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

Tesis de Licenciatura en Ciencias de la Computación

$\lambda\sigma_{\mathrm{gc}}$: Un cálculo basado en $\lambda\sigma$ con Garbage Collection

Luis Francisco Ziliani - lziliani@dc.uba.ar - L.U. 356/01

Director: Alejandro Ríos

Julio de 2009

Resumen

Desde hace unos 20 años que distintos cálculos de sustituciones explícitas han dado diferentes resultados en la forma de imitar al λ-cálculo. Uno de los primeros, $\lambda\sigma$, ha sido inspiración para muchos de ellos por su forma de componer sustituciones y por su forma de trabajar los índices de de Bruijn. Sin embargo, ningún cálculo derivado de éste logra combinar satisfactoriamente todas las buenas propiedades que se esperan de un cálculo de sustituciones explícitas: preservación de la normalización fuerte (PSN), simulación del λ-cálculo (Sim), metaconfluencia (MC), etc.

En un reciente trabajo de D. Kesner se presenta un cálculo derivado de λx que tiene todas estas propiedades. Parte fundamental de este cálculo es la eliminación de "basura" -sustituciones que no modifican el término a sustituir, que al componerse con otras sustituciones terminan generando más basura, resultando en términos con derivaciones infinitas que en el λ-cálculo tradicional son fuertemente normalizantes.

En este trabajo presentamos $\lambda\sigma_{\mathrm{gc}}$, un cálculo basado en $\lambda\sigma$, que resulta de agregar una regla que elimina sustituciones "basura", y evita la composición y distribución de esta basura. Demostramos que este nuevo cálculo presenta las mismas propiedades que $\lambda\sigma$. Además, demostramos que $\lambda\sigma_{\mathrm{gc}}$ consigue incluir al término de Mèllies en el conjunto de términos fuertemente normalizantes, aunque PSN para este cálculo sigue siendo una pregunta abierta.

En el estudio de algunas propiedades del nuevo cálculo hemos desarrollado herramientas lógicas originales, que pueden servir a futuras extensiones de $\lambda\sigma$, y hemos abierto y desarrollado la discusión sobre cómo evitar la basura en cálculos derivados de dicho cálculo.

Abstract

For the last 20 years different calculi with explicit substitutions have been developed in order to imitate the λ-calculus, with diverse results. One of them, $\lambda\sigma$, has been an inspiration to many of them, because of the way it composes substitutions, and the way it handles de Bruijn indices. However, no calculus derived from it is known to achieve all the good properties expected from a calculus with explicit substitutions: preservation of strong normalization (PSN), simulation of λ-calculus (Sim), metaconfluence (MC), etc.

In a recent work, D. Kesner presents a new calculus obtained from λx with all these properties. To achieve this result, it does "garbage collection" -throw away substitutions that are not going to modify the term to substitute, whose composition with other substitutions ends up in the generation of more garbage, creating infinite derivations in terms which are strongly normalizing in traditional λ-calculus.

In this work we present $\lambda\sigma_{\mathrm{gc}}$, a calculus based on $\lambda\sigma$ that performs garbage collection, avoids the distribution of garbage through the terms, and does not allow for the composition of garbage. We prove that this new calculus has the same properties as $\lambda\sigma$. Moreover, we prove that $\lambda\sigma_{\mathrm{gc}}$ includes the Mèllies term in the set of strongly normalizing terms. But PSN for this calculus remains an open question.

In the study of some of the properties of the new calculus, we have developed some original logic tools. Future extensions of $\lambda\sigma$ can benefit from these ideas. Also, we have opened and advanced the discussion about how garbage can be avoided in calculi derived form $\lambda\sigma$.

A Marilí y Rody, mis padres.

Agradecimientos

Esta tesis es el producto de una serie infinita de eventos, en los que participaron diversas personas, muchas de las cuales no me conocen ni las conozco. Como nombrar a todas es una tarea imposible, voy a robar una frase que un compañero de militancia, Guido de Caso, puso en su tesis: agradezco a todos los que, desde su fundación, lucharon por una universidad pública, gratuita, autónoma y laica.

En concreto, en orden más o menos cronológico:

A mis padres, a quienes dedico este trabajo, mi reconocimiento a todo el saber que nunca dejaron de transmitirnos a mí y a mis hermanos, sin el cual posiblemente no podría estar presentando este trabajo.

A Ceci, mi novia, le debo nada menos que mi cambio de carrera, junto con la infinita paciencia que me tuvo ante cada instancia de presentación de trabajos, elecciones de centro de estudiantes, etc. Además, todas sus enseñanzas, ya sean académicas, poltícias, o cotidianas. A ella, todo mi amor.

A mis compañeros de cursada, Alexis, Edu, Elvis, Gastón, gracias por el rock!

A mis compañeros de La Mella, les debo lo que lamentablemente no se enseña en las aulas, pero se vive en toda intereacción humana: la política. A todos y cada uno de ellos, un abrazo lleno de esperanza.

A Alejandro Ríos, le agradezco su guía paciente y sabia.

A Nicolás di Tada, Director de Manas Technology Solutions, la empresa de la que formo parte desde hace casi 5 años, un profundo agradecimiento por haber sido sponsor de este trabajo. Espero que este ejemplo de ayudar a la investigación universitaria **sin condicionarla** sea imitado por quienes intentan llevar "agua para su molino", consumiendo los recursos del estado, es decir, del pueblo.

Finalmente, y pido disculpas si ofendo a alguien, a la estampita de Sai Baba, cábala oficial de toda mi carrera.

Índice general

Introducción

La implementación del λ-cálculo ha presentado varios desafíos desde sus primeros intentos, ya sea en el estudio y creación de lenguajes funcionales, como en demostradores de teoremas. Así es como de Bruijn concibe en el año 1972 lo que actualmente se conoce como cálculo con índices de de Bruijn. La finalidad de este cálculo era eliminar los problemas de trabajar *módulo* α-congruencia, definiendo un único término para cada clase de equivalencia.

En el año 1990, Abadi *et al.* crean el cálculo $\lambda\sigma$, un *cálculo de sustituciones explícitas* (SE) basado en Categorical Combinatory Logic ([Cur94], [CC87], [Har92]). Estos cálculos internalizan la metaoperación de sustitución como pasos atómicos propios del cálculo. Pero, como lo muestran Abadi *et al.*, el cálculo $\lambda\sigma$ no es *metaconfluente* (MC), es decir, confluente en los términos con *metavariables*, propiedad necesaria para representar pruebas incompletas en demostradores de teoremas basados en tipos. Además, en el año 1995 Melliès muestra otro inconveniente en la teoría de este cálculo: no todo término *fuertemente normalizante* del λ-cálculo lo es en $\lambda\sigma$. Esta propiedad, deseable en todo cálculo con SE, es denominada *preservación de la normalización fuerte* (PSN).

Desde entonces, han proliferado distintos tipos de cálculos de sustituciones explícitas en la búsqueda de uno que satisfaga MC, PSN, y además simule la β-reducción (Sim).

Una extensión de $\lambda\sigma$ estudiada por Ríos en [Rio93], $\lambda\sigma_{\mathrm{SP}}$, consigue MC de términos *semiabiertos*, es decir, con metavariables en los términos y no en las sustituciones. Otra variante presentada por Curien *et al.* en [CHL96], el $\lambda\sigma_{\Uparrow}$, consigue MC y Sim, pero pierde PSN. Lescanne en [Les94] presenta $\lambda\upsilon$, variante que conserva PSN y Sim, pero no MC, y Muñoz en [Mun96] consigue con $\lambda\zeta$ PSN y MC, pero no Sim.

En el año 1995, Kamareddine y Ríos presentan en [KR95] al cálculo λs, que incluye dos familias de operadores para simular la sustitución y la actualización de de Bruijn. Si bien cumple PSN y Sim, no consigue MC. Dos años más tarde (en [KR97]), los mismos autores lo extienden con el fin de lograr MC. Esta extensión, denominada λs_e, consigue Sim y MC, pero no cumple PSN.

La búsqueda de un cálculo que preservara MC, PSN, y Sim parecía infructuosa, pero en el 2001, David y Guillaume presentan en [DG01] al cálculo λws, que logra esas tres propiedades con una signatura infinita, marcada por *labels*.

Hasta aquí hemos mencionado todos cálculos que utilizan índices para referir a las variables. En el conjunto de cálculos de SE y variables nombradas, λx ([Ros93]) fue el pionero, aunque no consigue MC. Kesner en [Kes07] presenta λes, un cálculo que posee MC, PSN, Sim, entre otras buenas propiedades.

En el presente trabajo hemos tomado algunas ideas del cálculo λes para llevarlas al terreno de los índices de de Bruijn, modificando al cálculo $\lambda\sigma$ res-

tringiendo la composición.

Plan

En el capítulo 1 presentaremos las nociones básicas de la teoría de la reescritura. Introduciremos el λ-cálculo en la versión tradicional y à la de Bruijn. Luego, nos concentraremos en $\lambda\sigma$, por ser la base del cálculo presentado en el capítulo 2. Finalizaremos el capítulo con una breve presentación de los cálculos λx y λes, el segundo por ser inspirador del presente trabajo, el primero por ser antecesor de λes.

Dedicaremos el capítulo 2 a definir el concepto de *basura* para el cálculo $\lambda\sigma$, concepto que nos permitirá definir una variante de $\lambda\sigma$ con *garbage collection*: $\lambda\sigma_{\mathrm{gc}}$.

En el capítulo 3 demostraremos que vale la normalización fuerte para el cálculo asociado de sustituciones, σ_{gc}. Daremos una demostración detallada utilizando la técnica de *Distribution elimination* utilizada por Zantema para demostrar la misma propiedad para el cálculo σ.

A lo largo del capítulo 4 demostraremos varios lemas, con el fin de obtener un conjunto de herramientas teóricas para los siguientes capítulos.

En el capítulo 5 analizaremos todos los pares críticos de σ_{gc} para concluir su *confluencia débil*, y luego su *confluencia*, necesaria para la demostración de confluencia del cálculo $\lambda\sigma_{\mathrm{gc}}$ del siguiente capítulo.

Dedicaremos el capítulo 6 a demostrar la confluencia del cálculo $\lambda\sigma_{\mathrm{gc}}$, para lo cual será necesario primero establecer que se cumplen dos propiedades: *Simulación* y *Corrección*. Demostraremos estas propiedades extendiendo las pruebas realizadas por Ríos para $\lambda\sigma$. Finalmente, estableceremos la confluencia de $\lambda\sigma_{\mathrm{gc}}$ utilizando el lema de interpretación.

Brevemente introduciremos en el capítulo 7 una versión tipada de $\lambda\sigma_{\mathrm{gc}}$, que se corresponderá con la versión tipada de $\lambda\sigma$.

En el capítulo 8 mostraremos que el contraejemplo de Mèllies pertenece al conjunto de términos fuertemente normalizantes de $\lambda\sigma_{\mathrm{gc}}$, y comenzaremos la discusión acerca de la relación entre los términos fuertemente normalizantes de los cálculos λ, $\lambda\sigma$, y $\lambda\sigma_{\mathrm{gc}}$.

Para terminar, presentaremos distintos problemas que consideramos interesantes para continuar este trabajo.

Por último, en los anexos mostraremos intérpretes para $\lambda\sigma$ y $\lambda\sigma_{\mathrm{gc}}$ que serán utilizados en demostraciones que requieren un análisis exhaustivo de todas las posibles derivaciones de un término.

