rhd M : Nat@w \mid s \quad \Sigma;\Delta;\Gamma \rhd P : A@w \mid t \quad \Sigma;\Delta;\Gamma, a : Nat@w \rhd Q : A@w \mid r}{\Sigma;\Delta;\Gamma \rhd \text{case } M \text{ of } \mathbf{z} \rightarrow P \boxplus \mathbf{s}a \rightarrow Q : A@w \mid case(s,t,r)}\ Case$$

El esquema *Zero* establece que siempre es posible tipar el valor **z** como un número natural. La regla *Succ* refleja el hecho de que si M es un valor que representa un número natural, entonces $\mathbf{s}M$ también representa un número natural. El esquema *Add* determina que si se tienen dos términos que representan números naturales,

entonces la suma entre ellos también representa un número natural. El esquema *Mul* es análogo al anterior para la multiplicación. Finalmente la regla *Case* permite tipar una expresión case M of $\mathbf{z}{\rightarrow}P \boxplus \mathbf{s}a \rightarrow Q$: si M tiene tipo Nat, P tiene tipo A y Q también tiene tipo A con la hipótesis de que x tiene tipo Nat, entonces toda la expresión tiene tipo A. La aplicación de todos estos esquemas se ve reflejada en sus correspondientes certificados.

5.2.3. Semántica operacional

La máquina abstracta debe ser extendida para poder reducir un término M bajo un constructor $\mathbf{s}$ y para reflejar la semántica de los nuevos operadores aritméticos de la suma y la multiplicación. La definición de capas en la sintaxis de la máquina se extiende de la siguiente manera:

Definición 5.8.

$$l \;::=\; \ldots \mid \mathbf{s} \circ \mid M \;+\; \circ \mid M \;*\; \circ \mid \text{case } \circ \text{ of } \mathbf{z}{\rightarrow}P \boxplus \mathbf{s}a \rightarrow Q$$

Los nuevos pasos de reducción necesarios son los siguientes:

Definición 5.9.

$$\begin{array}{lrcl}
(20) & \mathbb{W}; w : [k, \mathbf{s}M] & \longrightarrow & \mathbb{W}; w : [k \lhd \mathbf{s}\circ\, , M] \\
(21) & \mathbb{W}; w : [k \lhd \mathbf{s}\circ\, , V] & \longrightarrow & \mathbb{W}; w : [k, \mathbf{s}V] \\
(22) & \mathbb{W}; w : [k, M + N] & \longrightarrow & \mathbb{W}; w : [k \lhd M + \circ\, , N] \\
(23) & \mathbb{W}; w : [k \lhd M + \circ\, , \mathbf{z}] & \longrightarrow & \mathbb{W}; w : [k, M] \\
(24) & \mathbb{W}; w : [k \lhd M + \circ\, , \mathbf{s}N] & \longrightarrow & \mathbb{W}; w : [k, \mathbf{s}(M + N)] \\
(25) & \mathbb{W}; w : [k, M * N] & \longrightarrow & \mathbb{W}; w : [k \lhd M * \circ\, , N] \\
(26) & \mathbb{W}; w : [k \lhd M * \circ\, , \mathbf{z}] & \longrightarrow & \mathbb{W}; w : [k, \mathbf{z}] \\
(27) & \mathbb{W}; w : [k \lhd M * \circ\, , \mathbf{s}N] & \longrightarrow & \mathbb{W}; w : [k \lhd, M + (M * N)] \\
(28) & \mathbb{W}; w : [k, \text{case } M \text{ of } \mathbf{z}{\rightarrow}P \boxplus \mathbf{s}a \rightarrow Q] & \longrightarrow & \mathbb{W}; w : [k \lhd \text{case } \circ \text{ of } \mathbf{z}{\rightarrow}P \boxplus \mathbf{s}a \rightarrow Q, M] \\
(29) & \mathbb{W}; w : [k \lhd \text{case } \circ \text{ of } \mathbf{z}{\rightarrow}P \boxplus \mathbf{s}a \rightarrow Q, \mathbf{z}] & \longrightarrow & \mathbb{W}; w : [k, P] \\
(30) & \mathbb{W}; w : [k \lhd \text{case } \circ \text{ of } \mathbf{z}{\rightarrow}P \boxplus \mathbf{s}a \rightarrow Q, \mathbf{s}N] & \longrightarrow & \mathbb{W}; w : [k \lhd, Q\{a/N\}]
\end{array}$$

Las reglas (20) y (21) permiten reducir un término M bajo un constructor $\mathbf{s}$. Las reglas (22) a (24) permiten evaluar una suma $M + N$. En primer lugar se evalúa N y se espera que este término evalúe a $\mathbf{z}$ o $\mathbf{s}N'$. En el primer caso el resultado de la suma es M. En otro caso, el resultado es el sucesor de $M + N'$. Las reglas (25) a (27) permiten evaluar de manera similar una multiplicación $M * N$. Primero se evalúa N: si el resultado es $\mathbf{z}$ entonces toda la expresión evalúa a $\mathbf{z}$, y si el resultado es $\mathbf{s}N'$ la expresión evalúa a $M + \mathbf{s}(M * N')$. Las reglas (28) a (30) permiten evaluar un término de la forma case M of $\mathbf{z}{\rightarrow}P \boxplus \mathbf{s}a \rightarrow Q$. En primer lugar se evalúa el término M. Si el resultado es $\mathbf{z}$, se continúa evaluando P. Sino, el resultado tiene que ser de la forma $\mathbf{s}N$, en cuyo caso se continúa evaluando el resultado de sustituir a por N en Q.

Finalmente, las nuevas reglas de tipado para los estados de la máquina son las siguientes:

Definición 5.10.

$$\frac{\Sigma \vdash \mathbb{W}\,; k : Nat@w}{\Sigma \vdash \mathbb{W}\,; k \lhd \mathbf{s}\circ : Nat@w}\; C.Succ$$

$$\frac{\Sigma \vdash \mathbb{W}\,; k : Nat@w \quad \Sigma;\cdot\,;\cdot \rhd M : Nat@w \,|\, s}{\Sigma \vdash \mathbb{W}\,; k \lhd M \;+\; \circ : Nat@w}\; C.Add$$

$$\frac{\Sigma \vdash \mathbb{W}\,; k : Nat@w \quad \Sigma;\cdot\,;\cdot \rhd M : Nat@w \,|\, s}{\Sigma \vdash \mathbb{W}\,; k \lhd M \;*\; \circ : Nat@w}\; C.Mul$$

$$\frac{\Sigma \vdash \mathbb{W}\,; k : A@w \quad \Sigma;\cdot\,;\cdot \rhd P : A@w \,|\, s \quad \Sigma;\cdot\,; a : Nat@w \rhd Q : A@w \,|\, t}{\Sigma \vdash \mathbb{W}\,; k \lhd \mathrm{case}\;\circ\;\mathrm{of}\;\mathbf{z}{\rightarrow}P \boxplus \mathbf{s}a \rightarrow Q : Nat@w}\; C.Case$$

5.3. Concatenación de certificados

En esta sección se analiza la introducción del operador suma ($+$) de certificados. La motivación para introducir este operador, que se encuentra en la definición original de la lógica LP, proviene de dos de los nuevos constructores introducidos en este capítulo: case y if $\cdot$ then $\cdot$ else. Para ilustrar esto, supongamos que se tienen tres términos P, M y N con los siguientes tipos:

Ejemplo 5.1.

$$\begin{array}{l}\Sigma;\cdot\,;\cdot \rhd P : Bool@w \,|\, r\\ \Sigma;\cdot\,;\cdot \rhd M : A@w \,|\, s\\ \Sigma;\cdot\,;\cdot \rhd N : A@w \,|\, t\end{array}$$

Claramente, si se aplica la regla *If*, el término "if P then M else N"puede tiparse como A. Ahora bien, supongamos que se consideran las versiones móviles de M yN aplicando la regla $\Box I$:

Ejemplo 5.2.

$$\begin{array}{l}\Sigma;\cdot\,;\cdot \rhd box_s\, M : [s]A@w \,|\, !s\\ \Sigma;\cdot\,;\cdot \rhd box_t\, N : [t]A@w \,|\, !t\end{array}$$

Como $[s]A \neq [t]A$ (se asume $s \neq t$), la expresión "if P then $box_t\, M$ else $box_t\, N$"no puede ser tipada. Sin embargo, esto sería deseable, ya que ambas ramas del if tienen esencialmente el mismo tipo (código móvil que computa un valor de tipo A) con la diferencia que utilizan diferentes certificados. La solución a este problema consiste en introducir la operación de concatenación de certificados $s{+}t$ junto a las siguientes reglas de tipado:

Definición 5.11.

$$\frac{\Sigma;\Delta;\Gamma \rhd M : A@w \,|\, s}{\Sigma;\Delta;\Gamma \rhd M : A@w \,|\, s+t}\; CertSum1 \qquad \frac{\Sigma;\Delta;\Gamma \rhd M : A@w \,|\, t}{\Sigma;\Delta;\Gamma \rhd M : A@w \,|\, s+t}\; CertSum2$$

Ahora es posible tipar la expresión de más arriba de la siguiente manera:

Ejemplo 5.3.

$$\dfrac{\dfrac{\dfrac{\Sigma;\cdot;\cdot \rhd M : A@w \,|\, s}{\Sigma;\cdot;\cdot \rhd M : A@w \,|\, s+t}\,\mathit{CertSum1}}{\Sigma;\cdot;\cdot \rhd \mathit{box}_{s+t}\, M : [s+t]A@w \,|\, !(s+t)}\,\Box I \qquad \dfrac{\dfrac{\Sigma;\cdot;\cdot \rhd N : A@w \,|\, t}{\Sigma;\cdot;\cdot \rhd N : A@w \,|\, s+t}\,\mathit{CertSum2}}{\Sigma;\cdot;\cdot \rhd \mathit{box}_{s+t}\, N : [s+t]A@w \,|\, !(s+t)}\,\Box I \qquad \Sigma;\cdot;\cdot \rhd P : \mathit{Bool}@w \,|\, r}{\Sigma;\cdot;\cdot \rhd \text{if } P \text{ then } \mathit{box}_{s+t}\, M \text{ else } \mathit{box}_{s+t}\, N : [s+t]A@w \,|\, \mathit{if}(r, !(s+t), !(s+t))}\,\mathit{If}$$

Un razonamiento similar puede aplicarse sobre el operador case para justificar el uso de concatenaciones de certificados sobre este tipo de expresiones.

Finalmente, el siguiente ejemplo hace uso de alguna de las extensiones introducidas en este capítulo:

Ejemplo 5.4. En primer lugar, la siguiente función F recibe tres parámetros que representan código móvil que calculan un valor booleano b y dos valores números m y n respectivamente. Luego de inspeccionar los términos de las unidades móviles, la función realiza una suma o una multiplicación de los números obtenidos, de acuerdo al valor de verdad del código de b:

$$\lambda b.\lambda m.\lambda n.\mathit{unpack}\ b\ \mathit{to}\ \langle t^\bullet, t^\circ\rangle\ \mathit{in}\ \mathit{unpack}\ m\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ \mathit{unpack}\ n\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}$$
$$\text{if } t^\bullet \text{ then } v^\bullet + u^\bullet \text{ else } v^\bullet * u^\bullet$$

Ahora se puede aplicar la función a valores en concreto que representen unidades móviles:[1]

$$\mathrm{F}\ \ (\mathit{fetch}[w_1]\ \mathit{box}_{\mathit{true}}\, \mathbf{true})\ (\mathit{box}_{\overline{3}}\, 3)\ (\mathit{box}_{\overline{5}}\, 5)$$

Si se evalúa el término, el resultado de la computación será 8.

Las extensiones introducidas en este capítulo permiten darle mayor expresividad al lenguaje. Las mismas son utilizadas en la implementación del prototipo que se trata en el próximo capítulo.

[1]Se utiliza n directamente para denotar $\mathbf{s}^{(n)}\mathbf{z}$ y $\overline{n}$ para denotar el certificado $\mathit{Succ}^{(n)}\mathit{Zero}$

Capítulo 6

Implementación

En este capítulo se analizan algunos conceptos utilizados para la implementación de un prototipo para $\lambda_{\square}^{\mathsf{Cert}}$ en un lenguaje funcional. En primer lugar se introducen algunos conceptos generales y luego algunos detalles de la implementación relacionados al parseo, la inferencia de tipos y la evaluación.

6.1. Introducción

La implementación del prototipo para el cálculo $\lambda_{\square}^{\mathsf{Cert}}$ se realizó en el lenguaje funcional Haskell [Has09]. El motivo por el cual se eligió un lenguaje funcional para la implementación está relacionado con las facilidades que tienen este tipo de lenguajes para expresar las definiciones formales con las que ya se contaba. Para la generación de los ejecutables se utilizó el compilador GHC [GHC09].

El programa implementado es una aplicación de consola que permite cargar desde un archivo un programa escrito en el lenguaje $\lambda_{\square}^{\mathsf{Cert}}$ para luego realizar operaciones sobre el mismo. Los archivos que reconoce el programa tienen la siguiente estructura:

```
-- Estructura de archivo lambda-cert
WorldCount=2 --Especifica la cantidad total de mundos
InitialWorld=0 --Especifica el mundo inicial
Term=(\x:Nat@w0.x+1) 3 --Especifica el término a reducir
```

Como se puede apreciar en el ejemplo, en primer lugar es necesario indicar cuántos mundos o nodos cuenta la red sobre la cual se quiere ejecutar el programa (WorldCount). En segundo lugar, es necesario indicar cuál es el mundo inicial sobre el que se empieza a reducir (InitialWorld). Los mundos se identifican con un número, a partir del 0. Por último, se especifica cuál es el término a reducir (Term). Este término puede incluir todas las operaciones básicas (abstracciones, funciones, términos de desempaquetamiento, etc.) y también las extensiones (booleanos y números naturales). La sintaxis es similar a la introducida en la presentación teórica del cálculo con algunas excepciones. En primer lugar, se han realizado algunas adaptaciones para poder representar los términos utilizando el conjunto de

caracteres para archivos (por ejemplo, para representar $v^\bullet$ se utiliza `v"`). En segundo lugar, como se ve en el ejemplo y por motivos relacionados con el sistema de tipos que se explican más adelante, se ha modificado la sintaxis de las funciones para poder especificar el tipo del parámetro que reciben. Por último, se utilizan los números naturales de manera habitual (`0,1,2,...`) en lugar de la notación introducida ($zero, succ(zero), succ(succ(zero)), \ldots$).

Una vez que el programa carga el archivo especificado, se puede utilizar el mismo para visualizar el término cargado, inferir su tipo y reducirlo de acuerdo a la semántica de la máquina abstracta, ya sea paso a paso, o directamente hasta llegar a su formal normal.

Conceptualmente, se puede dividir la implementación del prototipo en tres partes: el parser, el sistema de tipos y el evaluador. En el resto de este capítulo se detallan aspectos de cada una de estas partes.