Capítulo 1

Presentación de formalismos

En este capítulo se introducirán algunos de los formalismos utilizados o referenciados en el presente trabajo.

La sección 1.1 estará dedicada a presentar algunas definiciones y resultados de la teoría de la reescritura, enfocada en las herramientas utilizadas en el estudio de las propiedades del cálculo $\lambda\sigma_{\mathrm{gc}}$. En la sección 1.2 se presentará el cálculo que ha inspirado gran parte de los trabajos aquí mencionados: el λ-cálculo. Luego, en 1.2.1 se mostrará una versión tipada del mismo, que será útil para comprender las versiones tipadas de $\lambda\sigma$ y $\lambda\sigma_{\mathrm{gc}}$. En la sección 1.3 se explicará un refinamiento del λ-cálculo que evita la α-congruencia, λ_{DB}. En la sección 1.4 se detallará una variante de λ_{DB} que explicita la sustitución y admite composición de sustituciones: el cálculo $\lambda\sigma$, base del presente trabajo. En la sección 1.4.1 se describirá la versión tipada de este cálculo. Finalmente, en la sección 1.5 se retomarán los cálculos de sustituciones con variables nombradas para introducir λx, el cálculo base de λes (sección 1.6), que ha inspirado el presente trabajo.

1.1 Introducción a la teoría de la reescritura

Aquí serán presentados los elementos básicos de la teoría de la reescritura. Dada la vastedad de este tema, nos centraremos estrictamente en lo requerido para la comprensión del presente trabajo. Para una profundización en el tema, el lector puede encontrar mayor información en [BN98] y en [Ter03].

1.1.1 Sistemas de Reducción Abstractos (ARS)

Sea A un conjunto y R una relación binaria sobre A. Notaremos $a \to_R b$ para $(a, b) \in R$. Llamaremos *reducción* a esta relación, y ARS al par (A, R). Además, notaremos

- R^+ o $\overset{+}{\to}_R$ a la clausura transitiva de R.
- R^* o $\twoheadrightarrow_R$ a la clausura reflexiva transitiva de R.

- $a \overset{n}{\twoheadrightarrow}_R b$ a la reducción de a en n pasos a b, es decir,

$$a \to_R a_1 \to_R \ldots \to_R a_n = b$$

  También llamaremos *derivación* a una reducción (posiblemente infinita) $a_1 \to_R a_2 \to_R \ldots$

Un ejemplo sencillo de ARS puede ser $(\mathbb{N}, <)$, la relación $<$ entre los naturales. Luego,

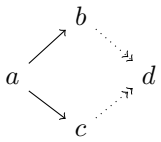
$$\begin{array}{lll} 0 & \to_< & 1 \\ 1 & \to_< & 3 \\ 5 & \twoheadrightarrow_< & 9 \end{array}$$

Confluencia

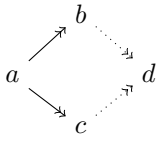
En el estudio de los ARS, resulta indispensable el estudio de la *confluencia*.

Definición 1. Sea (A, R) un *ARS*, decimos que

- R es localmente confluente o *WCR* (weakly Church-Rosser) sii R satisface $(\forall a, b, c \in A)\ (\exists d \in A)$



- R es confluente o *CR* (Church-Rosser) sii R satisface $(\forall a, b, c \in A)\ (\exists d \in A)$



Observación 2. Si R es CR, entonces R es WCR.

Normalización

Otro concepto importante en la teoría de la reescritura es el de *formas normales*, que son aquellos elementos que no puede reducirse más. Cómo ejemplo, si en vez de tomar la relación $<$ tomamos la relación $>$ sobre los naturales, obtenemos que 0 es la única forma normal.

A continuación presentaremos las definiciones relacionadas con este concepto.

Definición 3. Sea (A, R) un *ARS*, decimos que

- $a \in A$ es una *R-forma normal* (o R-f.n.) si $(\nexists b \in A)\ a \to_R b$.
- $a \in A$ es una *forma normal de* b si es una R-f.n. y $b \twoheadrightarrow_R a$.
- R es *débilmente normalizante* o *WN* (weakly normalizing) si todo elemento de A tiene al menos una forma normal.
- R es *fuertemente normalizante* o *SN* (strongly normalizing), si no hay derivaciones infinitas.
- $\mathcal{SN}_R$ es el subconjunto de A cuyos elementos no tienen derivaciones infinitas en R.
- $a \in A$ es *SN* si $a \in \mathcal{SN}_R$.

Lema 4. Si R es SN, entonces R es WN.

Demostración. Si R no fuera WN, entonces existe un elemento a_0 que no tiene forma normal, con lo que existe un elemento a_1 tal que $a_0 \to_R a_1$, pero nuevamente a_1 no tiene forma normal, con lo que existe a_2 tal que $a_1 \to_R a_2$... y así se construye una derivación infinita, absurdo. □

Los siguientes lemas nos permiten relacionar los conceptos hasta aquí mencionados:

Lema 5 (Newman)**.** Toda relación R fuertemente normalizante y localmente confluente es confluente.

Demostración. Ver [BN98], lema 2.7.2. □

Lema 6. Sea (A, R) un *ARS*, tal que R es *SN* y *WCR*, entonces todo elemento de A tiene una única forma normal.

En este caso, notaremos $R(a)$ a la R-f.n. de a.

Demostración. Por *SN* y lema 4 se tiene que todo elemento tiene una R-f.n. Si la forma normal no fuera única, entonces R no sería confluente, pero Newman nos lo asegura. □

Presentamos ahora un lema que permite relacionar la confluencia entre dos ARS distintos. Este lema es utilizado en el capítulo 6 para demostrar confluencia del cálculo presentado en este trabajo.

Lema 7 (Lema de interpretación)**.** Sea R_1 una reducción que cumple CR y SN, R_2 una relación cualquiera. Sea $R = R_1 \cup R_2$, y R' una relación en las R_1-f.n. tal que

$$R' \subseteq R^* \quad \text{y} \quad a \to_{R_2} b \implies R_1(a) \twoheadrightarrow_{R'} R_1(b)$$

Entonces, R' es *CR* $\iff$ R es *CR*.

Demostración. Ver [CHL96], lema 1.1. □

1.1.2 Sistemas de Reescritura de Términos (TRS)

Mientras en los ARS se reescriben objetos arbitrarios, en los TRS se reescriben términos de primer orden. Para ello, se definen las *reglas de reescritura* que especifican como reemplazar parte de un término por otro. Cómo ejemplo, una definición de los números naturales y la operación de suma se puede definir como

$$\begin{array}{lcl} x+0 & \longrightarrow & x \\ x+S(y) & \longrightarrow & S(x)+y \end{array}$$

donde 0 representa el cero, $S(0)$ el uno, etc. Las variables x e y pueden ser reemplazadas por cualquier término del sistema. Para computar la suma de $(0+1)+2$ se puede realizar

$$\begin{array}{lcl} (0+S(0))+S(S(0)) & \rightarrow & (S(0)+0)+S(S(0)) \rightarrow S(S(0)+0)+S(0) \\ & \rightarrow & S(S(0))+S(0) \rightarrow S(S(S(0)))+0 \rightarrow S(S(S(0))) \end{array}$$

A continuación presentaremos formalmente el concepto de TRS, para el cual necesitaremos primero las siguientes definiciones:

Definición 8.

- Una *signatura* es un conjunto de símbolos de funciones. Usualmente se la nota Σ. Un símbolo de función f de aridad n es notado $f^{(n)}$, pero si por contexto puede deducirse la aridad, ésta se puede obviar.
- Un *conjunto de términos* $T_\Sigma(V)$ (también llamados Σ-términos), con V un conjunto de variables y Σ una signatura, se define inductívamente como
  - $V \subseteq T_\Sigma(V)$
  - $(\forall n \geq 0)\ (\forall f^{(n)} \in \Sigma)\ (\forall t_1, \ldots, t_n \in T_\Sigma(V))$

    $$f(t_1, \ldots, t_n) \in T_\Sigma(V)$$

  En el ejemplo: $x, y \in V$, $0^{(0)}, S^{(1)}, +^{(2)} \in \Sigma$, y $0, S(0), 0+S(0) \in T_\Sigma(V)$.
- Un *contexto* (denotado C) es un término que contiene exactamente una ocurrencia de un símbolo especial $\Box$. Notaremos $C[t]$ al reemplazo de $\Box$ por t en C.
- Una *sustitución* σ es una función $V \rightarrow T_\Sigma(V)$ de variables a términos. Notaremos $\sigma(t)$ al reemplazo de toda variable x en t por $\sigma(x)$.

  Dadas dos sustituciones σ y τ, notaremos $\sigma \leq \tau$ si existe una sustitución ρ tal que $\tau = \sigma \circ \rho$ ($\circ$ es la composición de funciones).
- La sustitución σ *unifica* dos términos s y t si $\sigma(s) = \sigma(t)$. Si existe tal sustitución, decimos que s y t son *unificables*.
- La sustitución τ es el *Unificador Más General* (*umg*) de dos términos unificables s y t si para toda sustitución σ que unifica a s y t, $\tau \leq \sigma$. Dados dos términos unificables, siempre se puede computar su *umg* (por ejemplo, ver el capítulo 2 de [Ter03]).

- Una *regla de reescritura* r es un par (s,t) donde $s,t \in T_\Sigma(V)$, y
  1. $s \notin V$
  2. $V(t) \subseteq V(s)$

  donde $V(t)$ son las variables que aparecen en t. Notamos $s \to_r t$ a dicha regla.

Ahora sí podemos definir TRS:

Definición 9. Un *Sistema de Reescritura de Términos* (*TRS*) es un par $(T_\Sigma(V), \mathcal{R})$, donde $\mathcal{R}$ es un conjunto de reglas de reescritura.

La reducción R creada por $\mathcal{R}$ se define como

$$s \to_R t$$

sii existen $(u,v) \in \mathcal{R}$, σ una sustitución, C un contexto, tales que

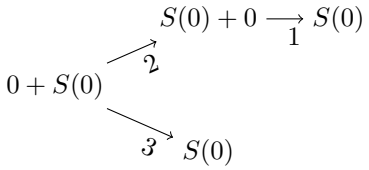
$$s = C[\sigma(u)] \quad \wedge \quad t = C[\sigma(v)]$$

Par Crítico

Para el estudio de la confluencia débil, resulta necesario estudiar la interacción entre las reglas. Por ejemplo, en el TRS

$$\begin{array}{lllll} (1) & x+0 & \longrightarrow & x \\ (2) & x+S(y) & \longrightarrow & S(x)+y \\ (3) & 0+x & \longrightarrow & x \end{array}$$

desde el término $0+S(0)$ existen dos posibles derivaciones a $S(0)$:



En este ejemplo, ambos casos reducen al mismo resultado, pero esto podría no ser así.

Definición 10 (Par crítico)**.** Sean $s \to t$ y $u \to v$ dos reglas de reescritura. Sea w un subtérmino de u que no es una variable, y que unifica con s. Sea σ el *umg* de w y s, y sea C el contexto tal que $u = C[w]$. El término $\sigma(u)$ puede entonces reducirse de dos maneras:

$$\sigma(u) = \sigma(C[w]) \to \sigma(C[t]) \quad \text{y} \quad \sigma(u) \to \sigma(v)$$

El par $(\sigma(C[t]), \sigma(v))$ se llama *par crítico* (CP) resultante de la superposición de las reglas $s \to t$ y $u \to v$.