6.2. Parser

El parser es la parte del programa que permite interpretar el contenido en texto de los archivos en estructuras de datos del lenguaje para poder ser manipulados. Para escribir el parser se utilizó una herramienta llamada Happy [Hap09] que permite generar parsers de manera muy sencilla y directa. En primer lugar es necesario escribir la definición de las estrucutras de datos que se van a utilizar para representar a los términos. Por ejemplo, una versión simplificada de la estructura de los mundos, tipos, términos y certificados es la siguiente:

```
type Var = String
type CertVar = String

data World = World Int

data Type = TypeImplies Type Type
          | TypeMobile Cert Type
          | TypeNat
          | TypeBool

data Cert = CertVar CertVar
          | CertCode CertVar
          | CertApp Cert Cert
          | CertFunc Var Type Cert

data Term = Var Var
          | VarCode Var
          | Func Var Type World Term
          | App Term Term
```

Luego, es necesario definir un lexer, que transforma el contenido del archivo en una secuencia de tokens que son la entrada para la función de parseo. Ejemplos de

tokens utilizados son: una variable alfanumérica, un número, ':', '@', etc.

Por último, se deben especificar las reglas gramaticales en donde se indica la forma de escribir los programas y cómo estos últimos se mapean a las estructuras de datos. La gramática se basa en la definición 3.1 de las reglas sintácticas de $\lambda_{\square}^{\mathsf{Cert}}$ y en las extensiones definidas en 5.1 y 5.6. Estas definiciones no pudieron ser utilizadas directamente ya que las mismas son ambiguas. Por esta razón, se tuvieron que adaptar para formar una gramática no ambigua que pueda ser interpretada por el generador de parser sin ningún inconveniente. Un ejemplo de regla gramatical para funciones es el siguiente:

```
Func : '\\' a ':' Type '@' world '.' Term { Func $2 $4 (World $6) $8}
```

Aquí se está diciendo que para armar una expresión que es una función se necesita un token '\', seguido de una variable a, seguido del token ':', seguido de un tipo (otra regla), seguido del token '@', seguido de un mundo, seguido del token '.' y, por último, seguido de un término. A la derecha de la expresión se indica la manera de construir el tipo de datos Func en base a los valores leídos por el parser ($ 2, $4, etc).

Con toda esta información, el generador de parsers Happy genera automáticamente una función para parsear una lista de tokens que se utiliza desde el programa principal para cargar el contenido de los archivos.

6.3. Sistema de tipos

La implementación realizada cuenta con un inferidor de tipos que permite calcular el tipo del término cargado junto a su certificado, según las reglas definidas en 3.7. El tipo se calcula en el mundo inicial especificado para el programa. La función de inferencia tiene el siguiente tipo:

```
infer :: World -> Term -> Error(Cert, Type)
```

Como se puede ver en el tipo de retorno de la función, se han utilizado mónadas de excepción ([Wad95]) para manejar los errores de inferencia.

Para la implementación del algoritmo, se ha modificado la sintaxis de las funciones para que en las mismas se especifique el tipo del parámetro que reciben. Esto es necesario, ya que se trata de un sistema de tipos no polimórfico. Por ejemplo, no es posible inferir el tipo de la función $(\lambda x.x)$ ya que nada se sabe de x y tampoco se cuenta con variables de tipo. Una manera correcta de escribir este término es, por ejemplo, $(\lambda x : Nat@w_0.x)$, de la cual se puede inferir que tiene tipo $Nat \supset Nat$ en w_0.

El hecho de que en $\lambda_{\square}^{\mathsf{Cert}}$ se internalicen las derivaciones de tipos a través de los certificados hace que el inferidor de tipos se convierta en realidad en un híbrido entre inferidor y verificador de tipos al mismo tiempo. Por ejemplo, si se tiene un término de la forma $box_s\, M$, el algoritmo intentará inferir el tipo para M. Eso tendrá como resultado un tipo A y un certificado t. Este último certificado t deberá ser igual a s, de lo contrario el inferidor fallará. Pero s es un certificado escrito por el programador, por lo cual el algoritmo en realidad está verificando que el certificado sea correcto.

Algo similar ocurre con las funciones que toman código móvil como parámetro. En este caso, el programador debe especificar el tipo del parámetro, lo que necesariamente implica indicar un certificado para el tipo móvil.

6.4. Evaluador

El componente más importante de la implementación lo constituye el evaluador, el cual permite reducir una expresión para obtener el valor de resultado. El programa ofrece dos maneras de realizar la implementación. Una es directamente reduciendo todo el término hasta la encontrar su forma normal (lo cual está garantizado para programas bien tipados, según la propiedad de normalización fuerte). La otra forma de reducción se realiza paso a paso, aplicando una regla de reducción a la vez. Esto permite un mejor entendimiento de cómo se realizan las reducciones en $\lambda_{\square}^{\mathsf{Cert}}$. Ambas formas de reducción verifican que el programa esté bien tipado antes de realizar cualquier acción.

Para poder implementar el evaluador, fue necesario extender la definición de los tipos de datos, para incluir la representación de los estados de la máquina, el ambiente, el contexto, las capas, etc. El siguiente es un resumen de las definiciones:

```
data MachineContextItem = Return World | Finish | AppLeftCircle Term |
                          AppRightCircle Term | ...

type MachineContext = [MachineContextItem]

type MachineEnv = [(World, MachineContextStack)]

type MachineContextStack = [MachineContext]

data MachineState = MS MachineEnv World MachineContext Term
```

Como puede apreciarse, la especificación de arriba es una traducción casi directa de la definición 3.8 (por razones de espacio se omite la definición completa de `MachineContextItem`).

La función para reducir un solo paso toma el siguiente tipo:

```
reduceOneStepMachineState ::  MachineState -> MachineState
```

La implementación de la misma también sale de manera muy directa de la definición de los pasos de reducción 3.9 (y sus extensiones 5.4 y 5.9). La única excepción se hizo con los números naturales, en donde, aprovechando que la implementación utiliza los naturales en su representación natural, se implementó una semántica más directa que opere con la suma y multiplicación heredados del lenguaje.

En la implementación del evaluador se ve claramente la ventaja de haber utilizado un lenguaje funcional para el prototipo de $\lambda_{\square}^{\mathsf{Cert}}$.

Capítulo 7

Conclusiones

7.1. Trabajo relacionado

Existen varios cálculos fundacionales para programación distribuida y concurrente. Dado que el foco de este trabajo está puesto en los cálculos motivados desde la lógica, se comentan trabajos relacionados desde ese punto de vista. De acuerdo a lo investigado, ningún trabajo en la actualidad considera un cálculo que incluya tanto movilidad como generación de certificados en una teoría unificada. Sin embargo, existen varias ideas respecto al tema de movilidad. Las más cercanas a lo presentado aquí son el trabajo de Moody [Moo04], el trabajo de Murphy et al [VCHP04, VCH05, VCH07, Mur08] y el trabajo de Jia y Walker [JW04]. Moody presenta una interpretación operacional de una formulación en deducción natural de S4 en donde en donde el tipo $\Box A$ denota movilidad o independencia de locación. También considera el operador diamante, donde el tipo $\diamond A$ describe términos de tipo A localizados remotamente. Además incluye extensiones para dar soporte a un lenguaje más completo que incluye referencias. El trabajo de Murphy et al también introduce movilidad mediante la interpretación computacional de una formulación en deducción natural, pero en este caso de S5. Al igual que en este trabajo, también introduce referencias explícitas a los mundos en su modelo. Se consideran los dos operadores modales de necesidad y posibilidad. Con respecto a la semántica operacional, se considera una máquina abstracta ([VCHP04, VCH05]) y también una semántica definida en términos de una relación de evaluación *big-step* ([Mur08]). Por último, se define un lenguaje de programación completo implementado a través de un compilador especializado para aplicaciones web. Finalmente, Jia y Walker [JW04] desarrollan una lógica modal híbrida intuicionista y proveen una interpretación computacional de la misma. La lógica se define con el propósito de facilitar el razonamiento sobre sistemas distribuidos.

7.2. Conclusiones

En este trabajo se presentó un cálculo para modelar computaciones móviles que incluye la generación de certificados. El mismo se definió a partir del análisis de un

fragmento intuicionista de la lógica de pruebas LP, aplicando la técnica del isomorfismo de Curry-Howard-DeBruijn para asociar proposiciones y pruebas de la lógica ILPnd a tipos y términos del cálculo $\lambda_{\Box}^{\mathsf{Cert}}$. La interpretación computacional que se le dio al constructor modal $[s]A$ es la de una unidad móvil, una expresión que incluye tanto el código como el certificado. El sistema de tipos de $\lambda_{\Box}^{\mathsf{Cert}}$ obtenido constituye una teoría unificada para la construcción correcta de tanto código como certificados. Cuando se construye una unidad móvil a partir de otras unidades móviles, el sistema de tipos no solo asegura que la nueva unidad móvil no dependa de recursos locales, sino que también verifica que el certificado de esta se construya a partir de los certificados de sus componentes. Una vez obtenido el lenguaje y su sistema de tipos, se definió de manera precisa una semántica para el mismo basada en la ejecución de una máquina abstracta. El hecho de haber definido $\lambda_{\Box}^{\mathsf{Cert}}$ y su semántica de manera formal permitió hacer un estudio y demostración de sus propiedades más importantes. Se demostró la seguridad de tipos, que garantiza, entre otras cosas, que la ejecución de un programa bien tipado no puede fallar debido a una unidad móvil que tiene un certificado que no se corresponde a su código y normalización fuerte, que asegura que los programas bien tipados siempre terminan. Finalmente, se realizaron algunas extensiones a la definición original del cálculo que aportaron mayor riqueza a la posterior implementación del prototipo del lenguaje, que a su vez permitió poner en práctica los conceptos introducidos.

7.3. Trabajo futuro

Existen varios temas a trabajar a futuro alrededor del cálculo $\lambda_{\Box}^{\mathsf{Cert}}$ que han quedado fuera del alcance del trabajo.

Un tema sobre el que se puede seguir investigando es la inclusión del operador de posibilidad $\diamond$. La razón por la cual no se consideró en este trabajo es que la lógica de pruebas no lo trata pues, como se basa en la lógica clásica, no lo necesita. Una opción podría ser investigar una interpretación de este conectivo en el fragmento intuicionista de LP. Otra opción podría ser directamente agregar los esquemas de inferencia al cálculo siguiendo la línea de los trabajos relacionados. En general, se interpreta que un término de tipo $\diamond A$ denota el valor de un término en un nodo remoto. Si bien esta opción rompe la conexión con la lógica de pruebas, podría llegar a tener sentido desde el punto de vista de la programación.

Otro grupo de extensiones posibles están relacionadas con la posibilidad de darle más expresividad al lenguaje $\lambda_{\Box}^{\mathsf{Cert}}$. Si bien este cálculo se definió recortado para poder ser estudiado formalmente, claramente carece de herramientas necesarias para construir ejemplos más extensos. Dos adiciones básicas que deben considerarse son referencias y recursión (utlizando el operador de punto fijo fix). Otra adición posible es agregar polimorfismo sobre las variables de certificado. Así, el ejemplo 3.1 en lugar de tener tipo $[s](A \supset B) \supset [t]A \supset [s \cdot t]$, pasaría a tener tipo $\forall\alpha\forall\beta.[\alpha](A \supset B) \supset [\beta]A \supset [\alpha \cdot \beta]$. Esto está relacionado con el problema comentado en el final de la sección de sistema de tipos de la implementación. Este tipo de polimorfismo permitiría más flexibilidad en relación al uso de certificados a la

hora escribir programas. También sería interesante agregar polimorfismo a las variables de función, de manera que, por ejemplo, la función $(\lambda x.x)$ pueda tener tipo $\forall\alpha.\alpha \supset \alpha$.

Apéndice A

Demostraciones

Propiedad 1. La demostración se realiza por inducción sobre la derivación de $\Sigma; \Delta; \Gamma \rhd M : A@w'' \mid s$

- Caso 1: $M = a$. Supongamos que:

  (1.1) $\Sigma; \Delta; \Gamma \rhd a : A@w'' \mid s$

  De (1.1) y por la regla VarT, $s = a$ y $\Gamma = \Gamma_1, a : A@w'', \Gamma_2$ para algún Γ_1 y Γ_2. Luego, $\Gamma\{w/w'\} = \Gamma_1\{w/w'\}, a : A@w''\{w/w'\}, \Gamma_2\{w/w'\}$. Además, $a\{w/w'\} = a$. Aplicando la regla VarT vale que:

  (1.2) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd a\{w/w'\} : A@w''\{w/w'\} \mid s$

- Caso 2: $M = v^\bullet$. Supongamos que:

  (2.1) $\Sigma; \Delta; \Gamma \rhd v^\bullet : A@w'' \mid s$

  De (2.1) y por la regla VarV, $s = v^\circ$ y $\Delta = \Delta_1, v : A@w'', \Delta_2$ para algún Δ_1 y Δ_2. Luego, $\Delta\{w/w'\} = \Delta_1\{w/w'\}, v : A@w''\{w/w'\}, \Delta_2\{w/w'\}$. Además, $v^\bullet\{w/w'\} = v^\bullet$. Aplicando la regla VarV vale que:

  (2.2) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd v^\bullet\{w/w'\} : A@w''\{w/w'\} \mid s$

- Caso 3: $M = PQ$. Supongamos que:

  (3.1) $\Sigma; \Delta; \Gamma \rhd PQ : A@w'' \mid s$

  De (3.1) y por la regla $\supset E$, $\exists t_1, t_2, B$, tales que $s = t_1 \cdot t_2$ y además:

  (3.2) $\Sigma; \Delta; \Gamma \rhd P : B \supset A@w'' \mid t_1$

  (3.3) $\Sigma; \Delta; \Gamma \rhd Q : B@w'' \mid t_2$

  De (3.2) y por HI

  (3.4) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd P\{w/w'\} : (B \supset A)@w''\{w/w'\} \mid t_1$

  De (3.3) y por HI

(3.5) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd Q\{w/w'\} : B@w''\{w/w'\} \mid t_2$

De (3.4) y (3.5) por $\supset E$

(3.6) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd P\{w/w'\}Q\{w/w'\} : A@w''\{w/w'\} \mid s$

De (3.6) y por la definición de sustitución:

(3.7) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd (PQ)\{w/w'\} : A@w''\{w/w'\} \mid s$

- Caso 4: $M = \lambda a.P$. Supongamos que:

(4.1) $\Sigma; \Delta; \Gamma \rhd \lambda a.P : A@w'' \mid s$

De (4.1) y por la regla $\supset I$, $\exists B, B', t$ tales que $s = \lambda a : B.t$, $A = B \supset B'$ y además:

(4.2) $\Sigma; \Delta; \Gamma, a : B@w'' \rhd P : B'@w'' \mid t$

De (4.2) y por HI:

(4.3) $\Sigma; \Delta\{w/w'\}; (\Gamma, a : B@w'')\{w/w'\} \rhd P\{w/w'\} : B'@w''\{w/w'\} \mid t$

Como $(\Gamma, a : B@w'')\{w/w'\} = \Gamma\{w/w'\}, a : B@w''\{w/w'\}$, de (4.3) aplicando la regla $\supset I$

(4.4) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd \lambda a.P\{w/w'\} : A@w''\{w/w'\} \mid s$

De (4.4) por la definición de sustitución:

(4.5) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd (\lambda a.P)\{w/w'\} : A@w''\{w/w'\} \mid s$

- Caso 5: $M = box_t\, P$. Supongamos que:

(5.1) $\Sigma; \Delta; \Gamma \rhd box_t\, P : A@w'' \mid s$

De (5.1) y por la regla $\Box I$ $\exists B$ tal que $A = [t]B$, $s =!t$ y además

(5.2) $\Sigma; \Delta; \cdot \rhd P : B@w'' \mid t$

De (5.2), por HI vale que

(5.3) $\Sigma; \Delta\{w/w'\}; \cdot \rhd P\{w/w'\} : B@w''\{w/w'\} \mid t$

Luego, de (5.3) por la regla $\Box I$

(5.4) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd box_t\, P\{w/w'\} : [t]B@w''\{w/w'\} \mid !t$

De (5.4) por la definición de sustitución:

(5.5) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd (box_t\, P)\{w/w'\} : [t]B@w''\{w/w'\} \mid !t$

- Caso 6: $M = fetch[w_f]\, P$. Supongamos que:

(6.1) $\Sigma; \Delta; \Gamma \rhd \mathit{fetch}[w_f]\, P : A@w'' \mid s$

De (6.1) por la regla Fetch $\exists B, t, r$ tales que $A = [t]B$, $s = fetch(r)$ y además

(6.2) $\Sigma; \Delta; \Gamma \rhd P : [t]B@w_f \mid r$

De (6.2) por HI

(6.3) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd P\{w/w'\} : [t]B@w_f\{w/w'\} \mid r$

De (6.3) aplicando la regla Fetch

(6.4) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd \mathit{fetch}[w_f\{w/w'\}]\, P\{w/w'\} : [t]B@w''\{w/w'\} \mid fetch(r)$

De (6.4) y por la definición de sustitución

(6.5) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd (\mathit{fetch}[w_f]\, P)\{w/w'\} : A@w''\{w/w'\} \mid s$

- Caso 7: $M = \mathit{unpack}\ P\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ Q$. Supongamos que:

(7.1) $\Sigma; \Delta; \Gamma \rhd \mathit{unpack}\ P\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ Q : B@w'' \mid s$

De (7.1) por la regla $\Box E$, $\exists C, t_1, t_2, r$ tales que $s = \mathit{letc}\ t_1\ \mathit{be}\ v : A\ \mathit{in}\ t_2$, $B = C\{v^\circ/r\}$ y además

(7.2) $\Sigma; \Delta; \Gamma \rhd P : [r]A@w'' \mid t_1$

(7.3) $\Sigma; \Delta, v : A@w''; \Gamma \rhd Q : C@w'' \mid t_2$

De (7.2) y por HI

(7.4) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd P\{w/w'\} : [r]A@w''\{w/w'\} \mid t_1$

(7.5) $\Sigma; (\Delta, v : A@w'')\{w/w'\}; \Gamma\{w/w'\} \rhd Q\{w/w'\} : C@w''\{w/w'\} \mid t_2$

Como $(\Delta, v : A@w'')\{w/w'\} = \Delta\{w/w'\}, v : A@w''\{w/w'\}$, de (7.4) y (7.5) por la regla $\Box E$

(7.6) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd \mathit{unpack}\ P\{w/w'\}\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ Q\{w/w'\} : B@w''\{w/w'\} \mid \mathit{letc}\, t_1\, \mathit{be}\, v : A\ \mathit{in}\ t_2$

De (7.6) por la definición de sustitución:

(7.7) $\Sigma; \Delta\{w/w'\}; \Gamma\{w/w'\} \rhd (\mathit{unpack}\ P\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ Q)\{w/w'\} : B@w''\{w/w'\} \mid s$

□

Lema 1. Demostración: por inducción sobre la derivación de $\Sigma; \Delta; \Gamma \rhd M : A@w \mid s$

- Caso 1 (base). $M = a$. Supongamos que:

(1.1) $\Sigma; \Delta; \Gamma \rhd a : A@w \mid s$

De (1.1) y por la regla VarT $\exists \Gamma_1, \Gamma_2$ tal que $\Gamma = \Gamma_1, a : A@w, \Gamma_2$ y además $s = a$. Como $fv(a) = a = fv(s)$ y $a \in dom(\Gamma)$ se cumple la propiedad.

- Caso 2 (inductivo) $M = PQ$. Supongamos que:

(2.1) $\Sigma; \Delta; \Gamma \rhd PQ : A@w \mid s$

De (2.1) y por la regla $\supset E$ $\exists s_1, s_2, B$ tal que $s = s_1 \cdot s_2$ y además

(2.2) $\Sigma; \Delta; \Gamma \rhd P : B \supset A@w \mid s_1$

(2.3) $\Sigma; \Delta; \Gamma \rhd Q : B@w \mid s_2$

De (2.2) por HI vale que si $a \in fv(P)$ $(a \in fv(s_1))$ entonces $a \in dom(\Gamma)$. De (3.3) y por HI vale que si $a \in fv(Q)$ $(a \in fv(s_2))$ entonces $a \in dom(\Gamma)$. Sea $a \in fv(PQ)$ $(a \in s_1 \cdot s_2)$, entonces $a \in fv(P)$ o $a \in fv(Q)$ $(a \in f(s_1)$ o $a \in (s_2))$. Luego, $a \in dom(\Gamma)$.

- Caso 3 (inductivo) $M = \lambda a.P$. Supongamos que:

(3.1) $\Sigma; \Delta; \Gamma \rhd \lambda a.P : A@w \mid s$

De (3.1) y por la regla $\supset I$ $\exists A_1, A_2, t$ tales que $A = A_1 \supset A_2$, $s = \lambda a : A_1.t$ y además

(3.2) $\Sigma; \Delta; \Gamma, a : A_1@w \rhd P : A_2@w \mid t$

De (3.2) y por HI si $b \in fv(P)$ $(b \in fv(t))$ entonces $b \in dom(\Gamma, a : A_1@w) = a, dom(\Gamma)$. Sea $b \in fv(\lambda a.P)$, entonces $b \in fv(P) - \{a\}$. Por la hipótesis y por ser $b \neq a$, $b \in dom(\Gamma)$. Análogamente se demuestra que si $b \in fv(s)$ entonces $b \in dom(\Gamma)$

- Caso 4 (base) $M = v^\bullet$. Como $fv(v^\bullet) = \emptyset$ la propiedad se cumple.

- Caso 5 (inductivo) $M = box_t\, P$. Supongamos que:

(5.1) $\Sigma; \Delta; \Gamma \rhd box_t\, P : A@w \mid s$

De (5.1) por la regla $\Box I$ $\exists B$ tal que $A = [t]B$ y $s =!t$ y además

(5.2) $\Sigma; \Delta; \cdot \rhd P : B@w \mid t$

De (5.2) por HI si $a \in fv(P)$ entonces $a \in \cdot$. Luego, $fv(P) = \emptyset$ (análogamente, $fv(t) = \emptyset$). Como $fv(box_t\, P) = fv(P) = \emptyset$ y $fv(!t) = fv(t) = \emptyset$ se cumple la propiedad.

- Caso 6 (inductivo) $M = unpack\ P\ to\ \langle v^\bullet, v^\circ \rangle\ in\ Q$. Supongamos que:

(6.1) $\Sigma; \Delta; \Gamma \rhd unpack\ P\ to\ \langle v^\bullet, v^\circ \rangle\ in\ Q : A@w \mid s$

De (6.1) y por la regla $\Box E$, $\exists C, s_1, s_2, r$ tales que $s = letc\ s_1\ be\ v : B\ in\ s_2$, $A = C\{v^\circ / r\}$ y además

(6.2) $\Sigma; \Delta; \Gamma \rhd P : [r]B@w \mid s_1$

(6.3) $\Sigma; \Delta, v : B; \Gamma \rhd Q : C@w \mid s_2$

De (6.2) y por HI, si $a \in fv(P)$ ($a \in fv(s1)$) entonces $a \in dom(\Gamma)$. De (6.3) y por HI, si $a \in fv(Q)$ entonces $a \in dom(\Gamma)$ ($a \in fv(s2)$). Sea $a \in fv(unpack\ P\ to\ \langle v^\bullet, v^\circ \rangle\ in\ Q)$ ($a \in letc\ s_1\ be\ v : B\ in\ s_2$), entonces $a \in fv(P)$ o $fv(Q)$ ($a \in fv(s_1)$ o $fv(s_2)$). Luego, $a \in dom(\Gamma)$.

- Caso 7 (inductivo) $M = fetch[w']\, P$. Supongamos que:

(7.1) $\Sigma; \Delta; \Gamma \rhd fetch[w']\, P : A@w \mid s$

De (7.1) y por la regla Fetch $\exists B, t, r$ tales que $A = [t]B$, $s = fetch(r)$ y además

(7.2) $\Sigma; \Delta; \Gamma \rhd P : [t]B@w' \mid r$

De (7.2) y por HI si $a \in fv(P)$ ($a \in fv(s)$) entonces $a \in dom(\Gamma)$. Sea $a \in fv(fetch[w']\, P)$, luego $a \in fv(P)$, por lo tanto, $a \in dom(\Gamma)$

□

Propiedad 2. La demostración se realiza por inducción en la derivación de $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd N : B@w' \mid t$

- Caso 1:
  - Subcaso 1: $N = a$. Supongamos que

  (1.1) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd M : A@w \mid s$

  (1.2) $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd a : B@w' \mid t$

  Como $a \notin dom(\Gamma_1)$ y $a \notin dom(\Gamma_2)$, por la regla VarT, $w = w'$, $t = a$ y $A = B$. Luego, de (1.1),

  (1.3) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd M : B@w' \mid s$

  Como $a\{a/M\} = M$ y $t\{a/s\} = s$, de (1.3) vale que

  (1.4) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd a\{a/M\} : A@w' \mid t\{a/s\}$
  - Subcaso 2: $N = b \neq a$. Supongamos que

  (1.5) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd M : A@w \mid s$

  (1.6) $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd b : B@w' \mid t$

  Como $a \neq b$, por la regla VarT, deben existir Γ_1' y Γ_2' tal que $\Gamma_1, \Gamma_2 = \Gamma_1', b : B@w', \Gamma_2'$; además, $t = b$. Luego, por VarT

  (1.7) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd b : B@w' \mid b$

  Como $b\{a/M\} = b$ y $b\{a/s\} = b$, de (1.7) vale que

  (1.8) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd b\{a/M\} : B@w' \mid t\{a/s\}$
- Caso 2: $N = PQ$. Supongamos que

(2.1) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd M : A@w \mid s$

(2.2) $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd PQ : B@w' \mid t$

De (2.2) y por $\supset E$ $\exists t_1, t_2$ tales que $t = t_1 \cdot t_2$ y $\exists B'$ tal que

(2.3) $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd P : B' \supset B@w' \mid t_1$

(2.4) $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd Q : B'@w' \mid t_2$

De (2.1) y (2.3) por H.I.

(2.5) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd P\{a/M\} : B' \supset B@w' \mid t_1\{a/s\}$

De (2.1) y (2.4) por H.I.

(2.6) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd Q\{a/M\} : B'@w' \mid t_2\{a/s\}$

De (2.5) y (2.6) por $\supset E$

(2.7) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd P\{a/M\}Q\{a/M\} : B@w' \mid t_1\{a/s\} \cdot t_2\{a/s\}$

Por la definición de sustitución en términos y certificados vale que

(2.8) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd (PQ)\{a/M\} : B@w' \mid (t_1 \cdot t_2)\{a/s\}$

- Caso 3: $N = \lambda b.P$. Supongamos que

(3.1) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd M : A@w \mid s$

(3.2) $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd \lambda b.P : B@w' \mid t$

De (3.2) por la regla $\supset$I, $b \notin dom(\Gamma_1, a : A@w, \Gamma_2)$. Luego $a \neq b$. Además $\exists t_1, B_1, B_2$ tales que $t = \lambda b : B_1.t_1$ y $B = B_1 \supset B_2$ y

(3.3) $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2, b : B_1@w' \rhd P : B_2@w' \mid t_1$

De (3.1) y (3.3) por H.I.

(3.4) $\Sigma; \Delta; \Gamma_1, \Gamma_2, b : B_1@w' \rhd P\{a/M\} : B_2@w' \mid t_1\{a/s\}$

De (3.4) y por $\supset$I

(3.5) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd \lambda b.P\{a/M\} : B_1 \supset B_2@w' \mid \lambda y : B_1.t_1\{a/s\}$

De (3.5) y por la definición de sustitución en términos y certificados vale que

(3.6) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd (\lambda b.P)\{a/M\} : B@w' \mid t\{a/s\}$

- Caso 4: $N = v^\bullet$. Supongamos que

(4.1) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd M : A@w \mid s$

(4.2) $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd v^\bullet : B@w' \mid t$

De (4.2), por la regla VarV, $\Delta = \Delta_1, v : A@w', \Delta_2$ para algún Δ_1 y Δ_2. Además, $t = v^\circ$. Por otro lado, por las definiciones de sustitución en términos y certificados, $v^\bullet\{a/M\} = v^\bullet$ y $v^\circ\{a/s\} = v^\circ$. Luego, aplicando nuevamente la regla VarV

(4.3) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd v^\bullet\{a/s\} : B@w' \mid t\{a/s\}$

- Caso 5: $N = box_t\, P$. Supongamos que

(5.1) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd M : A@w \mid s$

(5.2) $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd box_t\, P : [t]B@w' \mid !t$

De (5.2) y por la regla $\Box I$

(5.3) $\Sigma; \Delta; \cdot \rhd P : B@w' \mid t$

De (5.3) y aplicando el lema 1, a no aparece libre ni en P ni en t (ya que si apareciera libre debería estar en $dom(\cdot) = \cdot$). Luego $P = P\{a/M\}$ y $t = t\{a/s\}$. Por lo tanto de (5.2) y aplicando la sustitución en términos y certificados vale que

(5.4) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd (box_t\, P)\{a/M\} : [t]B@w' \mid (!t)\{a/s\}$

- Caso 6 : $N = unpack\ P\ to\ \langle v^\bullet, v^\circ\rangle\ in\ Q$. Supongamos que

(6.1) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd M : A@w \mid s$

(6.2) $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd unpack\ P\ to\ \langle v^\bullet, v^\circ\rangle\ in\ Q : B@w' \mid t$

De (6.2) y por la regla $\Box E$, $\exists r, s_1, t_1, C$ tales que $B = C\{v^\circ/r\}$ y $t = letc\, s_1\, be\, v : A'\ in\ t_1$ y además

(6.3) $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd P : [r]A'@w' \mid s_1$

(6.4) $\Sigma; \Delta, v : A@w'; \Gamma_1, a : A@w, \Gamma_2 \rhd Q : C@w' \mid t_1$

De (6.1), (6.3) y por H.I.

(6.5) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd P\{a/M\} : [r]A'@w' \mid s_1\{a/s\}$

De (6.1), (6.4) y por H.I.