Si $s = u$ y $t = v$, se pide que s sea unificable con un subtérmino *propio* de u.

En el ejemplo, el par $(S(0)+0, S(0))$ es el par crítico resultante de la superposición de las reglas 2 y 3.

Para más detalles, ver [BN98] cap. 6 o [Ter03] cap. 2.

Lema 11 (Critical Pair Lemma)**.** Un *TRS* es *WCR* sii todos sus pares críticos convergen, es decir, para todo par de reglas r, t que forman un par crítico (b, c), existe d tal que

$$b \twoheadrightarrow d \twoheadleftarrow c$$

Demostración. Ver [BN98], teorema 6.2.4. □

Terminación Total

Presentaremos en esta subsección un concepto de terminación que nos va a permitir en el capítulo 3 determinar SN de un cálculo. Este concepto se encuentra desarrollado de [Ter03] sección 6.3.2.

Definición 12 (Σ-álgebra monótona bien fundada)**.**

1. Una Σ-*álgebra* definida sobre la signatura Σ está compuesta de un conjunto base A y una interpretación $f_A : A^n \to A$ para cada símbolo de función $f \in \Sigma$ de aridad n.

2. Se define una Σ-*álgebra monótona bien fundada* $(A, >)$ como una Σ-álgebra cuyo conjunto base A está provisto de un orden bien fundado[1] $>$ y para todo $f \in \Sigma$ de aridad n, y todos los $a_1, \ldots, a_n, b_1, \ldots, b_n \in A$ en los cuales $a_i > b_i$ para algún i, y $a_j = b_j$ para todo $j \neq i$, se tiene

$$f_A(a_1, \ldots, a_n) > f_A(b_1, \ldots, b_n)$$

3. Decimos que una Σ-álgebra monótona bien fundada y no vacía $(A, >)$ es *compatible* con un TRS si $l_A > r_A$ para todas las reglas $l \to r$ del TRS, donde t_A es la interpretación del término t.

Definición 13. Un TRS es *totalmente terminante* (TT) si admite una Σ-álgebra monótona y bien fundada $(A, >)$ compatible con él, en el cual $>$ es total en A.

Los teoremas presentados a continuación nos permitirán deducir cuándo un cálculo es SN a partir de TT.

Teorema 14. Si un TRS es TT, entonces es SN.

Demostración. Ver [Zan00] proposición 29, o [Ter03] sección 6.3.2. □

Definición 15. Se define el TRS $Emb(\Sigma)$ para una signatura Σ como aquél que tiene el conjunto de reglas

$$f(x_1, \ldots, x_n) \to x_i \qquad (\forall f \in \Sigma)(\forall i \in [1, n])$$

Teorema 16. Un TRS R es totalmente terminante *sii* $R \cup Emb(R)$ es totalmente terminante.

Demostración. Ver [Zan94], proposición 8. □

[1] Que no admite cadenas infinitas decrecientes $a_1 > a_2 > \ldots$

"Distribution Elimination"

Esta subsección estará dedicada a una técnica presentada por Zantema en [Zan94], que permite simplificar un TRS sin perder la terminación total.

Definición 17. Sea $l \to r$ una regla de reescritura. Se dice que es una *regla distribuidora* del símbolo f de aridad n si existe un contexto no trivial C tal que

$$l = C[f(x_1, \ldots, x_n)] \quad r = f(C[x_1], \ldots, C[x_n])$$

y f no ocurre en C.

Definición 18. Dado un conjunto de términos $T_\Sigma(V)$, se define la función[2] $Erase_f : T_\Sigma(V) \to \mathcal{P}(T_\Sigma(V))$, para $f \in \Sigma$, como

$$\begin{array}{lcll} Erase_f(x) & = & \{x\} & (\forall x \in V) \\ Erase_f(g(t_1, \ldots, t_k)) & = & \{g(u_1, \ldots, u_k) | (\forall i)\ u_i \in Erase_f(t_i)\} & (\forall g \in \Sigma, g \neq f) \\ Erase_f(f(t_1, \ldots, t_n)) & = & \bigcup_{i=1}^{n} Erase_f(t_i) & \end{array}$$

Definición 19. Sea R un TRS tal que toda regla es o una regla distribuidora de f o una regla donde f no aparece en el lado izquierdo. $Erase_f(R)$ es el TRS definido a partir del conjunto de reglas

$$\{l \to u | l \to r \in R \text{ no es una regla distribuidora de } f \text{ y } u \in Erase_f(r)\}$$

Teorema 20 (Distribution Elimination)**.** Sea R un TRS tal que toda regla es o una regla distribuidora de f o una regla donde f no aparece en el lado izquierdo. Entonces $Erase_f(R)$ es totalmente terminante sii R es totalmente terminante.

Demostración. Ver [Zan94], teorema 12. □

1.2 Cálculo λ

El cálculo λ fue creado por Alonzo Church en la década de 1930. Nosotros utilizaremos la versión presentada en [Bar84].

El conjunto de términos del λ cálculo Λ se encuentra definido para un conjunto de variables V por

$$t ::= x \mid t\ t' \mid (\lambda x.t) \quad (x \in V)$$

Definición 21. Se define para un término $t \in \Lambda$ el conjunto de *variables libres* de t como

$$\begin{array}{rcl} \mathrm{FV}(x) & = & x \\ \mathrm{FV}(t\ u) & = & \mathrm{FV}(t) \cup \mathrm{FV}(u) \\ \mathrm{FV}(\lambda x.t) & = & \mathrm{FV}(t) - \{x\} \end{array}$$

Definición 22. Para $t, u \in \Lambda$ y $x \in V$, definimos $t[\![x/u]\!]$ como la sustitución en t de la variable x por u:

$$\begin{array}{rcll} x[\![x/u]\!] & = & u & \\ y[\![x/u]\!] & = & y & (x \neq y) \\ (t_1\ t_2)[\![x/u]\!] & = & t_1[\![x/u]\!]\ t_2[\![x/u]\!] & \\ (\lambda x.t)[\![x/u]\!] & = & \lambda x.t & \\ (\lambda y.t)[\![x/u]\!] & = & \lambda y.(t[\![x/u]\!]) & (y \neq x \wedge (y \notin \mathrm{FV}(u) \vee x \notin \mathrm{FV}(t))) \\ (\lambda y.t)[\![x/u]\!] & = & \lambda z.(t[\![y/z]\!][\![x/u]\!]) & (y \neq x \wedge y \in \mathrm{FV}(u) \wedge x \in \mathrm{FV}(t)) \end{array}$$

[2] Donde $\mathcal{P}$ es el conjunto de partes

En el útlimo caso, z es una variable "fresca" (que no figura libre en t ni en u).

Definición 23 ($\equiv_\alpha$)**.** Sea $t = C[\lambda x.u]$, y sea $y \notin \mathrm{FV}(u)$. Se dice que t' se obtiene a partir de t por un cambio de variable ligada si $t' = C[\lambda y.u[\![x/y]\!]]$.

Decimos que t y t' son α-congruentes, notado $t \equiv_\alpha t'$, si t' puede obtenerse por una serie de cambios de variables ligadas.

Definición 24 (β-reducción)**.** Se define la relación denominada β-reducción entre términos, notada $\to_\beta$, como

$$t \to_\beta u \quad \text{sii existe un contexto C y términos } v, w \text{ tales que}$$

$$t = C[(\lambda x.v)\ w] \quad u = C[v[\![x/w]\!]]$$

Definición 25. El λ-cálculo es el sistema de reducción $(\Lambda/\equiv_\alpha, \beta)$

Observación 26. El λ-cálculo no es SN.

Demostración. Se muestra un contraejemplo. Sea el término

$$\Omega = (\lambda x.x\ x)\ (\lambda x.x\ x)$$

notar que $\Omega \to_\beta \Omega$. □

Teorema 27. El λ-cálculo es CR.

Demostración. Hay varias pruebas en la literatura. Por ejemplo, [Bar84], sección 11.1 □

1.2.1 λ-cálculo simplemente tipado

Presentamos aquí una versión explícitamente tipada del cálculo. El conjunto de tipos de primer orden T se encuentra definido para un conjunto de variables de tipos V_T como

$$T ::= v \mid T \to T \quad (v \in V_T)$$

Los términos son modificados para indicar el tipo de las variables en las abstracciones:

$$t ::= x \mid t\ t' \mid (\lambda x : T.t) \quad (x \in V)$$

Las siguientes definiciones nos permitirán establecer el tipo de un término:

Definición 28.

- Dados un término M y un tipo σ, una *aserción* se nota como $M : \sigma$, y se lee "el término M tiene tipo σ".
- Se define *base* (usualmente notado Γ) a un conjunto de aserciones cuyos términos son variables disjuntas.

- Una aserción $M : \sigma$ se *infiere* o *deriva* de una base Γ, notado $\Gamma \vdash M : \sigma$, si se obtiene de los siguientes axiomas y reglas:

$$\Gamma \vdash x : \sigma \quad \text{si } (x : \sigma) \in \Gamma$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\ N : \tau}$$

$$\frac{\Gamma \cup \{x : \sigma\} \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma.M : \sigma \to \tau}$$

El siguiente teorema asegura la perservación del tipo de un término bajo β-reducciones.

Teorema 29 (Subject Reduction)**.** Sean M, N términos, y Γ una base,

$$M \to_\beta N \quad \text{entonces} \quad \Gamma \vdash M : \sigma \implies \Gamma \vdash N : \sigma$$

Demostración. Ver [Bar92] proposición 3.2.11. □

Teorema 30. Todo término que puede tiparse dadas las reglas de inferencia de tipos es SN.

Demostración. Existen varias pruebas en la literatura, por ejemplo [GTL89] cap. 6. □

1.3 Cálculo $\lambda_{\mathbf{DB}}$

Nicolaas Govert de Bruijn introduce en [dB72] un cálculo basado en el λ-cálculo, en el cual reemplaza el conjunto de variables por un conjunto de "índices" naturales. Informalmente, cada índice i hace referencia a la cantidad de λ's que se deben cruzar para encontrar el λ que lo abstrae. A continuación mostramos algunos ejemplos, traduciendo los términos desde el λ-cálculo:

$\lambda x.x$	se traduce como	$\lambda 1$
$\lambda z.z$	se traduce como	$\lambda 1$
$\lambda y.(\lambda x.y)$	se traduce como	$\lambda\lambda 2$
$\lambda x.(\lambda y.x\ y)$	se traduce como	$\lambda\lambda(2\ 1)$

Al realizar esta traducción debe considerarse un caso especial: cuando el λ-término tiene variables libres. En ese caso se deben realizar algunos pasos adicionales. Primero, se debe establecer un orden entre las variables. Luego se debe cerrar el término, agregando λ's abstrayendo las variables libres en el orden establecido. Finalmente se realiza la traducción y se quitan los λ's agregados.

Cómo ejemplo, traduciremos el término $\lambda u.z$.