(6.6) $\Sigma; \Delta, v : A@w'; \Gamma_1, \Gamma_2 \rhd Q\{a/M\} : C@w' \mid t_1\{a/s\}$

De (6.5) y (6.6) y por la regla $\Box E$

(6.7) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd unpack\ P\{a/M\}\ to\ \langle v^\bullet, v^\circ\rangle\ in\ Q\{a/M\} : B@w' \mid letc\, s_1\{a/s\}\, be\, v : A'\ in\ t_1\{a/s\}$

De (6.7) y por las definiciones de sustitución en términos y certificados

(6.8) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd (unpack\ P\ to\ \langle v^\bullet, v^\circ\rangle\ in\ Q)\{a/M\} : B@w' \mid t\{a/s\}$

- Caso 7 : $N = fetch[w'']\, P$. Supongamos que

(7.1) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd M : A@w \,|\, s$

(7.2) $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd fetch[w'']\, P : B@w' \,|\, t$

De (7.2) y por la regla Fetch, $\exists C, t_1, t_2$ tal que $B = [t_2]C$, $t = fetch(t_1)$ y además:

(7.3) $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd P : [t_2]C@w'' \,|\, t_1$

De (7.1), (7.3) y por H.I

(7.3) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd P\{a/M\} : [t_2]C@w'' \,|\, t_1\{a/s\}$

De (7.3) y por la regla Fetch

(7.4) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd fetch[w'']\, P\{a/M\} : [t_2]C@w'' \,|\, fetch(t_1\{a/s\})$

De (7.4) y por las definiciones de sustitución en términos y certificados

(7.5) $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd (fetch[w'']\, P)\{a/M\} : [t_2]C@w'' \,|\, t\{a/s\}$

□

Propiedad 3. La demostración es por inducción en la derivación de $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd N : B@w' \,|\, t$

- Caso 1
  - Subcaso 1: $N = v^\bullet$. Supongamos que:

(1.1) $\Sigma; \Delta_1, \Delta_2; \cdot \rhd M : A@w \,|\, s$

(1.2) $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd v^\bullet : B@w' \,|\, t$

De (1.2) y por la regla VarV, $B = A$, $w = w'$ y $t = v^\circ$. Además, $v^\bullet\{v^\circ/s\}\{v^\bullet/M\} = M$, $B\{v^\circ/s\} = B$ pues v° no puede ocurrir en B y $t\{v^\circ/s\} = s$. Luego, de (1.1) vale que

(1.3) $\Sigma; \Delta_1, \Delta_2; \cdot \rhd v^\bullet\{v^\circ/s\}\{v^\bullet/M\} : B\{v^\circ/s\}@w' \,|\, t\{v^\circ/s\}$

De (1.3) y aplicando el lema 2

(1.4) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd v^\bullet\{v^\circ/s\}\{v^\bullet/M\} : B\{v^\circ/s\}@w' \,|\, t\{v^\circ/s\}$

  - Subcaso 2: $N = u^\bullet \neq v^\bullet$. Supongamos que:

(1.5) $\Sigma; \Delta_1, \Delta_2; \cdot \rhd M : A@w \,|\, s$

(1.6) $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd u^\bullet : B@w' \,|\, t$

De (1.5) y por la regla VarV, $t = u^\circ$ y $\exists \Delta_1', \Delta_2'$ tales que $\Delta_1, \Delta_2 = \Delta_1', u : B@w', \Delta_2'$. Además, $u^\bullet\{v^\circ/s\}\{v^\bullet/M\} = u^\bullet$, $B\{v^\circ/s\} = B$ pues v° no puede ocurrir en B y $t\{v^\circ/s\} = u^\circ$. Luego, aplicando la regla VarV vale que:

(1.7) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd u^\bullet\{v^\circ/s\}\{v^\bullet/M\} : B\{v^\circ/s\}@w' \,|\, t\{v^\circ/s\}$

- Caso 2 $N = a$. Supongamos que:

(2.1) $\Sigma; \Delta_1, \Delta_2; \cdot \rhd M : A@w \,|\, s$

(2.2) $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd a : B@w' \,|\, t$

De (2.2) y por la regla VarT, $t = a$ y $\exists \Gamma_1, \Gamma_2$ tal que $\Gamma = \Gamma_1, a : B@w', \Gamma_2$. Además $a\{v^\circ/s\}\{v^\bullet/M\} = a$, $t\{v^\circ/s\} = t$ y $B\{v^\circ/s\} = B$ pues v° no puede ocurrir en B. Luego, aplicando la regla VarT vale que:

(2.3) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd a\{v^\circ/s\}\{v^\bullet/M\} : B\{v^\circ/s\}@w' \,|\, t\{v^\circ/s\}$

- Caso 3 $N = PQ$. Supongamos que:

(3.1) $\Sigma; \Delta_1, \Delta_2; \cdot \rhd M : A@w \,|\, s$

(3.2) $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd PQ : B@w' \,|\, t$

De (3.2) y por la regla $\supset E$, $\exists B', t_1, t_2$ tales que $t = t_1 \cdot t_2$ y además

(3.3) $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd P : B' \supset B@w' \,|\, t_1$

(3.4) $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd Q : B'@w' \,|\, t_2$

De (3.1), (3.3) y por HI:

(3.5) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd P\{v^\circ/s\}\{v^\bullet/M\} : (B' \supset B)\{v^\circ/s\}@w' \,|\, t_1\{v^\circ/s\}$

De (3.1), (3.4) y por HI:

(3.6) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd Q\{v^\circ/s\}\{v^\bullet/M\} : B'\{v^\circ/s\}@w' \,|\, t_2\{v^\circ/s\}$

Como $(B' \supset B)\{v^\circ/s\} = B'\{v^\circ/s\} \supset B\{v^\circ/s\}$, de (3.5) y (3.6) aplicando la regla $\supset$E

(3.7) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd P\{v^\circ/s\}\{v^\bullet/M\}Q\{v^\circ/s\}\{v^\bullet/M\} : B\{v^\circ/s\}@w' \,|\, t_1\{v^\circ/s\} \cdot t_2\{v^\circ/s\}$

De (3.7) y por la definición de sustitución,

(3.7) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd (PQ)\{v^\circ/s\}\{v^\bullet/M\} : B\{v^\circ/s\}@w' \,|\, t\{v^\circ/s\}$

- Caso 4 $N = \lambda a.P$. Supongamos que:

(4.1) $\Sigma; \Delta_1, \Delta_2; \cdot \rhd M : A@w \,|\, s$

(4.2) $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd \lambda a.P : B@w' \,|\, t$

De (4.2) y por la regla $\supset I$, $\exists B_1, B_2, t_1$ tales que $B = B_1 \supset B_2$, $t = \lambda a : B_1.t_1$ y además

(4.3) $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma, a : B_1@w' \rhd P : B_2@w' \,|\, t_1$

Notar que de (4.3) v° no puede estar en B_1. De (4.1) y (4.3) por HI:

(4.4) $\Sigma; \Delta_1, \Delta_2; \Gamma, a : B_1@w' \rhd P : B_2\{v^\circ/s\}@w' \,|\, t_1\{v^\circ/s\}$

De (4.4) y por regla $\supset$I

(4.5) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd \lambda a.P\{v^\circ/s\}\{v^\bullet/M\} : B_1 \supset B_2\{v^\circ/s\}@w' \,|\, \lambda a : B_1.t_1\{v^\circ/s\}$

Como $B_1\{v^\circ/s\} = B_1$, de (4.2) por la definición de sustitución

(4.6) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd (\lambda a.P)\{v^\circ/s\}\{v^\bullet/M\} : B\{v^\circ/s\}@w' \,|\, t\{v^\circ/s\}$

- Caso 5 $N = box_r\, P$. Supongamos que:

(5.1) $\Sigma; \Delta_1, \Delta_2; \cdot \rhd M : A@w \,|\, s$

(5.2) $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd box_r\, P : B@w' \,|\, t$

De (5.2) y por la regla $\Box I$ $t = !r$, $\exists C$ tal que $B = [r]C$ y además

(5.3) $\Sigma; \Delta_1, v : A@w, \Delta_2; \cdot \rhd P : C@w' \,|\, r$

De (5.1) y (5.3) por HI

(5.4) $\Sigma; \Delta_1, \Delta_2; \cdot \rhd P\{v^\circ/s\}\{v^\bullet/M\} : C\{v^\circ/s\}@w' \,|\, r\{v^\circ/s\}$

De (5.4) por $\Box I$

(5.5) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd box_{r\{v^\circ/s\}}\, P\{v^\circ/s\}\{v^\bullet/M\} : [r\{v^\circ/s\}]C\{v^\circ/s\}@w' \,|\, !r\{v^\circ/s\}$

De (5.5) por las definiciones de sustitución,

(5.5) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd (box_r\, P)\{v^\circ/s\}\{v^\bullet/M\} : B\{v^\circ/s\}@w' \,|\, t\{v^\circ/s\}$

- Caso 6 $N = fetch[w'']\, P$. Supongamos que:

(6.1) $\Sigma; \Delta_1, \Delta_2; \cdot \rhd M : A@w \,|\, s$

(6.2) $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd fetch[w'']\, P : B@w' \,|\, t$

De (6.2) y por la regla Fetch, $\exists C, t_1, r$ tal que $B = [t_1]C$, $t = fetch(r)$ y además

(6.3) $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd P : [t_1]C@w'' \,|\, r$

De (6.1) y (6.3) por HI

(6.4) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd P\{v^\circ/s\}\{v^\bullet/M\} : ([t_1]C)\{v^\circ/s\}@w'' \,|\, r\{v^\circ/s\}$

De (6.4) por la regla Fetch

(6.5) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd fetch[w'']\, P\{v^\circ/s\}\{v^\bullet/M\} : ([t_1]C)\{v^\circ/s\}@w' \,|\, fetch(r\{v^\circ/s\})$

De (6.5), por la definición de sustitución,

(6.6) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd (fetch[w'']\, P)\{v^\circ/s\}\{v^\bullet/M\} : B\{v^\circ/s\}@w' \,|\, t\{v^\circ/s\}$

- Caso 7 $N = \mathit{unpack}\ P\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}\ Q$. Supongamos que:

  (7.1) $\Sigma; \Delta_1, \Delta_2; \cdot \rhd M : A@w \mid s$

  (7.2) $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd \mathit{unpack}\ P\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}\ Q : C@w' \mid t$

  De (7.2) por $\Box E$, $u \notin (\Delta_1, v : A@w, \Delta_2)$ por lo que $u \neq v$. Además, $\exists r, t_1, t_2, B, D$ tales que $t = \mathit{letc}\ t_1\ \mathit{be}\ u : B\ \mathit{in}\ t_2$, $C = D\{u^\circ/r\}$ y

  (7.3) $\Sigma; \Delta_1, v : A@w, \Delta_2, u : B@w'; \Gamma \rhd Q : D@w' \mid t_2$

  (7.4) $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd P : [r]B@w' \mid t_1$

  Notar de (7.3) que v° no puede estar en B. De (7.1), (7.3) y por HI

  (7.5) $\Sigma; \Delta_1, \Delta_2, u : B@w'; \Gamma \rhd Q\{v^\circ/s\}\{v^\bullet/M\} : D\{v^\circ/s\}@w' \mid t_2\{v^\circ/s\}$

  De (7.1), (7.4) y por HI

  (7.6) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd P\{v^\circ/s\}\{v^\bullet/M\} : ([r]B)\{v^\circ/s\}@w' \mid t_1\{v^\circ/s\}$

  De (7.5) y (7.6) aplicando la regla $\Box E$

  (7.7) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd \mathit{unpack}\ P\{v^\circ/s\}\{v^\bullet/M\}\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}\ Q\{v^\circ/s\}\{v^\bullet/M\} : D\{v^\circ/s\}\{u^\circ/r\}@w' \mid \mathit{letc}\ t_1\{v^\circ/s\}\ \mathit{be}\ u : B\ \mathit{in}\ t_2\{v^\circ/s\}$

  $s\{u^\circ/r\} = s$ pues u° no puede aparecer en s ya que no aparece en Δ_1, Δ_2. Como $D\{v^\circ/s\}\{u^\circ/r\} = D\{u^\circ/r\}\{v^\circ/s\{u^\circ/r\}\} = D\{u^\circ/r\}\{v^\circ/s\}$, de (7.7) por las definiciones de sustitución,

  (7.8) $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd (\mathit{unpack}\ P\ \mathit{to}\ \langle u^\bullet, u^\circ\rangle\ \mathit{in}\ Q)\{v^\circ/s\}\{v^\bullet/M\} : C\{v^\circ/s\}@w' \mid t\{v^\circ/s\}$

□

Lema 11. *Si* $\Sigma; \Delta; \Gamma \rhd M : A@w \mid s$ *y* $\Sigma \vdash \bullet$ *entonces* $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \overline{M} : A@\bullet \mid s$

Demostración. Por inducción en la derivación de $\Sigma; \Delta; \Gamma \rhd M : A@w \mid s$

- Caso 1: $M = a$. Supongamos que:

  (1.1) $\Sigma; \Delta; \Gamma \rhd a : A@w \mid s$

  De (1.1) por VarT, $s = a$ y $\Gamma = \Gamma_1, a : A@w, \Gamma_2$ para algún Γ_1 y Γ_2. Luego, $\overline{\Gamma} = \overline{\Gamma_1}, a : A@\bullet, \overline{\Gamma_2}$. Por la regla VarT,

  (1.2) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd a : A@\bullet \mid s$

  De (1.2), como $a = \overline{a}$

  (1.3) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \overline{a} : A@\bullet \mid s$

- Caso 2: $M = v^\bullet$. Supongamos que:

(2.1) $\Sigma; \Delta; \Gamma \rhd v^{\bullet} : A@w \,|\, s$

De (2.1) y por VarV, $s = v^{\circ}$ y $\Delta = \Delta_1, v : A@w, \Delta_2$ para algún Δ_1 y Δ_2. Luego, $\overline{\Delta} = \overline{\Delta_1}, v : A@\bullet, \overline{\Delta_2}$. Por la regla VarV,

(2.2) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd v^{\bullet} : A@\bullet \,|\, s$

De (2.2) como $v^{\bullet} = \overline{v^{\bullet}}$

(2.2) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \overline{v^{\bullet}} : A@\bullet \,|\, s$

- Caso 3: $M = PQ$. Supongamos que:

(3.1) $\Sigma; \Delta; \Gamma \rhd PQ : A@w \,|\, s$

De (3.1) y por $\supset E$, $\exists t_1, t_2, B$, tales que $s = t_1 \cdot t_2$ y,

(3.2) $\Sigma; \Delta; \Gamma \rhd P : B \supset A@w \,|\, t_1$
(3.3) $\Sigma; \Delta; \Gamma \rhd Q : B@w \,|\, t_2$

De (3.2) y por HI

(3.4) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \overline{P} : (B \supset A)@\bullet \,|\, t_1$

De (3.3) y por HI

(3.5) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \overline{Q} : B@\bullet \,|\, t_2$

De (3.4) y (3.5) por $\supset E$

(3.6) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \overline{P}\,\overline{Q} : A@\bullet \,|\, s$

De (3.6) como $\overline{P}\,\overline{Q} = \overline{P\,Q}$

(3.7) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \overline{P\,Q} : A@\bullet \,|\, s$

- Caso 4: $M = \lambda a.P$. Supongamos que:

(4.1) $\Sigma; \Delta; \Gamma \rhd \lambda a.P : A@w \,|\, s$

De (4.1) y por $\supset I$, $\exists B, B', t$ tales que $s = \lambda a : B.t$, $A = B \supset B'$ y,

(4.2) $\Sigma; \Delta; \Gamma, a : B@w \rhd P : B'@w \,|\, t$

De (4.2) y por HI:

(4.3) $\Sigma; \overline{\Delta}; \overline{\Gamma, a : B@w} \rhd \overline{P} : B'@\bullet \,|\, t$

Como $\overline{\Gamma, a : B@w} = \overline{\Gamma}, a : B@\bullet$, de (4.3) por la regla $\supset I$

(4.4) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \lambda a.\overline{P} : A@\bullet \,|\, s$

De (4.4) como $\lambda a.\overline{P} = \overline{\lambda a.P}$

(4.5) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \overline{\lambda a.P} : A@\bullet \mid s$

- Caso 5: $M = box_t\, P$. Supongamos que:

  (5.1) $\Sigma; \Delta; \Gamma \rhd box_t\, P : A@w \mid s$

  De (5.1) y por $\Box I$ $\exists B$ tales que $A = [t]B$, $s = !t$ y

  (5.2) $\Sigma; \Delta; \cdot \rhd P : B@w \mid t$

  De (5.2) y por HI

  (5.3) $\Sigma; \overline{\Delta}; \cdot \rhd \overline{P} : B@\bullet \mid t$

  De (5.3) y por la regla $\Box I$

  (5.4) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd box_t\, \overline{P} : [t]B@\bullet \mid !t$

  De (5.4) como $box_t\, \overline{P} = \overline{box_t\, P}$:

  (5.5) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \overline{box_t\, P} : [t]B@\bullet \mid !t$

- Caso 6: $M = fetch[w_f]\, P$. Supongamos que:

  (6.1) $\Sigma; \Delta; \Gamma \rhd fetch[w_f]\, P : A@w \mid s$

  De (6.1) por Fetch $\exists B, t, r$ tales que $A = [t]B$, $s = fetch(r)$ y

  (6.2) $\Sigma; \Delta; \Gamma \rhd P : [t]B@w_f \mid r$

  De (6.2) por HI

  (6.3) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \overline{P} : [t]B@\bullet \mid r$

  De (6.3) por la regla Fetch

  (6.4) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd fetch[\bullet]\, \overline{P} : [t]B@\bullet \mid fetch(r)$

  De (6.4) como $fetch[\bullet]\, \overline{P} = \overline{fetch[w_f]\, P}$

  (6.5) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \overline{fetch[w_f]\, P} : A@\bullet \mid s$

- Caso 7: $M = unpack\ P\ to\ \langle v^\bullet, v^\circ \rangle\ in\ Q$. Supongamos que:

  (7.1) $\Sigma; \Delta; \Gamma \rhd unpack\ P\ to\ \langle v^\bullet, v^\circ \rangle\ in\ Q : B@w \mid s$

  De (7.1) por la regla $\Box E$, $\exists C, t_1, t_2, r$ tales que $s = letc\ t_1\ be\ v : A\ in\ t_2$, $B = C\{v^\circ / r\}$ y,

  (7.2) $\Sigma; \Delta; \Gamma \rhd P : [r]A@w \mid t_1$

  (7.3) $\Sigma; \Delta, v : A@w; \Gamma \rhd Q : C@w \mid t_2$

  De (7.2) por HI

(7.4) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \overline{P} : [r]A@\bullet \mid t_1$

(7.5) $\Sigma; \overline{\Delta, v : A@w}; \overline{\Gamma} \rhd \overline{Q} : C@\bullet \mid t_2$

Como $\overline{\Delta, v : A@w} = \overline{\Delta}, v : A@\bullet$, de (7.4) y (7.5) por la regla $\Box E$

(7.6) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \mathit{unpack}\ \overline{P}\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ \overline{Q} : B@\bullet \mid \mathit{letc}\ t_1\ \mathit{be}\ v : A\ \mathit{in}\ t_2$

De (7.6) como $\mathit{unpack}\ \overline{P}\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ \overline{Q} = \overline{\mathit{unpack}\ P\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ Q}$

(7.7) $\Sigma; \overline{\Delta}; \overline{\Gamma} \rhd \overline{\mathit{unpack}\ P\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ Q} : B@\bullet \mid s$

□

Lema 12. $\overline{M\{w'/w\}} = \overline{M}$

Demostración. Por inducción sobre M

- Caso $v^\bullet$

$$\overline{v^\bullet\{w'/w\}} = \overline{v^\bullet}$$

- Caso a

$$\overline{a\{w'/w\}} = \overline{a}$$

- Caso $M\,N$

$$\begin{array}{lll} & \overline{(M\,N)\{w'/w\}} & \\ = & \overline{M\{w'/w\}\,N\{w'/w\}} & \\ = & \overline{M\{w'/w\}}\,\overline{N\{w'/w\}} & \text{HI} \\ = & \overline{M}\,\overline{N} & \\ = & \overline{M\,N} & \end{array}$$

- Caso $\lambda a.M$

$$\begin{array}{lll} & \overline{(\lambda a.M)\{w'/w\}} & \\ = & \overline{\lambda a.M\{w'/w\}} & \\ = & \lambda a.\overline{M\{w'/w\}} & \text{HI} \\ = & \lambda a.\overline{M} & \\ = & \overline{\lambda a.M} & \end{array}$$

- Caso $\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N$

$$\begin{array}{lll} & \overline{(\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N)\{w'/w\}} & \\ = & \overline{\mathit{unpack}\ M\{w'/w\}\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N\{w'/w\}} & \\ = & \mathit{unpack}\ \overline{M\{w'/w\}}\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ \overline{N\{w'/w\}} & \text{HI} \\ = & \mathit{unpack}\ \overline{M}\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ \overline{N} & \\ = & \overline{\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N} & \end{array}$$

- Caso $fetch[w'']\, M$

$$\begin{array}{lll} & \overline{(fetch[w'']\, M)\{w'/w\}} & \\ = & \overline{fetch[w''\{w'/w\}]\, M\{w'/w\}} & \\ = & fetch[\bullet]\, \overline{M\{w'/w\}} & \text{HI} \\ = & fetch[\bullet]\, \overline{M} & \\ = & \overline{M} & \end{array}$$

□

Lema 13. *Sea* $\mathbb{N}$ *igual* $\mathbb{W}; w : [k, M]$. *Si* $\Sigma \vdash \mathbb{N}$ *y* $\overline{M} = \overline{M'}$ *entonces* $F(\mathbb{W}; w : [k, M]) = F(\mathbb{W}; w : [k, M'])$

En particular por el lema 12 $\overline{M\{w'/w\}} = \overline{M}$, $F(\mathbb{W}; w : [k, M\{w'/w\}]) = F(\mathbb{W}; w : [k, M])$

Demostración. Por inducción sobre $\langle |\mathbb{W}|, k\rangle$.

- Caso $\langle |\mathbb{W}|, finish\rangle$. $\mathbb{N} = \mathbb{W}; w : [finish, M]$.
  $F(\mathbb{W}; w : [finish, M]) = \overline{M} = \overline{M'} = F(\mathbb{W}; w : [k, M'])$.

- Caso $\langle |\mathbb{W}|, k \lhd \circ\, N\rangle$. $\mathbb{N} = \mathbb{W}; w : [k \lhd \circ\, N, M]$.
  $F(\mathbb{W}; w : [k \lhd \circ\, N, M]) = F(\mathbb{W}; w : [k, M\, N])$. Si $\Sigma \vdash \mathbb{W}; w : [k \lhd \circ\, N, M]$, entonces por *MState* existen C, s tales que:

  (1) $\Sigma; \cdot; \cdot \rhd M : C@w \mid s$
  (2) $\Sigma \vdash \mathbb{W}; k \lhd \circ\, N : C@w$

  De (2) y por $C.Abs$ existen A, B, t tales que $C = A \supset B$ and

  (3) $\Sigma; \cdot; \cdot \rhd N : A@w \mid t$
  (4) $\Sigma \vdash \mathbb{W}; k : B@w$

  De (1) y (3) por $\supset E$:

  (5) $\Sigma; \cdot; \cdot \rhd M\, N : B@w \mid s \cdot t$

  De (4) y (5) por *MState*, $\Sigma \vdash \mathbb{W}; w : [k, M\, N]$. Como $\overline{M\, N} = \overline{M}\, \overline{N} = \overline{M'}\, \overline{N} = \overline{M'\, N}$, luego por HI, $F(\mathbb{W}; w : [k, M\, N]) = F(\mathbb{W}; w : [k, M'\, N]) = F(\mathbb{W}; w : [k \lhd \circ\, N, M'])$

- Caso $\langle |\mathbb{W}|, k \lhd V \circ\rangle$. $\mathbb{N} = \mathbb{W}; w : [k \lhd V \circ, N]$.
  $F(\mathbb{W}; w : [k \lhd V \circ, N]) = F(\mathbb{W}; w : [k, V\, N])$. Si $\Sigma \vdash \mathbb{W}; w : [k \lhd V \circ, N]$, luego por *MState* existen A y s tales que:

  (1) $\Sigma; \cdot; \cdot \rhd N : A@w \mid s$
  (2) $\Sigma \vdash \mathbb{W}; k \lhd V \circ : A@w$

De (2) por $C.App$ existen B y t tales que:

(3) $\Sigma; \cdot ; \cdot \rhd V : A \supset B@w \,|\, t$

(4) $\Sigma \vdash \mathbb{W} ; k : B@w$

De (1) y (3) por $\supset E$:

(5) $\Sigma; \cdot ; \cdot \rhd V\ N : B@w \,|\, s \cdot t$

De (4) y (5) por $MState$, $\Sigma \vdash \mathbb{W} ; w : [k, V\ N]$. Como $\overline{V\ N} = \overline{V}\ \overline{N} = \overline{V}\ \overline{N'} = \overline{V\ N'}$. Luego, por HI $F(\mathbb{W} ; w : [k, V\ N]) = F(\mathbb{W} ; w : [k, V\ N']) = F(\mathbb{W} ; w : [k \lhd V \circ, N'])$

- Caso $\langle |\mathbb{W}|, k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ M \rangle$. $\mathbb{N} = \mathbb{W} ; w : [k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N, M]$.

  $F(\mathbb{W} ; w : [k \lhd unpack \circ to\ \langle v^\bullet, v^\circ \rangle\ in\ N, M]) = F(\mathbb{W} ; w : [k, unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N])$.
  Si $\Sigma \vdash \mathbb{W} ; w : [k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N, M]$, luego por$MState$ existen D, s tales que:

  (1) $\Sigma; \cdot ; \cdot \rhd M : D@w \,|\, s$

  (2) $\Sigma \vdash \mathbb{W} ; k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N : D@w$

  De (2) por $C.Box$ existen A, C, t, r tales que $D = [r]A$ y

  (3) $\Sigma; v : A; \cdot \rhd N : C@w \,|\, t$

  (4) $\Sigma \vdash \mathbb{W} ; k : C\{v^\circ / r\}@w$

  De (1) y (3) por $\Box E$:

  (5) $\Sigma; \Delta; \Gamma \rhd unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N : C\{v^\circ / r\}@w \,|\, letc\ s\ be\ v : A\ in\ t$

  De (4) y (5) por $MState$, $\Sigma \vdash \mathbb{W} ; w : [k, unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N]$. Como $\overline{unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N} = unpack\ \overline{M}\ to\ \langle v^\bullet, v^\circ \rangle\ in\ \overline{N} = unpack\ \overline{M'}\ to\ \langle v^\bullet, v^\circ \rangle\ in\ \overline{N} = \overline{unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N'}$, luego por HI, $F(\mathbb{W} ; w : [k, unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N]) = F(\mathbb{W} ; w : [k, unpack\ M'\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N]) = F(\mathbb{W} ; w : [k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N, M'])$

- Caso $\langle |\mathbb{W}|, \text{return } w \rangle$. Como $M = \text{return } w$ y $\mathbb{N}$ es tipable, vale que $\mathbb{W} = \{w : C :: k_1; w_s\}$ para algún C, k_1, w_s. Por lo tanto, $\mathbb{N} = \{w : C :: k_1; w_s\} ; w' : [\text{return } w, M]$.

  $F(\{w : C :: k_1; w_s\} ; w' : [\text{return } w, M]) = F(\{w : C; w_s\} ; w : [k_1, M])$.
  Como $\Sigma \vdash \{w : C :: k_1; w_s\} ; w' : [\text{return } w, M]$, por $MState$ existen A y s tales que:

  (1) $\Sigma; \cdot ; \cdot \rhd M : A@w' \,|\, s$

  (2) $\Sigma \vdash \{w : C :: k_1; w_s\} ; \text{return } w : A@w'$

De (2) por $C.Return$:

(3) $\Sigma \vdash \{w : C; w_s\}\,; k_1 : A@w$

De (1) y por el lema de sustitución de mundos:

(4) $\Sigma; \cdot; \cdot \rhd M\{w'/w\} : A@w \mid s$

De (3) y (4) por *MState* $\Sigma \vdash \{w : C; w_s\}\,; w : [k_1, M\{w'/w\}]$ Como $|\{w : C; w_s\}| < |\{w : C :: k_1; w_s\}|$ y $\overline{M} = \overline{M\{w'/w\}} = \overline{M'}$ then, by HI, $F(\{w : C; w_s\}\,; w : [k_1, M]) = F(\{w : C; w_s\}\,; w : [k_1, M\{w'/w\}]) = F(\{w : C; w_s\}\,; w : [k_1, M']) = F(\{w : C :: k_1; w_s\}\,; w' : [\text{return } w, M'])$

□

Lema 3. Demostración por inducción sobre $\langle |\mathbb{W}|, k\rangle$.

- Caso $\langle |\mathbb{W}|, finish\rangle$. $\mathbb{N} = \mathbb{W}\,; w : [finish, M]$.

  $F(\mathbb{W}\,; w : [finish, M]) = \overline{M}$. Si $\Sigma \vdash \mathbb{W}\,; w : [finish, M]$, entonces por *MState* existen A y s tales que $\Sigma; \cdot; \cdot \rhd M : A@w \mid s$ es derivable. Luego, por el lema 11, $\Sigma; \cdot; \cdot \rhd \overline{M} : A@\bullet \mid s$

- Caso $\langle |\mathbb{W}|, k \lhd \circ N\rangle$. $\mathbb{N} = \mathbb{W}\,; w : [k \lhd \circ N, M]$.