1. Establecemos el orden de variables como $x, y, z, u, \ldots$.
2. Cerramos el término: $\lambda z.\lambda y.\lambda x.(\lambda u.z)$.

$$
\begin{array}{llll}
(\beta) & (\lambda a)\ b & \longrightarrow & a\{1 \leftarrow b\} \\
 & & & \\
 & j\{i \leftarrow b\} & = & \begin{cases} j-1 & j > i \\ \mathrm{U}_0^i(b) & j = i \\ j & j < i \end{cases} \\
 & (a\ b)\{i \leftarrow c\} & = & a\{i \leftarrow c\}\ b\{i \leftarrow c\} \\
 & (\lambda a)\{i \leftarrow b\} & = & \lambda a\{i+1 \leftarrow b\} \\
 & & & \\
 & \mathrm{U}_j^k(i) & = & \begin{cases} i & i \leq j \\ i+k-1 & i > j \end{cases} \\
 & \mathrm{U}_j^k(a\ b) & = & \mathrm{U}_j^k(a)\ \mathrm{U}_j^k(b) \\
 & \mathrm{U}_j^k(\lambda a) & = & \mathrm{U}_{j+1}^k(a)
\end{array}
$$

Figura 1.1: Cálculo λ_{DB}

3. Traducimos: $\lambda\lambda\lambda 4$.

4. Quitamos los λ's agregados: $\lambda 4$.

Haciendo este reemplazo de variables por índices se tiene que términos α-congruentes resultan idénticos en λ_{DB}.

A continuación presentaremos la notación utilizada actualmente para describir este cálculo.

Los términos de λ_{DB} se definen como:

$$a ::= i \mid a\ a \mid \lambda a \quad (i \in \mathbb{N}_{>0})$$

El cálculo se encuentra conformado por la regla de reducción β y la metasustitución definida en la figura 1.1.

Una consecuencia de esta forma de indicar las variables, es que cada vez que se elimina un λ al aplicar la regla β, las variables deben ser actualizadas para reflejar este cambio. Esto explica la complejidad de la sustitución.

A modo de ejemplo, mostramos una derivación en λ-cálculo, y su traducción en λ_{DB}:

$$
\begin{array}{rcl}
\lambda & : & (\lambda u.(\lambda w.u)\ y)\ z\ \rightarrow_\beta\ (\lambda w.z)\ y\ \rightarrow_\beta\ z \\
\lambda_{\mathrm{DB}} & : & (\lambda(\lambda 2)\ 3)\ 3\ \rightarrow_\beta\ (\lambda 4)\ 2\ \rightarrow_\beta\ 3
\end{array}
$$

1.4 Cálculo $\lambda\sigma$

En el λ-cálculo y en λ_{DB}, la sustitución es una metaoperación del sistema.

En [ACCL91] se presenta un cálculo novedoso que utiliza indices de de Bruijn, en el cual las sustituciones son tratadas de forma explícita dentro del cálculo. Este cálculo, denominado $\lambda\sigma$, además permite la *composición de sustituciones*, es decir, dos o más sustituciones pueden ser compuestas en una única sustitución.

(Beta)	$(\lambda a)\ b$	$\longrightarrow$	$a[b \cdot \mathbf{id}]$
(VarId)	$1[\mathbf{id}]$	$\longrightarrow$	1
(VarCons)	$1[a \cdot s]$	$\longrightarrow$	a
(App)	$(a\ b)[s]$	$\longrightarrow$	$a[s]b[s]$
(Abs)	$(\lambda a)[s]$	$\longrightarrow$	$\lambda(a[1 \cdot (s \circ \uparrow)])$
(Clos)	$a[s][t]$	$\longrightarrow$	$a[s \circ t]$
(IdL)	$\mathbf{id} \circ s$	$\longrightarrow$	s
(ShiftId)	$\uparrow \circ \mathbf{id}$	$\longrightarrow$	$\uparrow$
(ShiftCons)	$\uparrow \circ (a \cdot s)$	$\longrightarrow$	s
(Map)	$(a \cdot s) \circ t$	$\longrightarrow$	$a[t] \cdot (s \circ t)$
(Ass)	$(s \circ t) \circ u$	$\longrightarrow$	$s \circ (t \circ u)$

Figura 1.2: Cálculo $\lambda\sigma$

Presentamos primero el cálculo, para dar luego una breve introducción informal de su funcionamiento.

En $\lambda\sigma$ los términos están definidos por dos conjuntos o *sorts*: el *sort* de términos propiamente dichos, y el *sort* de sustituciones.

$$\Lambda\sigma^t \ \{ \quad a ::= 1 \mid a\ a \mid \lambda a \mid a[s]$$

$$\Lambda\sigma^s \ \{ \quad s ::= \mathbf{id} \mid \uparrow \mid a \cdot s \mid s \circ s$$

El cálculo se encuentra definido por las reglas mostradas en la figura 1.2.

Se llama σ al cálculo de sustituciones asociado, es decir, $\lambda\sigma - \text{Beta}$.

Definición 31. Se define la composición n-ésima de una sustitución s como

$$\begin{aligned} s^0 &= \mathbf{id} \\ s^1 &= s \\ s^{i+1} &= s \circ s^i \end{aligned}$$

Informalmente, un índice de de Bruijn i es notado en este cálculo como $1[\uparrow^{i-1}]$. Luego, una sustitución de la forma

$$a_1 \cdot a_2 \cdot \ldots \cdot a_m \cdot \uparrow^n$$

sustituye las variables de un término reemplazando 1 por a_1, la variable 2 por a_2, ..., la variable m por a_m, y el resto de las variables son incrementadas por la diferencia de $n - m$ (que puede ser negativa). La sustitución **id** es interpretada como la sustitución $1 \cdot 2 \cdot 3 \cdot \ldots$, es decir, la que reemplaza cada variable por sí misma. La sustitución $\uparrow$ es interpretada como la sustitución $2 \cdot 3 \cdot 4 \cdot \ldots$, es decir, la que reemplaza cada variable i por $i + 1$.

1.4.1 Cálculo $\lambda\sigma$ simplemente tipado

En [ACCL91] se presenta una versión simplemente tipada del cálculo. Una *base* va a ser simplemente una lista de tipos, permitiendo asignar el i-ésimo tipo a la i-ésima variable. Los términos son modificados para permitir anotaciones de tipo:

Tipos	$A ::=$	$K \mid A \to B$
Bases	$E ::=$	$\text{nil} \mid A, E$
Términos	$a ::=$	$1 \mid a\ a \mid \lambda A.\, a \mid a[s]$
Sustituciones	$s ::=$	$\mathbf{id} \mid \uparrow \mid a : A \cdot s \mid s \circ s$

El cálculo $\lambda\sigma$ simplemente tipado consta de las mismas reglas que el original, con excepción de *Beta*, *VarCons*, *Abs*, *ShiftCons*, *Map* que se reemplazan por

(Beta)	$(\lambda A.\, a)\ b$	$\longrightarrow$	$a[b : A \cdot \mathbf{id}]$
(VarCons)	$1[a : A \cdot s]$	$\longrightarrow$	a
(Abs)	$(\lambda A.\, a)[s]$	$\longrightarrow$	$\lambda A.\, (a[1 : A \cdot (s \circ \uparrow)])$
(ShiftCons)	$\uparrow \circ (a : A \cdot s)$	$\longrightarrow$	s
(Map)	$(a : A \cdot s) \circ t$	$\longrightarrow$	$a[t] : A \cdot (s \circ t)$

Definición 32. En base a las reglas de inferencia mostradas abajo, se definen:

1. Un término a tiene tipo A, según una base E, notado $E \vdash a : A$
2. Una sustitución s actualiza una base E, resultando en la base E', notado $E \vdash s \rhd E'$

$$\text{var}\quad A, E \vdash 1 : A$$

$$\text{lambda}\quad \frac{A, E \vdash b : B}{E \vdash \lambda A.\, b : A \to B}$$

$$\text{app}\quad \frac{E \vdash a : B \to A \qquad E \vdash b : B}{E \vdash a\ b : A}$$

$$\text{clos}\quad \frac{E \vdash s \rhd E' \qquad E' \vdash a : A}{E \vdash a[s] : A}$$

$$\text{id}\quad E \vdash \mathbf{id} \rhd E$$

$$\text{shift}\quad A, E \vdash \uparrow \rhd E$$

$$\text{cons}\quad \frac{E \vdash a : A \qquad E \vdash s \rhd E'}{E \vdash (a : A \cdot s) \rhd A, E'}$$

$$\text{comp}\quad \frac{E \vdash s'' \rhd E'' \qquad E'' \vdash s' \rhd E'}{E \vdash s' \circ s'' \rhd E'}$$

En este cálculo se cumple que el tipo de un término se mantiene por reducción:

Teorema 33 (Subject Reduction)**.** Sean a y b términos, s y t sustituciones, E y E' bases,

$$a \to_{\lambda\sigma} b \quad \text{entonces} \quad E \vdash a : A \implies E \vdash b : A$$

$$s \to_{\lambda\sigma} t \quad \text{entonces} \quad E \vdash s \rhd E' \implies E \vdash t \rhd E'$$

Demostración. La demostración en [ACCL91] analiza todas las reglas del cálculo confirmando que se preserva el tipo para todas ellas. □

1.4.2 Propiedades de los cálculos de sustituciones explícitas

A la hora de estudiar cálculos con sustituciones explícitas resulta útil estudiar las siguientes propiedades:

Sea λ_Z un cálculo de sustituciones explícitas, Z su cálculo de sustituciones asociado, y to_Z una función que traduce un λ-término en un λ_Z-término,

($\mathbf{CR}_Z$) El cálculo Z es confluente.

($\mathbf{SN}_Z$) El cálculo Z es fuertemente normalizante.

(CR) La relación $\to_{\lambda_Z}$ es confluente (en términos cerrados).

(MC) La relación $\to_{\lambda_Z}$ es confluente en *términos abiertos* (términos con metavariables).

(PSN) La relación $\to_{\lambda_Z}$ *preserva la normalización fuerte* de la β-reducción: si $t \in \mathcal{SN}_\lambda$, entonces $\mathrm{to}_Z(t) \in \mathcal{SN}_{\lambda_Z}$

(SNt) La relación $\to_{\lambda_Z}$ es fuertemente normalizante para todos los términos tipados de λ_Z.

(Sim) La relación $\to_{\lambda_Z}$ *simula* la β-reducción: si $t \to_\beta t'$, entonces $\mathrm{to}_Z(t) \twoheadrightarrow_{\lambda_Z} \mathrm{to}_Z(t')$.

En [ACCL91] se muestra que $\lambda\sigma$ cumple $\mathbf{CR}_\sigma$, $\mathbf{SN}_\sigma$, **CR**, **Sim**. Una prueba directa de $\mathbf{SN}_\sigma$ se encuentra en [Zan94], y será presentada en el capítulo 3.

Sin embargo, $\lambda\sigma$ no cumple con **MC**, **SNt**, **PSN**:

Observación 34. El cálculo $\lambda\sigma$ no es metaconfluente, *i.e.* confluente en los términos extendidos con metavariables:

$$a ::= X \mid 1 \mid a\ a \mid \lambda a \mid a[s]$$

$$s ::= x \mid \mathbf{id} \mid \uparrow \mid a \cdot s \mid s \circ s$$

Demostración. Contraejemplo:

$$a[b[s] \cdot s] \twoheadleftarrow ((\lambda a)\ b)[s] \twoheadrightarrow a[b[s] \cdot (s \circ \mathbf{id})]$$

Si $a = X, b = Y, s = x$ se tienen dos formas normales distintas. □

Teorema 35. En el cálculo $\lambda\sigma$ no todo término tipable es fuertemente normalizante. Además, $\lambda\sigma$ no preserva la normalización fuerte.