  $F(\mathbb{W}\,; w : [k \lhd \circ N, M]) = F(\mathbb{W}\,; w : [k, M\,N])$. Si $\Sigma \vdash \mathbb{W}\,; w : [k \lhd \circ N, M]$, entonces por *MState* existen C, s tales que:

  (1) $\Sigma; \cdot; \cdot \rhd M : C@w \mid s$

  (2) $\Sigma \vdash \mathbb{W}\,; k \lhd \circ N : C@w$

  De (2) por $C.Abs$ existen A, B, t tales que $C = A \supset B$ y

  (3) $\Sigma; \cdot; \cdot \rhd N : A@w \mid t$

  (4) $\Sigma \vdash \mathbb{W}\,; k : B@w$

  De (1) y (3) por $\supset E$:

  (5) $\Sigma; \cdot; \cdot \rhd M\,N : B@w \mid s \cdot t$

  De (4) y (5) por *MState*, $\Sigma \vdash \mathbb{W}\,; w : [k, M\,N]$. Luego por HI, existen A', s' tales que $\Sigma; \cdot; \cdot \rhd F(\mathbb{W}\,; w : [k, M\,N]) : A'@\bullet \mid s'$.

- Caso $\langle |\mathbb{W}|, k \lhd V \circ\rangle$. $\mathbb{N} = \mathbb{W}\,; w : [k \lhd V \circ, N]$.

  $F(\mathbb{W}\,; w : [k \lhd V \circ, N]) = F(\mathbb{W}\,; w : [k, V\,N])$.Si $\Sigma \vdash \mathbb{W}\,; w : [k \lhd V \circ, N]$, entonces por *MState* existen A y s tales que:

  (1) $\Sigma; \cdot; \cdot \rhd N : A@w \mid s$

  (2) $\Sigma \vdash \mathbb{W}\,; k \lhd V \circ : A@w$

De (2) por $C.App$ existen B y t tales que:

(3) $\Sigma; \cdot ; \cdot \rhd V : A \supset B@w \,|\, t$

(4) $\Sigma \vdash \mathbb{W} ; k : B@w$

De (1) y (3) por $\supset E$:

(5) $\Sigma; \cdot ; \cdot \rhd V\ N : B@w \,|\, s \cdot t$

De (4) y (5) por $MState$, $\Sigma \vdash \mathbb{W} ; w : [k, V\ N]$. Luego por HI, existen A', s' tales que $\Sigma; \cdot; \cdot \rhd F(\mathbb{W} ; w : [k, V\ N]) : A'@\bullet \mid s'$.

- Caso $\langle |\mathbb{W}|, k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ M \rangle$. $\mathbb{N} = \mathbb{W} ; w : [k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N, M]$.

  $F(\mathbb{W} ; w : [k \lhd unpack \circ to\ \langle v^\bullet, v^\circ \rangle\ in\ N, M]) = F(\mathbb{W} ; w : [k, unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N])$. Si $\Sigma \vdash \mathbb{W} ; w : [k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N, M]$, entonces por $MState$ existen D, s tales que:

  (1) $\Sigma; \cdot ; \cdot \rhd M : D@w \,|\, s$

  (2) $\Sigma \vdash \mathbb{W} ; k \lhd unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N : D@w$

  De (2) por $C.Box$ existen A, C, t, r tales que $D = [r]A$ y, además,

  (3) $\Sigma; v : A; \cdot \rhd N : C@w \,|\, t$

  (4) $\Sigma \vdash \mathbb{W} ; k : C\{v^\circ / r\}@w$

  De (1) y (3) por $\Box E$:

  (5) $\Sigma; \Delta; \Gamma \rhd unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N : C\{v^\circ / r\}@w \,|\, letc\ s\ be\ v : A\ in\ t$

  De (4) y (5) por $MState$, $\Sigma \vdash \mathbb{W} ; w : [k, unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N]$. Luego por HI, existen A', s' tales que $\Sigma; \cdot; \cdot \rhd F(\mathbb{W} ; w : [k, unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N]) : A'@\bullet \mid s'$.

- Caso $\langle |\mathbb{W}|, \text{return } w \rangle$. Como $k = \text{return } w$ y $\mathbb{N}$ es tipable, vale que $\mathbb{W} = \{w : C :: k_1; w_s\}$ para algún C, k_1, w_s. Por lo tanto, $\mathbb{N} = \{w : C :: k_1; w_s\} ; w' : [\text{return } w, M]$.

  $F(\{w : C :: k_1; w_s\} ; w' : [\text{return } w, M]) = F(\{w : C; w_s\} ; w : [k_1, M\{w'/w\}])$. Como $\Sigma \vdash \{w : C :: k_1; w_s\} ; w' : [\text{return } w, M]$, por $MState$ existen A y s tales que:

  (1) $\Sigma; \cdot ; \cdot \rhd M : A@w' \,|\, s$

  (2) $\Sigma \vdash \{w : C :: k_1; w_s\} ; \text{return } w : A@w'$

  De (2) por $C.Return$:

  (3) $\Sigma \vdash \{w : C; w_s\} ; k_1 : A@w$

De (1) y por el lema de sustitución de mundos (1)

(4) $\Sigma; \cdot; \cdot \rhd M\{w'/w\} : A@w \mid s$

De (3) y (4) por *MState* $\Sigma \vdash \{w : C; w_s\}; w : [k_1, M\{w'/w\}]$. Como $|\{w : C; w_s\}| < |\{w : C :: k_1; w_s\}|$ por HI existen A' s' tales que

(5) $\Sigma; \cdot; \cdot \rhd F(\{w : C; w_s\}; w : [k_1, M\{w'/w\}]) : A'@ \bullet \mid s'$.

De (5) y por el lema 13

(6) $\Sigma; \cdot; \cdot \rhd F(\{w : C; w_s\}; w : [k_1, M]) : A'@ \bullet \mid s'$.

□

Lema 14. $\overline{M}\{a/\overline{N}\} = \overline{M\{a/N\}}$

Demostración. Por inducción sobre M.

- Caso a

$$\overline{a}\{a/\overline{N}\} = a\{a/\overline{N}\} = \overline{N} = \overline{a\{a/N\}}$$

- Caso $b \neq a$

$$\overline{b}\{a/\overline{N}\} = b\{a/\overline{N}\} = b = \overline{b} = \overline{b\{a/N\}}$$

- Caso $v^\bullet$

$$\overline{v^\bullet}\{a/\overline{N}\} = v^\bullet\{a/\overline{N}\} = v^\bullet = \overline{v^\bullet} = \overline{v^\bullet\{a/N\}}$$

- Caso $M\, M'$

$$\begin{array}{lll} & \overline{(M\, M')}\{a/\overline{N}\} & \\ = & (\overline{M}\, \overline{M'})\{a/\overline{N}\} & \\ = & \overline{M}\{a/\overline{N}\}\, \overline{M'}\{a/\overline{N}\} & \text{HI} \\ = & \overline{M\{a/N\}}\, \overline{M'\{a/N\}} & \\ = & \overline{M\{a/N\}\, M'\{a/N\}} & \\ = & \overline{(M\, M')\{a/N\}} & \end{array}$$

- Caso *unpack* M *to* $\langle v^\bullet, v^\circ \rangle$ *in* M'

$$\begin{array}{lll} & \overline{(unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ M')}\{a/\overline{N}\} & \\ = & (unpack\ \overline{M}\ to\ \langle v^\bullet, v^\circ \rangle\ in\ \overline{M'})\{a/\overline{N}\} & \\ = & unpack\ \overline{M}\{a/\overline{N}\}\ to\ \langle v^\bullet, v^\circ \rangle\ in\ \overline{M'}\{a/\overline{N}\} & \text{HI} \\ = & unpack\ \overline{M\{a/N\}}\ to\ \langle v^\bullet, v^\circ \rangle\ in\ \overline{M'\{a/N\}} & \\ = & \overline{unpack\ M\{a/N\}\ to\ \langle v^\bullet, v^\circ \rangle\ in\ M'\{a/N\}} & \\ = & \overline{(unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ M')\{a/N\}} & \end{array}$$

- Caso $\lambda b.M$

$$\begin{array}{lll} & \overline{(\lambda b.M)}\{a/\overline{N}\} & \\ = & (\lambda b.\overline{M})\{a/\overline{N}\} & \\ = & \lambda b.\overline{M}\{a/\overline{N}\} & \text{HI} \\ = & \lambda b.\overline{M\{a/N\}} & \\ = & \overline{\lambda b.M\{a/N\}} & \\ = & \overline{(\lambda b.M)\{a/N\}} & \end{array}$$

- Caso $box_t\, M$

$$\begin{array}{lll} & \overline{(box_t\, M)}\{a/\overline{N}\} & \\ = & (box_t\, \overline{M})\{a/\overline{N}\} & \\ = & box_t\, \overline{M}\{a/\overline{N}\} & \text{HI} \\ = & box_t\, \overline{M\{a/N\}} & \\ = & \overline{box_t\, M\{a/N\}} & \\ = & \overline{(box_t\, M)\{a/N\}} & \end{array}$$

- Caso $fetch[w]\, M$

$$\begin{array}{lll} & \overline{(fetch[w]\, M)}\{a/\overline{N}\} & \\ = & (fetch[\bullet]\, \overline{M})\{a/\overline{N}\} & \\ = & fetch[\bullet]\, \overline{M}\{a/\overline{N}\} & \text{HI} \\ = & fetch[\bullet]\, \overline{M\{a/N\}} & \\ = & \overline{fetch[w]\, M\{a/N\}} & \\ = & \overline{(fetch[w]\, M)\{a/N\}} & \end{array}$$

□

Lema 15. $\overline{N}\{v^\bullet/\overline{M}\}\{v^\circ/s\} = \overline{N\{v^\bullet/M\}\{v^\circ/s\}}$

Demostración. Por inducción sobre M.

- Caso a

$$\overline{a}\{v^\bullet/\overline{M}\}\{v^\circ/s\} = a\{v^\bullet/\overline{M}\}\{v^\circ/s\} = a = \overline{a} = \overline{a\{v^\bullet/M\}\{v^\circ/s\}}$$

- Caso $v^\bullet$

$$\overline{v^\bullet}\{v^\bullet/\overline{M}\}\{v^\circ/s\} = v^\bullet\{v^\bullet/\overline{M}\}\{v^\circ/s\} = \overline{M} = \overline{v^\bullet\{v^\bullet/M\}\{v^\circ/s\}}$$

- Caso $u^\bullet \neq v^\bullet$

$$\overline{u^\bullet}\{v^\bullet/\overline{M}\}\{v^\circ/s\} = u^\bullet\{v^\bullet/\overline{M}\}\{v^\circ/s\} = u^\bullet = \overline{u^\bullet} = \overline{u^\bullet\{v^\bullet/M\}\{v^\circ/s\}}$$

- Caso $N\, N'$

$$\begin{array}{lll} & \overline{(N\,N')}\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \\ = & (\overline{N}\,\overline{N'})\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \\ = & \overline{N}\{v^\bullet/\overline{M}\}\{v^\circ/s\}\,\overline{N'}\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \text{HI} \\ = & \overline{N\{v^\bullet/M\}\{v^\circ/s\}\,N'\{v^\bullet/M\}\{v^\circ/s\}} & \\ = & \overline{N\{v^\bullet/M\}\{v^\circ/s\}\,N'\{v^\bullet/M\}\{v^\circ/s\}} & \\ = & \overline{(N\,N')\{v^\bullet/M\}\{v^\circ/s\}} & \end{array}$$

- Caso $unpack\ N\ to\ \langle u^\bullet, u^\circ\rangle\ in\ N'$

$$\begin{array}{lll} & \overline{(unpack\ N\ to\ \langle u^\bullet, u^\circ\rangle\ in\ N')}\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \\ = & (unpack\ \overline{N}\ to\ \langle u^\bullet, u^\circ\rangle\ in\ \overline{N'})\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \\ = & unpack\ \overline{N}\{v^\bullet/\overline{M}\}\{v^\circ/s\}\ to\ \langle u^\bullet, u^\circ\rangle\ in\ \overline{N'}\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \text{HI} \\ = & \overline{unpack\ N\{v^\bullet/M\}\{v^\circ/s\}\ to\ \langle u^\bullet, u^\circ\rangle\ in\ N'\{v^\bullet/M\}\{v^\circ/s\}} & \\ = & \overline{unpack\ N\{v^\bullet/M\}\{v^\circ/s\}\ to\ \langle u^\bullet, u^\circ\rangle\ in\ N'\{v^\bullet/M\}\{v^\circ/s\}} & \\ = & \overline{(unpack\ N\ to\ \langle u^\bullet, u^\circ\rangle\ in\ N')\{v^\bullet/M\}\{v^\circ/s\}} & \end{array}$$

- Caso $\lambda b.N$

$$\begin{array}{lll} = & \overline{(\lambda b.N)}\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \\ = & (\lambda b.\overline{N})\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \\ = & \lambda b.\overline{N}\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \text{HI} \\ = & \overline{\lambda b.N\{v^\bullet/M\}\{v^\circ/s\}} & \\ = & \overline{\lambda b.N\{v^\bullet/M\}\{v^\circ/s\}} & \\ = & \overline{(\lambda b.N)\{v^\bullet/M\}\{v^\circ/s\}} & \end{array}$$

- Caso $box_t\,N$

$$\begin{array}{lll} = & \overline{(box_t\,N)}\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \\ = & (box_t\,\overline{N})\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \\ = & box_{t\{v^\circ/s\}}\,\overline{N}\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \text{HI} \\ = & \overline{box_{t\{v^\circ/s\}}\,N\{v^\bullet/M\}\{v^\circ/s\}} & \\ = & \overline{box_{t\{v^\circ/s\}}\,N\{v^\bullet/M\}\{v^\circ/s\}} & \\ = & \overline{(box_t\,N)\{v^\bullet/M\}\{v^\circ/s\}} & \end{array}$$

- Caso $fetch[w]\,N$

$$\begin{array}{lll} = & \overline{(fetch[w]\,N)}\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \\ = & (fetch[\bullet]\,\overline{N})\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \\ = & fetch[\bullet]\,\overline{N}\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \text{HI} \\ = & \overline{fetch[\bullet]\,N\{v^\bullet/M\}\{v^\circ/s\}} & \\ = & \overline{fetch[w]\,N\{v^\bullet/M\}\{v^\circ/s\}} & \\ = & \overline{(fetch[w]\,N)\{v^\bullet/M\}\{v^\circ/s\}} & \end{array}$$

□

Lema 16. *Si* $M \longrightarrow_{\beta,\beta_\Box,ftch} M'$ *entonces* $\overline{M} \longrightarrow_{\beta,\beta_\Box,ftch} \overline{M'}$

Demostración. Por inducción sobre M. Los casos base son triviales ya que no se generar pasos de reducción desde a o $v^\bullet$. Se muestran los casos donde la reducción ocurre en la raíz de M.

- Caso $\longrightarrow_\beta$. Supongamos que $(\lambda a.M)\, N \longrightarrow_\beta M\{a/N\}$. Luego:

$$\begin{array}{rll} & \overline{(\lambda a.M)\, N} & \\ = & \overline{\lambda a.M}\,\overline{N} & \\ = & (\lambda a.\overline{M})\,\overline{N} & \\ \longrightarrow_\beta & \overline{M}\{a/\overline{N}\} & \\ = & \overline{M\{a/N\}} & \text{(Lema 14)} \end{array}$$

- Caso $\longrightarrow_{\beta_\Box}$. Supongamos que $unpack\ box_s\, M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N \longrightarrow_{\beta_\Box} N\{v^\bullet/M\}\{v^\circ/s\}$. Luego:

$$\begin{array}{rll} & \overline{unpack\ box_s\, M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N} & \\ = & unpack\ \overline{box_s\, M}\ to\ \langle v^\bullet, v^\circ\rangle\ in\ \overline{N} & \\ = & unpack\ box_s\, \overline{M}\ to\ \langle v^\bullet, v^\circ\rangle\ in\ \overline{N} & \\ \longrightarrow_\beta & \overline{N}\{v^\bullet/\overline{M}\}\{v^\circ/s\} & \\ = & \overline{N\{v^\bullet/M\}\{v^\circ/s\}} & \text{(Lema 15)} \end{array}$$

- Caso $\longrightarrow_{ftch}$. Supongamos que $fetch[w]\, M \longrightarrow_{ftcs} M$. Luego:

$$\begin{array}{rl} & \overline{fetch[w]\, M} \\ = & fetch[\bullet]\,\overline{M} \\ \longrightarrow_\beta & \overline{M} \end{array}$$

□

Lema 17. *Si* $M \longrightarrow_{\beta,\beta_\Box,ftch} M'$ *y* $\Sigma \vdash \mathbb{W}\,; w : [k, M]$ *es derivable, entonces* $F(\Sigma \vdash \mathbb{W}\,; w : [k, M]) \longrightarrow_{\beta,\beta_\Box,ftch} F(\Sigma \vdash \mathbb{W}\,; w : [k, M'])$.

Demostración. Por inducción sobre $\langle |\mathbb{W}|, k\rangle$.

- Caso $\langle |\mathbb{W}|, finish\rangle$.

  Por el lema 16, $\overline{M} \longrightarrow_{\beta,\beta_\Box,ftch} \overline{M'}$. Luego,

$$F(\Sigma \vdash \mathbb{W}\,; w : [finish, M]) = \overline{M} \longrightarrow_{\beta,\beta_\Box,ftch} \overline{M'} = F(\Sigma \vdash \mathbb{W}\,; w : [finish, M'])$$

- Caso $\langle |\mathbb{W}|, k \lhd \circ\, N\rangle$.

$$F(\Sigma \vdash \mathbb{W}\,; w : [k \lhd \circ\, N, M]) = F(\Sigma \vdash \mathbb{W}\,; w : [k, M\, N])$$

Si $M \longrightarrow_{\beta,\beta_\square,ftch} M'$, entonces $M\,N \longrightarrow_{\beta,\beta_\square,ftch} M'\,N$. Luego por HI:

$$F(\Sigma \vdash \mathbb{W}\,; w : [k, M\,N]) \longrightarrow_{\beta,\beta_\square,ftch} F(\Sigma \vdash \mathbb{W}\,; w : [k, M'\,N]) = F(\Sigma \vdash \mathbb{W}\,; w : [k \lhd \circ\, N, M'])$$

- Caso $\langle |\mathbb{W}|, k \lhd V \circ \rangle$.

$$F(\Sigma \vdash \mathbb{W}\,; w : [k \lhd V \circ, M]) = F(\Sigma \vdash \mathbb{W}\,; w : [k, V\,M])$$

Si $M \longrightarrow_{\beta,\beta_\square,ftch} M'$ entonces $V\,M \longrightarrow_{\beta,\beta_\square,ftch} V\,M'$. Luego por HI

$$F(\Sigma \vdash \mathbb{W}\,; w : [k, V\,M]) \longrightarrow_{\beta,\beta_\square,ftch} F(\Sigma \vdash \mathbb{W}\,; w : [k, V\,M']) = F(\Sigma \vdash \mathbb{W}\,; w : [k \lhd V \circ, M'])$$

- Caso $\langle |\mathbb{W}|, k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N \rangle$.

$$F(\Sigma \vdash \mathbb{W}\,; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N, M]) = F(\Sigma \vdash \mathbb{W}\,; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N])$$

Si $M \longrightarrow_{\beta,\beta_\square,ftch} M'$, entonces $\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N \longrightarrow_{\beta,\beta_\square,ftch} \mathit{unpack}\ M'\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N$. Luego por HI:

$$F(\Sigma \vdash \mathbb{W}\,; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N]) \longrightarrow_{\beta,\beta_\square,ftch} F(\Sigma \vdash \mathbb{W}\,; w : [k, \mathit{unpack}\ M'\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N]) = F(\Sigma \vdash \mathbb{W}\,; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N, M'])$$

- Caso $\langle |\mathbb{W}|, \mathrm{return}\, w \rangle$. Como el estado de la máquina está bien tipado, vale que $\mathbb{W} = \{w : C :: k_1; w_s\}$, para algún C, k_1 y w_s.

$$F(\Sigma \vdash \{w : C :: k_1; w_s\}\,; w' : [\mathrm{return}\, w, M]) = F(\Sigma \vdash \{w : C; w_s\}\,; w : [k_1, M])$$

Luego, como $|\{w : C; w_s\}| < |\{w : C :: k_1; w_s\}|$, por HI:

$$F(\Sigma \vdash \{w : C; w_s\}\,; w : [k_1, M]) \longrightarrow_{\beta,\beta_\square,ftch} F(\Sigma \vdash \{w : C; w_s\}\,; w : [k_1, M']) = F(\Sigma \vdash \{w : C :: k_1; w_s\}\,; w' : [\mathrm{return}\, w, M'])$$

□

Lema 4. Se demuestran ambos items por análisis de casos sobre la regla de reducción

- Caso (1): $\mathbb{W}; w : [k, MN] \longrightarrow \mathbb{W}; w : [k \lhd \circ\ N, M]$

$$F(\mathbb{W}; w : [k, MN]) = F(\mathbb{W}; w : [k \lhd \circ\ N, M])$$

- Caso (2.1): $\mathbb{W}; w : [k \lhd \circ\ N, V] \longrightarrow \mathbb{W}; w : [k \lhd V\ \circ, N]$, N no es un valor

$$F(\mathbb{W}; w : [k \lhd \circ\ N, V]) = F(\mathbb{W}; w : [k, V\ N]) = F(\mathbb{W}; w : [k \lhd V\ \circ, N])$$

- Caso (4): $\mathbb{W}; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N] \longrightarrow \mathbb{W}; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, M]$

$$F(\mathbb{W}; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N]) = F(\mathbb{W}; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, M])$$

- Caso (7): $\{w : C :: k; w_s\}; w' : [\text{return } w, V] \longrightarrow \{w : C; w_s\}; w : [k, V\{w'/w\}]$

$$\begin{array}{lcl} F(\{w : C :: k; w_s\}; w' : [\text{return } w, V]) & = & \\ F(\{w : C; w_s\}; w : [k, V]) & = & \text{by lema 13} \\ F(\{w : C; w_s\}; w : [k, V\{w'/w\}]) & = & \end{array}$$

- Caso (2.2): $\mathbb{W}; w : [k \lhd \circ\ V, \lambda a.M] \longrightarrow \mathbb{W}; w : [k, M\{a/V\}]$

$$F(\mathbb{W}; w : [k \lhd \circ\ V, \lambda a.M]) = F(\mathbb{W}; w : [k, (\lambda a.M)\, V])$$

Como $(\lambda a.M)\, V \longrightarrow_\beta M\{a/V\}$, luego por el lema 17:

$$F(\mathbb{W}; w : [k, (\lambda a.M)\, V]) \longrightarrow_{\beta,\beta_\Box,ftch} F(\mathbb{W}; w : [k, M\{a/V\}])$$

- Caso (3) $\mathbb{W}; w : [k \lhd (\lambda a.M)\ \circ, V] \longrightarrow \mathbb{W}; w : [k, M\{a/V\}]$

$$F(\mathbb{W}; w : [k \lhd (\lambda a.M)\ \circ, V]) = F(\mathbb{W}; w : [k, (\lambda a.M)V])$$

Como $(\lambda a.M)\, V \longrightarrow_\beta M\{a/V\}$, luego por el lema 17:

$$F(\mathbb{W}; w : [k, (\lambda a.M)\, V]) \longrightarrow_{\beta,\beta_\Box,ftch} F(\mathbb{W}; w : [k, M\{a/V\}])$$

- Caso (5): $\mathbb{W}; w : [k \lhd \mathit{unpack} \circ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, \mathit{box}_s\, M] \longrightarrow \mathbb{W}; w : [k, N\{v^\circ/s\}\{v^\bullet/M\}]$

$$F(\mathbb{W}; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, \mathit{box}_s\, M]) = F(\mathbb{W}; w : [k, \mathit{unpack}\ \mathit{box}_s\, M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N])$$

Como $\mathit{unpack}\ \mathit{box}_s\, M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N \longrightarrow_{\beta_\Box} N\{v^\circ/s\}\{v^\bullet/M\}$, luego por el lema 17:

$$F(\mathbb{W}; w : [k, \mathit{unpack}\ \mathit{box}_s\, M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N]) \longrightarrow_{\beta,\beta_\Box,ftch} F(\mathbb{W}; w : [k, N\{v^\circ/s\}\{v^\bullet/M\}])$$

- Caso (6): $\{w : C; w_s\}; w : [k, \mathit{fetch}[w']\, M] \longrightarrow \{w : C :: k; w_s\}; w' : [\text{return}\, w, M]$

  $$F(\{w : C :: k; w_s\}; w' : [\text{return}\ w, M]) = F(\{w : C; w_s\}; w : [k, M])$$

  Ya que $\mathit{fetch}[w']\, M \longrightarrow_{\mathit{ftch}} M$, por el lema 17:

  $$F(\{w : C; w_s\}; w : [k, \mathit{fetch}[w']\, M]) \longrightarrow_{\beta,\beta_\Box,\mathit{ftch}} F(\{w : C; w_s\}; w : [k, M])$$

□

Prueba de Lem. 5.

Demostración. Por inducción en la derivación de $\Sigma; \Delta; \Gamma \rhd M : A@w \,|\, s$.

- Caso VarT.

  $$\frac{\Sigma \vdash w}{\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd a : A@w \,|\, a}\ \text{VarT}$$

  Se obtiene

  $$\frac{}{\Delta', \Gamma'_1, a : T(A), \Gamma'_2 \rhd a : T(A)}$$

- Caso $\supset I$.

  $$\frac{\Sigma; \Delta; \Gamma, a : A@w \rhd M : B@w \,|\, s}{\Sigma; \Delta; \Gamma \rhd \lambda a.M : A \supset B@w \,|\, \lambda a : A.s}\ \supset I$$

  Luego

  $$\frac{\Delta', \Gamma', a : T(A) \rhd T(M) : T(B)}{\Delta', \Gamma' \rhd \lambda a.T(M) : T(A) \supset T(B)}\ \supset I$$

- Caso $\supset E$.

  $$\frac{\Sigma; \Delta; \Gamma \rhd M : A \supset B@w \,|\, s \quad \Sigma; \Delta; \Gamma \rhd N : A@w \,|\, t}{\Sigma; \Delta; \Gamma \rhd M\, N : B@w \,|\, s \cdot t}\ \supset E$$

  Luego por HI se tiene que $\Delta', \Gamma' \rhd T(M) : T(A \supset B)$ es derivable y que además $T(A \supset B) = T(A) \supset T(B)$. Por lo tanto:

  $$\frac{\Delta', \Gamma' \rhd T(M) : T(A) \supset T(B) \quad \Delta', \Gamma' \rhd T(N) : T(A)}{\Delta', \Gamma' \rhd T(M)\ T(N) : T(B)}\ \supset E$$

- Case VarV.

$$\frac{\Sigma \vdash w}{\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd v^\bullet : A@w \mid v^\circ}\ \text{VarV}$$

Luego es fácil ver que $\Delta_1', v : \mathbf{1} \supset T(A), \Delta_2', \Gamma' \rhd v\, unit : T(A)$ es derivable.

- Caso $\Box I$.

$$\frac{\Sigma; \Delta; \cdot \rhd M : A@w \mid s}{\Sigma; \Delta; \Gamma \rhd box_s\, M : [s]A@w \mid !s}\ \Box I$$

Por la HI se tiene que $\Delta' \rhd T(M) : T(A)$ es derivable. Por weakening en $\lambda^{\mathbf{1},\to}$ se tiene que $\Delta', a : \mathbf{1} \rhd T(M) : T(A)$. Luego $\Delta' \rhd \lambda a.T(M) : \mathbf{1} \supset T(A)$ es derivable.

- Caso Fetch.

$$\frac{\Sigma; \Delta; \Gamma \rhd M : [s]A@w' \mid t \quad \Sigma \vdash w}{\Sigma; \Delta; \Gamma \rhd fetch[w']\, M : [s]A@w \mid fetch(t)}\ \text{Fetch}$$

Luego

$$\frac{\Delta', \Gamma' \rhd \lambda a.a : (\mathbf{1} \supset T(A)) \supset (\mathbf{1} \supset T(A)) \quad \Delta', \Gamma' \rhd T(M) : \mathbf{1} \supset T(A)}{\Delta', \Gamma' \rhd (\lambda a.a)\ T(M) : \mathbf{1} \supset T(A)}\ \supset E$$

- Caso $\Box E$.

$$\frac{\Sigma; \Delta; \Gamma \rhd M : [r]A@w \mid s \quad \Sigma; \Delta, v : A@w; \Gamma \rhd N : C@w \mid t}{\Sigma; \Delta; \Gamma \rhd unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N : C\{v^\circ/r\}@w \mid letc\ s\ be\ v : A\ in\ t}\ \Box E$$

Por la HI vale que

- $\Delta', \Gamma' \rhd T(M) : \mathbf{1} \supset T(A)$
- $\Delta', v : \mathbf{1} \supset T(A), \Gamma' \rhd T(N) : T(C)$

Teniendo en cuenta que $T(C) = T(C\{v^\circ/r\})$

$$\frac{\dfrac{\Delta', v : \mathbf{1} \supset T(A), \Gamma' \rhd T(N) : T(C)}{\Delta', \Gamma' \rhd \lambda v.T(N) : (\mathbf{1} \supset T(A)) \supset T(C)} \qquad \Delta', \Gamma' \rhd T(M) : \mathbf{1} \supset T(A)}{\Delta', \Gamma' \rhd (\lambda v.T(N))\ T(M) : T(C)}\ \supset E$$

□

Lema 6. Por inducción sobre M.