Demostración. En [Mel95], Melliès demuestra que el término

$$T = (\lambda x.(\lambda y.y)\ ((\lambda z.z)\ x))\ ((\lambda v.v)\ w)$$

es tipable en λ cálculo (que por el teorema 30 no tiene derivaciones infinitas), y sin embargo su traducción a $\lambda\sigma$ también es tipable, pero acepta una derivación infinita. □

1.4.3 Definiciones

En esta subsección se presentarán algunas definiciones y propiedades que permitirán el estudio en detalle de $\lambda\sigma$ y $\lambda\sigma_{\mathrm{gc}}$.

Notación (σ-forma normal de un término $\lambda\sigma$)**.** Dado un término t de $\lambda\sigma$, nos referimos como $\sigma(t)$ a la única σ-forma normal del término. Es única por ser σ SN y CR (ver subsección 1.4.2), y por lema 6.

Lema 36 (Caracterización de las formas normales)**.** Las σ-f.n. están dadas por

$$a ::= 1 \mid 1[\uparrow^n] \mid a\ a \mid \lambda a$$

$$s ::= \mathbf{id} \mid \ \uparrow^n \ \mid a \cdot s$$

Demostración. La prueba se encuentra en [Rio93], y una variación de la prueba se utiliza para demostrar el lema 62. □

Peso de una sustitución

Para caracterizar a las sustituciones, nos va a ser útil la presente definición de peso de una sustitución, obtenida de [ACCL91] :

$$|\cdot| : \text{Sustitución} \longrightarrow (\mathbb{N} \times \mathbb{N})$$

$$\begin{array}{lcll} |\mathbf{id}| & = & (0,0) & \\ |\uparrow| & = & (0,1) & \\ |a \cdot s| & = & (m+1,n) & \text{con } |s| = (m,n) \\ |s \circ t| & = & (m+p-n,q) & \text{con } |s| = (m,n), |t| = (p,q), p \geq n \\ |s \circ t| & = & (m,q+n-p) & \text{con } |s| = (m,n), |t| = (p,q), p < n \end{array}$$

La idea intuitiva de la definición de peso se aprecia por este lema:

Lema 37. Sea s una sustitución, $|s| = (m,n)$ sii existen términos $a_1, \ldots, a_m$ tales que

$$\sigma(s) = a_1 \cdot a_2 \cdot \ldots \cdot a_m \cdot \uparrow^n$$

Demostración. Por inducción en s. Esta prueba es muy similar a la prueba del lema 68, que sostiene la misma propiedad para el cálculo σ_{gc}. □

El siguiente lema permite mostrar que la σ-reducción no modifica el peso de una sustitución:

Lema 38 (Preservación del peso por reducción)**.** Para dos sustituciones s y t,

$$s \twoheadrightarrow_{\lambda\sigma} t \implies |s| = |t|$$

Demostración. Por inducción en la longitud de la derivación, y para el caso base por inducción en s. Esta prueba es muy similar a la prueba del corolario 64, que sostiene la misma propiedad para el cálculo σ_{gc} □

Variables libres de un término

No hemos encontrado en la literatura una definición formal de variables libres de un término $\lambda\sigma$, por lo que aquí presentamos nuestra definición. En la misma, hacemos uso de la siguiente notación para un conjunto de naturales C, un natural n, y un entero m:

$$\begin{aligned} C_{>n} &= \{i \mid i \in C \wedge i > n\} \\ C + m &= \{i + m \mid i \in C \wedge i + m > 0\} \end{aligned}$$

Observación 39. Para C un conjunto de naturales y m, n dos naturales,

1. $(C + m) - n = C + (m - n)$
2. Para $m - n \geq 0$, $(C + n)_{>m} = C_{>m-n} + n$

Demostración.

1. Se demuestra de forma directa

$$\begin{aligned} (C + m) - n &= \{i + m \mid i \in C \wedge i + m > 0\} - n \\ &= \{j - n \mid j \in \{i + m \mid i \in C \wedge i + m > 0\} \wedge j - n > 0\} \\ &= \{(i + m) - n \mid i \in C \wedge i + m > 0 \wedge (i + m) - n > 0\} \\ &= \{(i + m) - n \mid i \in C \wedge (i + m) - n > 0\} \\ &= \{i + (m - n) \mid i \in C \wedge i + (m - n) > 0\} \\ &= C + (m - n) \end{aligned}$$

2. También se demuestra de forma directa

$$\begin{aligned} (C + n)_{>m} &= \{i + n \mid i \in C\}_{>m} \\ &= \{j \mid j \in \{i + n \mid i \in C \wedge i + n > 0\} \ \wedge \ j > m\} \\ &= \{i + n \mid i \in C \ \wedge \ i + n > m\} \\ &= \{i + n \mid i \in C \ \wedge \ i > m - n\} \\ &= \{i + n \mid i \in C_{>m-n}\} \\ &= C_{>m-n} + n \end{aligned}$$

□

Definición 40 (Variables libres de un término)**.**

$$\begin{aligned} \mathrm{FV}(1) &= \{1\} \\ \mathrm{FV}(a\ b) &= \mathrm{FV}(a) \cup \mathrm{FV}(b) \\ \mathrm{FV}(\lambda a) &= \mathrm{FV}(a) - 1 \\ \mathrm{FV}(a[s]) &= (\mathrm{FV}(a)_{>m} + n - m) \cup \mathrm{FV}(s) \quad \text{con } |s| = (m, n) \\ \mathrm{FV}(\mathbf{id}) &= \emptyset \\ \mathrm{FV}(\uparrow) &= \emptyset \\ \mathrm{FV}(a \cdot s) &= \mathrm{FV}(a) \cup \mathrm{FV}(s) \\ \mathrm{FV}(s \circ t) &= (\mathrm{FV}(s)_{>m} + n - m) \cup \mathrm{FV}(t) \quad \text{con } |t| = (m, n) \end{aligned}$$

Lema 41 (Preservación de las variables libres por reducción)**.** Sean a y b dos términos,

$$a \twoheadrightarrow_{\lambda\sigma} b \implies \mathrm{FV}(b) \subseteq \mathrm{FV}(a)$$

Demostración. Por inducción en la cantidad de reducciones, y en el caso base en el término a. Esta prueba es muy similar a la prueba del corolario 66, que sostiene la misma propiedad para el cálculo σ_{gc}. □

Observación 42. Sean a un término, s y t sustituciones. Entonces,

$$\text{FV}(a[s][t]) = \text{FV}(a[s \circ t])$$

Demostración. Sean $|s| = (m, n)$ y $|t| = (p, q)$

$$\begin{aligned}
\text{FV}(a[s][t]) &= (\text{FV}(a[s])_{>p} + q - p) \cup \text{FV}(t) \\
&= (((\text{FV}(a)_{>m} + n - m) \cup \text{FV}(s))_{>p} + q - p) \cup \text{FV}(t) \\
&= ((\text{FV}(a)_{>m} + n - m)_{>p} + q - p) \cup \\
&\qquad \cup (\text{FV}(s)_{>p} + q - p) \cup \text{FV}(t)
\end{aligned}$$

Dividimos en casos:

- $p \geq n$: $|s \circ t| = (m + p - n, q)$

$$\begin{aligned}
\text{FV}(a[s][t]) &= ((\text{FV}(a)_{>m} + n - m)_{>p} + q - p) \cup \\
&\qquad \cup (\text{FV}(s)_{>p} + q - p) \cup \text{FV}(t) \\
&=_{\text{Obs 39}} ((\text{FV}(a)_{>m})_{>p-n+m} + q - p + n - m) \cup \\
&\qquad \cup (\text{FV}(s)_{>p} + q - p) \cup \text{FV}(t) \\
&= (\text{FV}(a)_{>p-n+m} + q - p + n - m) \cup \\
&\qquad \cup (\text{FV}(s)_{>p} + q - p) \cup \text{FV}(t)
\end{aligned}$$

$$\begin{aligned}
\text{FV}(a[s \circ t]) &= (\text{FV}(a)_{>m+p-n} + q - (m + p - n)) \cup \\
&\qquad \cup \text{FV}(s \circ t) \\
&= (\text{FV}(a)_{>p-n+m} + q - p + n - m) \cup \\
&\qquad \cup (\text{FV}(s)_{>p} + q - p) \cup \text{FV}(t)
\end{aligned}$$

- $p < n$: $|s \circ t| = (m, q + n - p)$

$$\begin{aligned}
\text{FV}(a[s][t]) &= ((\text{FV}(a)_{>m} + n - m)_{>p} + q - p) \cup \\
&\qquad \cup (\text{FV}(s)_{>p} + q - p) \cup \text{FV}(t) \\
&= (\text{FV}(a)_{>m} + n - m + q - p) \cup \\
&\qquad \cup (\text{FV}(s)_{>p} + q - p) \cup \text{FV}(t)
\end{aligned}$$

$$\begin{aligned}
\text{FV}(a[s \circ t]) &= (\text{FV}(a)_{>m} + q + n - p - m) \cup \\
&\qquad \cup \text{FV}(s \circ t) \\
&= ((\text{FV}(a)_{>m} + n - m + q - p) \cup \\
&\qquad \cup (\text{FV}(s)_{>p} + q - p) \cup \text{FV}(t)
\end{aligned}$$

En ambos casos se concluye que $\text{FV}(a[s][t]) = \text{FV}(a[s \circ t])$. □

(B)	$(\lambda x.t)\ u$	$\longrightarrow$	$t[x/u]$	
	$x[x/u]$	$\longrightarrow$	u	
	$y[x/u]$	$\longrightarrow$	y	$(x \neq y)$
	$(t\ u)[x/v]$	$\longrightarrow$	$t[x/v]\ u[x/v]$	
	$(\lambda y.t)[x/v]$	$\longrightarrow$	$\lambda y.t[x/v]$	

Figura 1.3: Cálculo λx

1.5 Cálculo λx

En [Ros93] se presenta λx, el λ-cálculo con sustituciones explícitas más simple que se puede definir, puesto que introduce la sustitución por reglas idénticas a la del cálculo tradicional.

La figura 1.3 muestra dicho cálculo.

Este cálculo cumple todas las propiedades mencionadas en 1.4.2, excepto metaconfluencia, como lo demuestra el siguiente contraejemplo:

$$t[y/v][x/u[y/v]] \twoheadleftarrow ((\lambda x.t)\ u)[y/v] \twoheadrightarrow t[x/u][y/v]$$

1.6 Cálculo λes

En [Kes07], la autora presenta un cálculo lambda con sustituciones explícitas y variables nombradas denominado λes. Dicho cálculo posee todas las propiedades mencionadas en 1.4.2

Los términos de λes son los mismos que los de λx:

$$t ::= x \mid t\ t' \mid \lambda x \mid t[x/t']$$

donde x pertenece a un conjunto enumerable de variables.

El cálculo se encuentra resumido en la figura 1.4. En él podemos apreciar una regla denominada *Gc* (*Garbage Collection*), es decir, "recolectora de basura". Esto es interpretado de la siguiente forma: si la sustitución x/u no va a modificar el término t, entonces es basura y debe ser eliminada. En este caso se cumple cuando x no aparece libre en t. Esta regla fue originalmente concebida para el cálculo λxgc[BR95].

Para el cálculo λes, el motivo de esta "limpieza" es para evitar que futuras composiciones (ver reglas $Comp_1$ y $Comp_2$) no compongan elementos inecesarios, que puedan provocar pérdidas en las propiedades mencionadas. Este tipo de tratamiento diferencial en las sustituciones cuyas variables a sustituir no aparecen en la variables libres de los términos a sustituir se aprecia también en el cálculo *λlxr* [KL05], quizás un precursor de λes.

Otra particularidad del cálculo es la regla de equivalencia C, con lo que la relación $t \to_{\lambda es} t'$ es interpretada en el conjunto de términos *módulo C*.

Como la aplicación de las reglas se encuentra condicionada al conocimiento de las variables libres del término, la *metaconfluencia* se consigue para metavariables anotadas con un conjunto de variables libres, es decir, para X una metavarible, y Δ un conjunto de variables, $\mathrm{FV}(X_\Delta) = \Delta$.

(C)	$t[x/u][y/v]$	$=$	$t[y/v][x/u]$	$(y \notin \mathrm{FV}(u) \wedge x \notin \mathrm{FV}(v))$
(B)	$(\lambda x.t)\ u$	$\longrightarrow$	$t[x/u]$	
(Var)	$x[x/u]$	$\longrightarrow$	u	
(Gc)	$t[x/u]$	$\longrightarrow$	t	$(x \notin \mathrm{FV}(t))$
(App_1)	$(t\ u)[x/v]$	$\longrightarrow$	$t[x/v]\ u[x/v]$	$(x \in \mathrm{FV}(t) \wedge x \in \mathrm{FV}(u))$
(App_2)	$(t\ u)[x/v]$	$\longrightarrow$	$t\ u[x/v]$	$(x \notin \mathrm{FV}(t) \wedge x \in \mathrm{FV}(u))$
(App_3)	$(t\ u)[x/v]$	$\longrightarrow$	$t[x/v]\ u$	$(x \in \mathrm{FV}(t) \wedge x \notin \mathrm{FV}(u))$
(Lamb)	$(\lambda y.t)[x/v]$	$\longrightarrow$	$\lambda y.t[x/v]$	
(Comp_1)	$t[x/u][y/v]$	$\longrightarrow$	$t[y/v][x/u[y/v]]$	$(y \in \mathrm{FV}(u) \wedge y \in \mathrm{FV}(t))$
(Comp_2)	$t[x/u][y/v]$	$\longrightarrow$	$t[x/u[y/v]]$	$(y \in \mathrm{FV}(u) \wedge y \notin \mathrm{FV}(t))$

Figura 1.4: Cálculo λes

Capítulo 2

Cálculo $\lambda\sigma_{\mathbf{gc}}$

En este capítulo presentaremos un cálculo que intenta ser un puente entre los cálculos λes y $\lambda\sigma$. Primero, en la sección 2.1, discutiremos cómo definir la "basura" en el cálculo $\lambda\sigma$, y daremos una definición. Luego, en la sección 2.2 mostraremos cómo esta definición interactúa con el cálculo $\lambda\sigma$. Finalmente, en la sección 2.3 presentaremos al cálculo $\lambda\sigma_{\mathrm{gc}}$.

2.1 Afectar o no afectar un término

En esta sección veremos los elementos a tener en cuenta a la hora de definir "basura" en un cálculo como $\lambda\sigma$. Recordemos la regla Gc para λes:

$$(\mathrm{GC}) \quad t[x/u] \quad \longrightarrow \quad t \quad (x \notin \mathrm{FV}(t))$$

Nosotros queremos una regla similar en $\lambda\sigma$:

$$(\mathrm{GC}) \quad a[s] \quad \longrightarrow \quad a \quad (\textit{condición})$$

Como se mostró en la sección 1.4, en $\lambda\sigma$ una sustitución s con peso (m, n) reemplaza los primeros m índices por los m términos de la sustitución. Por ello, es necesario condicionar la regla a que no haya variables libres en a que sean menores o igual a m. Dicho de otra manera, $(\forall i \in \mathrm{FV}(a))\ i > m$. Además, vamos a necesitar que ninguna variable sea actualizada, es decir, m deberá ser igual a n.

Cuando estas condiciones ocurren, decimos que:

Definición 43. Una sustitución s *no afecta* a un término a si

$$(|s| = (m, m) \wedge (\forall p \in \mathrm{FV}(a))\ p > m) \ \vee \ \mathrm{FV}(a) = \emptyset$$

En ese caso, notamos $s \not\triangleright a$.

Para el caso contrario, decimos que:

Definición 44. Se dice que una sustitución s *afecta* a un término a si

$$|s| = (m, n) \wedge (m \neq n \vee (\exists p \in \mathrm{FV}(a))\ p \leq m) \wedge \mathrm{FV}(a) \neq \emptyset$$

En ese caso, notamos $s \triangleright a$.

Notar que $1\cdot \uparrow \ \not\triangleright\ 1$. La idea intuitiva es que, aunque la sustitución $1\cdot \uparrow$ simule la sustitución **id**, está reemplazando la variable 1 por otro 1.

Definimos entonces la regla GC para $\lambda\sigma$ como

$$\text{(GC)} \quad a[s] \quad \longrightarrow \quad a \quad (\text{s } \not\triangleright \text{ a})$$

Esta definición de "basura" es, en efecto, muy restrictiva, en tanto que no siempre que se aplica GC en λes se puede aplicar GC en $\lambda\sigma$, como se puede apreciar en la reducción del término t del siguiente ejemplo:

$$t = (\lambda u.(\lambda v.y))\ x \quad \rightarrow_{Beta(\lambda\text{es})} \quad (\lambda v.y)[u/x] \quad \rightarrow_{GC(\lambda\text{es})} \quad (\lambda v.y)$$

En $\lambda\sigma$, ordenando las variables como $x, y, z, u, v, \ldots$, t se traduce [1] como

$$(\lambda\lambda 4)\ 1 \quad \rightarrow_{Beta(\lambda_{\text{DB}})} \quad (\lambda 4)[1 \cdot \mathbf{id}]$$

que se encuentra en $GC(\lambda\sigma)$-forma normal. El inconveniente radica, como se mencionó previamente, en que la variable libre del término $(\lambda 4)$ debe ser decrementada, puesto que las sustituciones provienen de una Beta-reducción (con la correspondiente elminación de un λ).

En [Rio93] y en [VARK07] aparece el problema de la actualización de índices, al tratar con la regla η. Esta regla, en el λ-cálculo, se define como

$$(\eta) \quad \lambda x.M\ x \quad = \quad M \quad (x \notin \text{FV}(M))$$

Para $\lambda\sigma$, en [Rio93] esto se traduce como

$$\text{(Eta)} \quad \lambda(a\ 1) \quad \longrightarrow \quad b \quad (\sigma(a) = \sigma(b[\uparrow]))$$

Es decir, el resultado de aplicar la regla debe computarse como reducir en 1 todos los índices libres de a, aunque no se encuentra explicitado cómo se debe realizar. En [VARK07] se comentan varios inconvenientes surgidos por esta falta de explicitación (por ejemplo, la versión tipada pierde la propiedad Subject Reduction), y como solución, sugieren una versión *constructiva* de la regla. Esta versión incluye un subcálculo ($\eta_{\lambda\sigma}$) cuya finalidad es realizar la actualización y verificar que la variable 1 no se encuentre libre en a. Para ello se introduce el operador η^i_j. Luego, la regla queda como

$$\text{(Eta)} \quad \lambda(a\ 1) \quad \longrightarrow \quad \eta_{\lambda\sigma}(a[\eta^1_1]) \quad (\text{si } \eta_{\lambda\sigma}(a[\eta^1_1]) \text{ es un término } \lambda\sigma)$$

Esta misma idea podría aplicarse para hacer más general la regla GC. Para ello se podría definir un subcálculo θ que en base a un operador θ^i_j actualice en $i-j$ las variables libres mayores a j del término al que se aplica. Luego, la regla GC quedaría como

$$\text{(GC)} \quad a[s] \quad \longrightarrow \quad \theta(a[\theta^n_m]) \quad (\text{si } |s| = (m,n) \wedge (\forall i \in \text{FV}(a))\ i > m)$$

Aún así, la definición de "basura" no es equivalente a la de λes, como lo muestra el siguiente ejemplo: el término $a = x[y/b]$ se traduce (siguiendo el orden lexicográfico de las variables) a $a' = 1[1 \cdot b'\cdot \uparrow]$ para b' la traducción del término b. Si bien en a se puede aplicar la regla GC, en a' no se puede aplicar.

[1] Para este ejemplo, nos basta con traducir primero de λ a λ_{DB}, y luego utilizar la convención $i = 1[\uparrow^{i-1}]$ para cada índice i. La traducción de los términos de λes a $\lambda\sigma$ es compleja y no será tratada en este trabajo.

Una forma de evitar esto es generalizando la restricción $(\forall i \in \mathrm{FV}(a))\ i > m$ como

$$(\forall i \in \mathrm{FV}(a))\ i > m \ \vee \ (i \leq m \wedge i[s] \twoheadrightarrow_\sigma i)$$

Cada evaluación de una posible aplicación de la regla debe calcular el valor $\sigma(i[s])$ para cada variable libre i, lo que hace que esta opción sea costosa desde el punto de vista práctico.

Nosotros vamos a estudiar en este trabajo la primer definición, dejando para el futuro el estudio de las otras alternativas.

2.2 Relación entre GC y $\lambda\sigma$

En esta sección mostraremos cuál es la relación entre los términos $a[s]$ y a cuando $s \Vdash a$ en $\lambda\sigma$.

Para ello, primero vamos a demostrar un lema que será utilizado frecuentemente durante este trabajo:

Lema 45. Sean a, b términos y s una sustitución. Entonces,

$$s \Vdash a \ \wedge \ s \Vdash b \iff s \Vdash (a\ b)$$

Demostración. Sea $|s| = (m, n)$. Por definición de FV, $\mathrm{FV}(a\ b) = \mathrm{FV}(a) \cup \mathrm{FV}(b)$.

$\Rightarrow$) Tenemos que

$$\mathrm{FV}(a) = \emptyset \ \vee \ m = n \wedge (\forall i \in \mathrm{FV}(a))\ i > m$$
$$\mathrm{FV}(b) = \emptyset \ \vee \ m = n \wedge (\forall i \in \mathrm{FV}(b))\ i > m$$

Queremos ver que

$$\mathrm{FV}(a\ b) = \emptyset \ \vee \ m = n \wedge (\forall i \in \mathrm{FV}(a\ b))\ i > m$$

Separamos en los cuatro casos:

(a) $\mathrm{FV}(a) = \emptyset \wedge \mathrm{FV}(b) = \emptyset$: $\mathrm{FV}(a\ b) = \emptyset$.

(b) $\mathrm{FV}(a) = \emptyset \wedge \mathrm{FV}(b) \neq \emptyset$:

$$\begin{aligned}(\forall i \in \mathrm{FV}(b))\ i > m &\iff (\forall i \in \mathrm{FV}(a) \cup \mathrm{FV}(b))\ i > m \\ &\iff (\forall i \in \mathrm{FV}(a\ b))\ i > m\end{aligned}$$

y $m = n$

(c) $\mathrm{FV}(a) \neq \emptyset \wedge \mathrm{FV}(b) = \emptyset$: Idem anterior.