- Caso a.

$$T(a)\{a/T(N)\} = a\{a/T(N)\} = T(N) = T(a\{a/N\})$$

- Caso $b \neq a$,

$$T(b)\{a/T(N)\} = b\{a/T(N)\} = b = T(b) = T(b\{a/N\})$$

- Caso $v^\bullet$.

$$\begin{array}{rl} & T(v^\bullet)\{a/T(N)\} \\ = & (v\,unit)\{a/T(N)\} \\ = & v\,unit \\ = & T(v^\bullet) \\ = & T(v^\bullet\{a/N\}) \end{array}$$

- Caso $M\,M'$.

$$\begin{array}{rll} & T(M\,M')\{a/T(N)\} & \\ = & (T(M)\,T(M'))\{a/T(N)\} & \\ = & T(M)\{a/T(N)\}\,T(M')\{a/T(N)\} & \\ = & T(M\{a/N\}\,M'\{a/N\}) & \text{HI} \\ = & T((M\,M')\{a/N\}) & \end{array}$$

- Caso $\lambda b.M$

$$\begin{array}{rll} & T(\lambda b.M)\{a/T(N)\} & \\ = & (\lambda b.T(M))\{a/T(N)\} & \\ = & \lambda b.T(M\{a/T(N)\}) & \text{HI} \\ = & \lambda b.T(M\{a/N\}) & \\ = & T(\lambda b.M\{a/N\}) & \\ = & T((\lambda b.M)\{a/N\}) & \end{array}$$

- Caso $box_t\,M$

$$\begin{array}{rll} & T(box_t\,M)\{a/T(N)\} & b\,\text{fresh} \\ = & (\lambda b.T(M))\{a/T(N)\} & \\ = & \lambda b.T(M\{a/T(N)\}) & \text{HI} \\ = & \lambda b.T(M\{a/N\}) & \\ = & T(box_t\,M\{a/N\}) & \\ = & T((box_t\,M)\{a/N\}) & \end{array}$$

- Caso $fetch[w]\,M$

$$\begin{array}{lll} & T(fetch[w]\,M)\{a/T(N)\} & \\ = & ((\lambda b.b)\,T(M))\{a/T(N)\} & \\ = & (\lambda b.b)\{a/T(N)\}\,T(M)\{a/T(N)\} & \text{HI} \\ = & (\lambda b.b)\,T(M\{a/N\}) & \\ = & T(fetch[w]\,M\{a/N\}) & \\ = & T((fetch[w]\,M)\{a/N\}) & \end{array}$$

- Caso *unpack* M *to* $\langle v^\bullet, v^\circ\rangle$ *in* M'.

$$\begin{array}{lll} & T(unpack\ M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ M')\{a/T(N)\} & \\ = & ((\lambda v.T(M'))T(M))\{a/T(N)\} & \\ = & (\lambda v.T(M'))\{a/T(N)\}T(M)\{a/T(N)\} & \\ = & (\lambda v.T(M)'\{a/T(N)\})T(M)\{a/T(N)\} & \text{HI} \\ = & (\lambda v.T(M'\{a/N\}))T(M\{a/N\}) & \\ = & T(unpack\ M\{a/N\}\ to\ \langle v^\bullet, v^\circ\rangle\ in\ M'\{a/N\}) & \\ = & T((unpack\ M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ M')\{a/N\}) & \end{array}$$

□

Lema 7. Por inducción sobre M.

- Caso b.

$$T(b)\{v/\lambda a.T(N)\} = b\{v/\lambda a.T(N)\} = b = T(b) = T(b\{v^\bullet/N\}\{v^\circ/s\})$$

- Caso $v^\bullet$.

$$T(v^\bullet)\{v/\lambda a.T(N)\} = (v\,unit)\{v/\lambda a.T(N)\} = (\lambda a.T(N))\,unit \longrightarrow_\beta T(N) = T(v^\bullet\{v^\bullet/N\}\{v^\circ/s\})$$

- Caso $u^\bullet \neq v^\bullet$.

$$T(u^\bullet)\{v/\lambda a.T(N)\} = (u\,unit)\{v/\lambda a.T(N)\} = u\,unit = T(u^\bullet) = T(u^\bullet\{v^\bullet/N\}\{v^\circ/s\})$$

- Caso $M\,M'$.

$$T(M\,M')\{v/\lambda a.T(N)\} = (T(M)\,T(M'))\{v/\lambda a.T(N)\} = T(M)\{v/\lambda a.T(N)\}\,T(M')\{v/\lambda a.T(N)\}$$

Por HI,

$$\begin{array}{c} T(M)\{v/\lambda a.T(N)\} \longrightarrow^*_\beta T(M\{v^\bullet/N\}\{v^\circ/s\}) \\ T(M')\{v/\lambda a.T(N)\} \longrightarrow^*_\beta T(M'\{v^\bullet/N\}\{v^\circ/s\}) \end{array}$$

Luego,

$$T(M)\{v/\lambda a.T(N)\}\,T(M')\{v/\lambda a.T(N)\} \longrightarrow_\beta^* \\ T(M\{v^\bullet/N\}\{v^\circ/s\})\,T(M'\{v^\bullet/N\}\{v^\circ/s\}) = \\ T(M\{v^\bullet/N\}\{v^\circ/s\}\,M'\{v^\bullet/N\}\{v^\circ/s\}) = T((M\,M')\{v^\bullet/N\}\{v^\circ/s\})$$

- Caso $\lambda b.M$ (notar que $a \neq b$ ya a es fresca).

  $$T(\lambda b.M)\{v/\lambda a.T(N)\} = (\lambda b.T(M))\{v/\lambda a.T(N)\} = \lambda b.T(M)\{v/\lambda a.T(N)\}$$

  Por HI,

  $$T(M)\{v/\lambda a.T(N)\} \longrightarrow_\beta^* T(M\{v^\bullet/N\}\{v^\circ/s\})$$

  Luego,

  $$\lambda b.T(M\{v^\bullet/N\}\{v^\circ/s\}) \longrightarrow_\beta^* \lambda b.T(M\{v^\bullet/N\}\{v^\circ/s\}) = \\ T(\lambda b.M\{v^\bullet/N\}\{v^\circ/s\}) = T((\lambda b.M)\{v^\bullet/N\}\{v^\circ/s\})$$

- Caso $box_t\,M$.

  $$T(box_t\,M)\{v/\lambda a.T(N)\} = (\lambda b.T(M))\{v/\lambda a.T(N)\} = \lambda b.T(M)\{v/\lambda a.T(N)\}$$

  Por HI,

  $$T(M)\{v/\lambda a.T(N)\} \longrightarrow_\beta^* T(M\{v^\bullet/N\}\{v^\circ/s\})$$

  Luego,

  $$\lambda b.T(M)\{v/\lambda a.T(N)\} \longrightarrow_\beta^* \lambda b.T(M\{v^\bullet/N\}\{v^\circ/s\}) = \\ T(box_{t\{s/v^\circ\}}\,M\{v^\bullet/N\}\{v^\circ/s\}) = T((box_t\,M)\{v^\bullet/N\}\{v^\circ/s\})$$

- Caso $unpack\ M\ to\ \langle u^\bullet, u^\circ\rangle\ in\ M'$.

  $$T(unpack\ M\ to\ \langle u^\bullet, u^\circ\rangle\ in\ M')\{v/\lambda a.T(N)\} = \\ ((\lambda u.T(M'))\,T(M))\{v/\lambda a.T(N)\} = \\ ((\lambda u.T(M')\{v/\lambda a.T(N)\})\,T(M)\{v/\lambda a.T(N)\})$$

  Por HI,

  $$T(M)\{v/\lambda a.T(N)\} \longrightarrow_\beta^* T(M\{v^\bullet/N\}\{v^\circ/s\}) \\ T(M')\{v/\lambda a.T(N)\} \longrightarrow_\beta^* T(M'\{v^\bullet/N\}\{v^\circ/s\})$$

  Luego,

  $$((\lambda u.T(M')\{v/\lambda a.T(N)\})\,\mathrm{T(M)}\{v/\lambda a.T(N)\}) \longrightarrow_\beta^* \\ ((\lambda u.T(M'\{v^\bullet/N\}\{v^\circ/s\}))\,\mathrm{T(M}\{v^\bullet/N\}\{v^\circ/s\}))= \\ \mathrm{T}(unpack\ M\{v^\bullet/N\}\{v^\circ/s\}\ to\ \langle u^\bullet, u^\circ\rangle\ in\ M'\{v^\bullet/N\}\{v^\circ/s\})= \\ \mathrm{T}((unpack\ M\ to\ \langle u^\bullet, u^\circ\rangle\ in\ M')\{v^\bullet/N\}\{v^\circ/s\})$$

- Caso $fetch[w]\, M$.

$$T(fetch[w]\, M)\{v/\lambda a.T(N)\} = ((\lambda b.b)\, T(M))\{v/\lambda a.T(N)\} = (\lambda b.b)\, (T(M)\{v/\lambda a.T(N)\})$$

Por HI,

$$T(M)\{v/\lambda a.T(N)\} \longrightarrow_\beta^* T(M\{v^\bullet/N\}\{v^\circ/s\})$$

Luego,

$$(\lambda b.b)\, (T(M)\{v/\lambda a.T(N)\}) \longrightarrow_\beta^* (\lambda b.b)\, (T(M\{v^\bullet/N\}\{v^\circ/s\})) = T(fetch[w]\, M\{v^\bullet/N\}\{v^\circ/s\}) = T((fetch[w]\, M)\{v^\bullet/N\}\{v^\circ/s\})$$

□

Lema 8. Demostración por inducción sobre M. Los casos base son triviales ya que no se pueden originar reducciones desde a o $v^\bullet$. Se muestran los casos en donde la reducción ocurre en la raíz de M.

- Caso $\longrightarrow_\beta$. Supongamos que $(\lambda a.M)\, N \longrightarrow_\beta M\{a/N\}$. Entonces:

$$\begin{array}{rll} & T((\lambda a.M)\, N) & \\ = & T(\lambda a.M)\, T(N) & \text{(Def. de } T(\cdot)) \\ = & (\lambda a.T(M))\, T(N) & \text{(Def. de } T(\cdot)) \\ \longrightarrow_\beta & T(M)\{a/T(N)\} & \\ = & T(M\{a/N\}) & \text{(Lema 6)} \end{array}$$

- Caso $\longrightarrow_{\beta_\Box}$. Supongamos que $unpack\ box_s\, M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N \longrightarrow_{\beta_\Box} N\{v^\bullet/M\}\{v^\circ/s\}$. Entonces:

$$\begin{array}{rll} & T(unpack\ box_s\, M\ to\ \langle v^\bullet, v^\circ\rangle\ in\ N) & \\ = & (\lambda v.T(N))\, T(box_s\, M) & \text{(Def. of } T(\cdot)) \\ = & (\lambda v.T(N))\, \lambda a.T(M) & \text{(Def. of } T(\cdot)) \\ \longrightarrow_\beta & T(N)\{v/\lambda a.T(M)\} & \\ \longrightarrow_\beta^* & T(N\{v^\bullet/M\}\{v^\circ/s\}) & \text{(Lema 7)} \end{array}$$

- Caso $\longrightarrow_{ftch}$. Supongamos que $fetch[w]\, M \longrightarrow_{ftcs} M$. Entonces:

$$\begin{array}{rll} & T(fetch[w]\, M) & \\ = & (\lambda a.a)\, T(M) & \text{(Def. of } T(\cdot)) \\ \longrightarrow_\beta & T(M) & \end{array}$$

□

Bibliografía

[AB04] Sergei Artëmov and Leo Beklemishev. Provability logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 13, pages 229–403. Kluwer, 2nd edition, 2004.

[AB07] Sergei Artëmov and Eduardo Bonelli. The intensional lambda calculus. In Artëmov and Nerode [AN07], pages 12–25.

[AN07] Sergei Artëmov and Anil Nerode, editors. *Logical Foundations of Computer Science, International Symposium, LFCS 2007, New York, NY, USA, June 4-7, 2007, Proceedings*, volume 4514 of *Lecture Notes in Computer Science*. Springer, 2007.

[AN09] Sergei Artëmov and Anil Nerode, editors. *Logical Foundations of Computer Science, International Symposium, LFCS 2009, Deerfield Beach, FL, USA, January 3-6, 2009. Proceedings*, volume 5407 of *Lecture Notes in Computer Science*. Springer, 2009.

[Art94] Sergei Artemov. Logic of proofs. *Ann. Pure Appl. Logic*, 67(1-3):29–59, 1994.

[Art95] Sergei Artemov. Operational modal logic. Technical Report MSI 95-29, Cornell University, 1995.

[Art01] Sergei Artemov. Unified semantics of modality and λ-terms via proof polynomials. *Algebras, Diagrams and Decisions in Language, Logic and Computation*, pages 89–118, 2001.

[BF00] Tijn Borghuis and Loe M. G. Feijs. A constructive logic for services and information flow in computer networks. *Comput. J.*, 43(4):274–289, 2000.

[BF08] Gilles Barthe and Cédric Fournet, editors. *Trustworthy Global Computing, Third Symposium, TGC 2007, Sophia-Antipolis, France, November 5-6, 2007, Revised Selected Papers*, volume 4912 of *Lecture Notes in Computer Science*. Springer, 2008.

[BF09] Eduardo Bonelli and Federico Feller. The logic of proofs as a foundation for certifying mobile computation. In Artëmov and Nerode [AN09], pages 76–91.

[DBL04] *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*. IEEE Computer Society, 2004.

[Eta05] Sandro Etalle, editor. *Logic Based Program Synthesis and Transformation, 14th International Symposium, LOPSTR 2004, Verona, Italy, August 26-28, 2004, Revised Selected Papers*, volume 3573 of *Lecture Notes in Computer Science*. Springer, 2005.

[GHC09] GHC. The glasgow haskell compiler. `http:\www.haskel.org/ghc`, 2009.

[GTL89] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*. Cambridge University Press, New York, NY, USA, 1989.

[Hap09] Happy. The parser generator for haskell. `http:\www.haskel.org/happy`, 2009.

[Has09] Haskell. Purely functional programming language. `http:\www.haskel.org`, 2009.

[JW04] Limin Jia and David Walker. Modal proofs as distributed programs (extended abstract). In Schmidt [Sch04], pages 219–233.

[Moo04] Jonathan Moody. Logical mobility and locality types. In Etalle [Eta05], pages 69–84.

[Mur08] Tom Murphy, VII. *Modal Types for Mobile Code*. PhD thesis, Carnegie Mellon, Janu3ary 2008. (draft).

[Nec00] George C. Necula. Proof-carrying code (abstract): design, implementation and applications. In *PPDP '00: Proceedings of the 2nd ACM SIGPLAN international conference on Principles and practice of declarative programming*, pages 175–177, New York, NY, USA, 2000. ACM.

[Ong05] C.-H. Luke Ong, editor. *Computer Science Logic, 19th International Workshop, CSL 2005, 14th Annual Conference of the EACSL, Oxford, UK, August 22-25, 2005, Proceedings*, volume 3634 of *Lecture Notes in Computer Science*. Springer, 2005.

[Sch04] David A. Schmidt, editor. *Programming Languages and Systems, 13th European Symposium on Programming, ESOP 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proceedings*, volume 2986 of *Lecture Notes in Computer Science*. Springer, 2004.

[VCH05] Tom Murphy VII, Karl Crary, and Robert Harper. Distributed control flow with classical modal logic. In Ong [Ong05], pages 51–69.

[VCH07] Tom Murphy VII, Karl Crary, and Robert Harper. Type-safe distributed programming with ml5. In Barthe and Fournet [BF08], pages 108–123.

[VCHP04] Tom Murphy VII, Karl Crary, Robert Harper, and Frank Pfenning. A symmetric modal lambda calculus for distributed computing. In *LICS* [DBL04], pages 286–295.

[Wad95] P. Wadler. Monads for functional programming. In *Advanced Functional Programming, Proceedings of the Bastad Spring School*, May 1995.