(d) $\mathrm{FV}(a) \neq \emptyset \wedge \mathrm{FV}(b) \neq \emptyset$:

$$(\forall i \in \mathrm{FV}(a))\ i > m \wedge (\forall i \in \mathrm{FV}(b))\ i > m \implies (\forall i \in \mathrm{FV}(a\ b))\ i > m$$

y $m = n$

⇐) Tenemos que

$$\mathrm{FV}(a\ b) = \emptyset\ \vee\ m = n \wedge (\forall i \in \mathrm{FV}(a\ b))\ i > m$$

Queremos ver que

$$\mathrm{FV}(a) = \emptyset\ \vee\ m = n \wedge (\forall i \in \mathrm{FV}(a))\ i > m$$
$$\mathrm{FV}(b) = \emptyset\ \vee\ m = n \wedge (\forall i \in \mathrm{FV}(b))\ i > m$$

Separamos en los dos casos:

(a) $\mathrm{FV}(a\ b) = \emptyset \iff \mathrm{FV}(a) \cup \mathrm{FV}(b) = \emptyset \implies \mathrm{FV}(a) = \emptyset \wedge \mathrm{FV}(b) = \emptyset$

(b) $\mathrm{FV}(a\ b) \neq \emptyset$: Luego $m = n$ y

$$\begin{aligned}(\forall i \in \mathrm{FV}(a\ b))\ i > m &\iff (\forall i \in \mathrm{FV}(a) \cup \mathrm{FV}(b))\ i > m \\ &\iff (\forall i \in \mathrm{FV}(a))\ i > m \wedge (\forall i \in \mathrm{FV}(b))\ i > m\end{aligned}$$

□

Observación 46. $s \Vdash a\ \vee\ s \Vdash b \iff s \Vdash (a\ b)$.

El siguiente lema establece que, bajo σ-forma normal, una sustitución que no afecta a un término efectivamente reduce al mismo término.

Lema 47. Sean a un término en σ-forma normal, y s una sustitución tal que $s \nVdash a$. Entonces,

$$a[s] \twoheadrightarrow a$$

Demostración. Por inducción en la forma normal a:

- $a = 1$: s sólo puede ser **id** (notar que para cualquier sustitución t, $1 \cdot t \Vdash 1$), luego $a[\mathbf{id}]\ \to_{\mathrm{VarId}}\ a$.
- $a = 1[\uparrow^n]$: Sólo puede suceder que $|s| = (m, m)$ con $m \leq n$. Por lemas 38, y 37, existen términos $b_1, \ldots, b_m$ tales que

  $$\sigma(s) = b_1 \cdot \ldots \cdot b_m \cdot \uparrow^m$$

  Luego,

  $$\begin{aligned}1[\uparrow^n][s] &\to_{\sigma} 1[\uparrow^n][b_1 \cdot \ldots \cdot b_m \cdot \uparrow^m] \\ &\to_{\mathrm{Clos}} 1[\uparrow^n \circ (b_1 \cdot \ldots \cdot b_m \cdot \uparrow^m)] \\ &\to_{\mathrm{Ass}^n} 1[(\uparrow \circ (\uparrow \circ \ldots (\uparrow \circ (b_1 \cdot \ldots \cdot b_m \cdot \uparrow^m)) \ldots))] \\ &\to_{\mathrm{ShiftCons}^m} 1[(\uparrow \circ \ldots (\uparrow \circ \uparrow^m) \ldots)] \\ &= 1[\uparrow^{n-m+m}] \\ &= 1[\uparrow^n]\end{aligned}$$

  Notar que se pueden realizar m pasos de *ShiftCons* por ser $n \geq m$. Con esto se eliminan m $\uparrow$'s, que se recuperan de los m $\uparrow$'s de s, con lo que efectivamente quedan n.

- $a = b\ c$: Por lema 45, $s \Subset b$ y $s \Subset c$, luego

$$(b\ c)[s] \ \to_{\text{App}}\ b[s]\ c[s] \ \twoheadrightarrow_{\text{h.i.}}\ b\ c$$

- $a = \lambda b$: No siempre puede aplicarse la hipótesis inductiva, con lo que debe utilizarse otra técnica. La demostración de este caso es el resultado principal del apéndice A.

□

A continuación presentamos el resultado principal de esta sección, que permite mostrar la relación existente entre un término a y una sustitución s cuando s no afecta a a.

Corolario 48 (Garbage Collection)**.** Si $s \Subset a$, entonces

$$a[s] \twoheadleftarrow\!\!\!\twoheadrightarrow a$$

Demostración. Por lema 41, $s \Subset a \implies s \Subset \sigma(a)$. Luego,

$$a[s] \quad \twoheadrightarrow_{\sigma} \quad \sigma(a)[s] \quad \twoheadrightarrow_{\text{L 47}} \quad \sigma(a) \quad \twoheadleftarrow_{\sigma} \quad a$$

□

Nota 1. No es cierto en general que si $s \Subset a$, entonces $a[s] \twoheadrightarrow a$.

Demostración. Tomar $a = 3[2{\cdot}\mathbf{id}]$ y $s = 1{\cdot}\uparrow$. Es fácil notar que $|s| = (1, 1)$, y que las variables libres de a son $\{2\}$, con lo que $s \Subset a$. En el apéndice B se muestra un programa que busca ocurrencias de a en todas las posibles derivaciones del término $a[s]$, sin encontrarlas, demostrando que efectivamente $a[s] \not\twoheadrightarrow a$.

□

2.3 Cálculo $\lambda\sigma_{\mathbf{gc}}$

En esta sección se muestra el cálculo que será estudiado en este trabajo, $\lambda\sigma_{\text{gc}}$. Este cálculo incluye a GC como una regla de reescritura dentro del cálculo, y evita la distribución de la "basura".

Los términos del cálculo son los mismos que los de $\lambda\sigma$:

Términos	$\Lambda\sigma^{t}_{\text{gc}}$ { $a ::=$	$1 \mid a\ a \mid \lambda a \mid a[s]$
Sustituciones	$\Lambda\sigma^{s}_{\text{gc}}$ { $s ::=$	$\mathbf{id} \mid \uparrow \mid a \cdot s \mid s \circ s$

El $\lambda\sigma_{\text{gc}}$-cálculo es el TRS conformado por las reglas mostradas en la figura 2.1. Notar que, a diferencia del cálculo $\lambda\sigma$, este TRS tiene infinitas reglas. Por ejemplo, la regla GC es en realidad una regla-esquema, que define una regla para cada a y s que cumplan la condición $s \Subset a$. Cabe aclarar que en este trabajo sólo se trabajará con términos cerrados, es decir, sin metavariables.

Esta definición de TRS con condiciones es diferente a la mencionada en la literatura como "Conditional TRS" (por ejemplo, [BK86]), donde las condiciones se encuentran definidas en términos del propio sistema de reescritura, como en el siguiente ejemplo ([BK86]):

(Beta)	$(\lambda a)\ b$	$\longrightarrow$	$a[b \cdot \mathbf{id}]$	
(GC)	$a[s]$	$\longrightarrow$	a	(si $s \not\Vdash a$)
(VarCons)	$1[a \cdot s]$	$\longrightarrow$	a	
(App_1)	$(a\ b)[s]$	$\longrightarrow$	$a[s]\ b[s]$	(si $s \Vdash a$ y $s \Vdash b$)
(App_2)	$(a\ b)[s]$	$\longrightarrow$	$a[s]\ b$	(si $s \Vdash a$ y $s \not\Vdash b$)
(App_3)	$(a\ b)[s]$	$\longrightarrow$	$a\ b[s]$	(si $s \not\Vdash a$ y $s \Vdash b$)
(Abs)	$(\lambda a)[s]$	$\longrightarrow$	$\lambda(a[1 \cdot (s \circ \uparrow)])$	(si $s \Vdash \lambda a$)
(Clos)	$a[s][t]$	$\longrightarrow$	$a[s \circ t]$	(si $s \Vdash a$ y $t \Vdash a[s]$ y $s \circ t \Vdash a$)
(ClosGC)	$a[s][t]$	$\longrightarrow$	a	(si $s \Vdash a$ y $t \Vdash a[s]$ y $s \circ t \not\Vdash a$)
(IdL)	$\mathbf{id} \circ s$	$\longrightarrow$	s	
(ShiftId)	$\uparrow \circ \mathbf{id}$	$\longrightarrow$	$\uparrow$	
(ShiftCons)	$\uparrow \circ (a \cdot s)$	$\longrightarrow$	s	
(Map)	$(a \cdot s) \circ t$	$\longrightarrow$	$a[t] \cdot (s \circ t)$	(si $t \Vdash a$)
(MapGC)	$(a \cdot s) \circ t$	$\longrightarrow$	$a \cdot (s \circ t)$	(si $t \not\Vdash a$)
(Ass)	$(s \circ t) \circ u$	$\longrightarrow$	$s \circ (t \circ u)$	

Figura 2.1: Cálculo $\lambda\sigma_{\mathrm{gc}}$

$S\ x\ y\ z$	$\longrightarrow$	$x\ z\ (y\ z)$	(si $x \twoheadrightarrow I \wedge y \twoheadrightarrow I \wedge z \twoheadrightarrow I$)
$K\ x\ y$	$\longrightarrow$	x	(si $x \twoheadrightarrow I \wedge y \twoheadrightarrow I$)
$I\ x$	$\longrightarrow$	x	(si $x \twoheadrightarrow I$)

Salvo indicación contraria, no se hará distinción entre regla y regla-esquema.

Se llama σ_{gc} al cálculo de sustituciones asociado, es decir, $\lambda\sigma_{\mathrm{gc}} - \{\mathrm{Beta}\}$.

Discusión acerca de las reglas de $\lambda\sigma_{\mathbf{gc}}$

El cálculo $\lambda\sigma_{\mathrm{gc}}$ tiene como principal objetivo evitar la propagación y generación de "basura", por ser motivo de pérdida de la propiedad PSN.

1. La regla *GC* generaliza a la regla *VarId* de $\lambda\sigma$.

2. La regla *Map* de $\lambda\sigma$ se divide en *Map* y *MapGC*, con el objetivo de evitar que la "basura" sea distribuida.

3. Lo mismo ocurre para la regla *App*, que se divide en las tres posibilidades de distribución a los términos a y b. Notar que la opción de no distribuir ni a a ni a b es llevada a cabo por la regla *GC*.

4. La regla *Abs* no se aplica a "basura". En el caso de tener basura, de debe aplicar la regla *GC*.

5. La regla *Clos* no permite componer "basura". Si alguna de las sustituciones es basura, deberá ser eliminada por la regla *GC*. Además, se evita la generación de basura, como lo muestra el siguiente ejemplo: Sea $b = 2$, $s = 1 \cdot \mathbf{id}$, $t = \uparrow$. Es claro que $s \Vdash b$ y $t \Vdash b[s]$. Pero $s \circ t \not\Vdash b$, pues $|s \circ t| = |(1 \cdot \mathbf{id}) \circ \uparrow| = (1, 1)$, y $\mathrm{FV}(b) = 2$. Por ello se define la regla *ClosGC*, que elimina la basura producto de la composición.

Otras opciones pueden estudiarse para las reglas, como la eliminación de la regla *ClosGC* (que no figura en λes). Cabe aclarar que en ese caso las demostraciones de los siguientes capítulos deben ser replanteadas (esto será señalado en el momento que corresponda).

Capítulo 3

Normalización fuerte de $\sigma_{\mathbf{gc}}$

En [Rio93] y [CHR92] se demuestra la normalización fuerte del cálculo σ utilizando un cálculo intermedio (σ_0), en el cual SN de este cálculo implica SN de σ. En estos trabajos la prueba de SN de σ_0 es compleja. Hans Zantema en [Zan94] presenta una prueba sencilla, mediante la técnica de *Distribution Elimination*. Nosotros reproducimos esta prueba aquí para demostrar que σ_{gc0}, una variante de σ_{gc}, es SN.

Zantema presenta otra prueba de terminación fuerte de σ_0 en [Ter03] y en [Zan00], mediante la técnica de *Semantic Labelling*. Esta prueba también puede ser adaptada para demostar terminación de σ_{gc0}.

Los pasos de la demostración serán:

1. En 3.1 presentamos σ_{gc0}, una variante simplificada de σ_{gc}.
2. En 3.2 mostramos que SN de σ_{gc0} implica SN de σ_{gc}.
3. En 3.3 presentamos al cálculo $AMAs$, una variante simplificada de σ_{gc0}. En esta sección mostramos que SN de $AMAs$ implica SN de σ_{gc0}.
4. En 3.4 mostramos que $AMAs$ es SN, concluyendo que σ_{gc} es SN.

Presentamos una definición y un lema que permiten relacionar la terminación de dos cálculos:

Definición 49. Sean ψ y ω dos cálculos cuyos términos se encuentran definidos por los conjuntos Ψ y Ω, respectivamente. Sea F una función de Ψ a Ω, decimos que F es *estricta*[1] sii

$$s \to_\psi t \implies F(s) \xrightarrow{+}_\omega F(t)$$

Lema 50. Sean ψ y ω dos cálculos y F una función estricta del primero al segundo. Entonces

$$\omega \text{ es SN} \implies \psi \text{ es SN}$$

[1] Esta definición es distinta de la presentada en [Rio93], donde un paso de ψ se traduce a exactamente un paso de ω.

(K)	$s \circ t$	$\longrightarrow$	s
(VarCons)	$1 \circ (s \cdot t)$	$\longrightarrow$	s
(Abs)	$(\lambda s) \circ t$	$\longrightarrow$	$\lambda(s \circ (1 \cdot (t \circ \uparrow)))$
(IdL)	$\mathbf{id} \circ s$	$\longrightarrow$	s
(ShiftId)	$\uparrow \circ \mathbf{id}$	$\longrightarrow$	$\uparrow$
(ShiftCons)	$\uparrow \circ (s \cdot t)$	$\longrightarrow$	t
(Map)	$(s \cdot t) \circ u$	$\longrightarrow$	$(s \circ u) \cdot (t \circ u)$
(Ass)	$(s \circ t) \circ u$	$\longrightarrow$	$s \circ (t \circ u)$

Figura 3.1: El cálculo σ_{gc0}

Demostración. Suponemos que no. Entonces, existe una derivación infinita en ψ, tal que

$$s_0 \to_\psi s_1 \to_\psi s_2 \to_\psi \dots$$

Pero por definición de estricta,

$$F(s_0) \overset{+}{\to}_\omega F(s_1) \overset{+}{\to}_\omega F(s_2) \overset{+}{\to}_\omega \dots$$

con lo que se generó una derivación infinita en ω, lo que es absurdo. El absurdo provino de suponer la existencia de una derivación infinita en ψ, luego ψ es SN. □

3.1 Presentación de $\sigma_{\mathbf{gc0}}$

Vamos a extender el cálculo presentado en [Rio93] como σ_0, agregando la regla K (similar a la regla GC pero sin condición) y quitando la regla *VarId*. Este nuevo cálculo, denominado σ_{gc0}, no distingue términos de sustituciones, generalizando la sustitución y la composición en una misma operación.

Los términos se encuentran definidos como

$$\Lambda\sigma_{\mathrm{gc0}} \ \{ \quad s ::= 1 \mid \mathbf{id} \mid \ \uparrow \ \mid \lambda s \mid s \circ t \mid s \cdot t$$

Las reglas se encuentran definidas en la figura 3.1.

3.2 SN de $\sigma_{\mathbf{gc0}}$ implica SN de $\sigma_{\mathbf{gc}}$

Definición 51. Sea $F : \Lambda\sigma_{\mathrm{gc}} \to \Lambda\sigma_{\mathrm{gc0}}$ la misma función que la presentada en [Rio93]:

$$\begin{array}{lcllcl} F(1) & = & 1 & F(\mathbf{id}) & = & \mathbf{id} \\ F(a\ b) & = & F(a) \cdot F(b) & F(\uparrow) & = & \uparrow \\ F(\lambda a) & = & \lambda F(a) & F(a \cdot s) & = & F(a) \cdot F(s) \\ F(a[s]) & = & F(a) \circ F(s) & F(s \circ t) & = & F(s) \circ F(t) \end{array}$$

Lema 52. La función F es estricta.

Demostración. Sea $v \to_{\sigma_{\mathrm{gc}}} w$, vamos a mostrar por inducción en la longitud de v que $F(v) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(w)$:

- $v \in \{1, \mathbf{id}, \uparrow\}$. No hay reducción posible.
- $v = a\ b$:

  Pueden darse dos casos:

  - $w = a'\ b$, con $a \to_{\sigma_{\mathrm{gc}}} a'$: Por h.i., $F(a) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(a')$, luego

    $$F(a\ b) = F(a) \cdot F(b) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(a') \cdot F(b) = F(a'\ b)$$

  - $w = a\ b'$, con $a \to_{\sigma_{\mathrm{gc}}} a'$: Por h.i., $F(b) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(b')$, luego

    $$F(a\ b) = F(a) \cdot F(b) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(a) \cdot F(b') = F(a\ b')$$

- $v = \lambda a$: $w = \lambda a'$, con $a \to_{\sigma_{\mathrm{gc}}} a'$. Por h.i., $F(a) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(a')$, luego

  $$F(\lambda a) = \lambda F(a) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} \lambda F(a') = F(\lambda a')$$

- $v = a[s]$:

  Si la reducción no ocurre en la raíz, tenemos los siguientes dos casos:

  - $w = a'[s]$, con $a \to_{\sigma_{\mathrm{gc}}} a'$: Por h.i., $F(a) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(a')$, luego

    $$F(a[s]) = F(a) \circ F(s) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(a') \circ F(s) = F(a'[s])$$

  - $w = a[s']$, con $s \to_{\sigma_{\mathrm{gc}}} s'$: Por h.i., $F(s) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(s')$, luego

    $$F(a[s]) = F(a) \circ F(s) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(a) \circ F(s') = F(a[s'])$$

  Si la reducción ocurre en la raíz, hay que considerar todas las posibles reglas:

  - $w = a$, y $s \not\rhd a$:

    $$F(a[s]) = F(a) \circ F(s) \to_K F(a)$$

  - $v = 1[b \cdot s]$, $\quad w = b$:

    $$F(1[b \cdot s]) = F(1) \circ (F(b) \cdot F(s)) \to_{VarCons} F(b)$$

  - $v = (b\ c)[s]$, $\quad w = b[s]\ c[s]$, con $s \rhd b$ y $s \rhd c$:

    $$\begin{array}{rcl} F((b\ c)[s]) & = & (F(b) \cdot F(c)) \circ F(s) \\ & \to_{Map} & (F(b) \circ F(s)) \cdot (F(c) \circ F(s)) \\ & = & F(b[s]\ c[s]) \end{array}$$

  - $v = (b\ c)[s]$, $\quad w = b[s]\ c$, con $s \rhd b$ y $s \not\rhd c$:

    $$\begin{array}{rcl} F((b\ c)[s]) & = & (F(b) \cdot F(c)) \circ F(s) \\ & \to_{Map} & (F(b) \circ F(s)) \cdot (F(c) \circ F(s)) \\ & \to_K & (F(b) \circ F(s)) \cdot F(c) \\ & = & F(b[s]\ c) \end{array}$$

- $v = (b\ c)[s]$, $w = b\ c[s]$, con $s \not\Rightarrow b$ y $s \Rightarrow c$: Análogo al anterior.
- $v = (\lambda b)[s]$, $w = \lambda b[1 \cdot (s \circ \uparrow)]$, con $s \Rightarrow \lambda b$:

$$\begin{array}{rcl} F((\lambda b)[s]) & = & (\lambda F(b)) \circ F(s) \\ & \to_{Abs} & \lambda(F(b) \circ (1 \cdot (F(s) \circ \uparrow))) \\ & = & F(\lambda b[1 \cdot (s \circ \uparrow)]) \end{array}$$

- $v = b[t][s]$, $w = b[t \circ s]$, con $t \Rightarrow b$, $s \Rightarrow b[t]$ y $t \circ s \Rightarrow b$:

$$\begin{array}{rcl} F(b[t][s]) & = & (F(b) \circ F(t)) \circ F(s) \\ & \to_{Ass} & F(b) \circ (F(t) \circ F(s)) \\ & = & F(b[t \circ s]) \end{array}$$

- $v = b[t][s]$, $w = b$, con $t \Rightarrow b$, $s \Rightarrow b[t]$ y $t \circ s \not\Rightarrow b$:

$$\begin{array}{rcl} F(b[t][s]) & = & (F(b) \circ F(t)) \circ F(s) \\ & \to_{Ass} & F(b) \circ (F(t) \circ F(s)) \\ & \to_{K} & F(b) \end{array}$$

- $v = b \cdot s$:

  Pueden darse dos casos:

  - $w = b' \cdot s$, con $b \to_{\sigma_{\mathrm{gc}}} b'$: Por h.i., $F(b) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(b')$, luego

$$F(b \cdot s) = F(b) \cdot F(s) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(b') \cdot F(s) = F(b' \cdot s)$$

  - $w = b \cdot s'$, con $s \to_{\sigma_{\mathrm{gc}}} s'$: Por h.i., $F(s) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(s')$, luego

$$F(b \cdot s) = F(b) \cdot F(s) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(b) \cdot F(s') = F(b \cdot s')$$

- $v = s \circ t$:

  Si la reducción no ocurre en la raíz, tenemos dos casos:

  - $w = s' \circ t$, con $s \to_{\sigma_{\mathrm{gc}}} s'$: Por h.i., $F(s) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(s')$, luego

$$F(s \circ t) = F(s) \circ F(t) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(s') \circ F(t) = F(s' \circ t)$$

  - $w = s \circ t'$, con $t \to_{\sigma_{\mathrm{gc}}} t'$: Por h.i., $F(t) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(t')$, luego

$$F(s \circ t) = F(s) \circ F(t) \overset{+}{\to}_{\sigma_{\mathrm{gc0}}} F(s) \circ F(t') = F(s \circ t')$$

  Si la reducción ocurre en la raíz, entonces hay que analizar todas las reglas que se pueden aplicar:

  - $v = \mathbf{id} \circ s$, $w = s$:

$$F(\mathbf{id} \circ s) = \mathbf{id} \circ F(s) \to_{IdL} F(s)$$

- $v = \uparrow \circ \mathbf{id}, \quad w = \uparrow$

$$F(\uparrow \circ \mathbf{id}) = \uparrow \circ \mathbf{id} \to_{ShiftId} \uparrow = F(\uparrow)$$

- $v = \uparrow \circ (a \cdot s), \quad w = s$

$$F(\uparrow \circ (a \cdot s)) = \uparrow \circ (F(a) \cdot F(s)) \to_{ShiftCons} F(s)$$

- $v = (a \cdot s) \circ t, \quad w = a[t] \cdot (s \circ t)$, con $t \triangleright a$

$$\begin{array}{rcl} F((a \cdot s) \circ t) & = & (F(a) \cdot F(s)) \circ F(t) \\ & \to_{Map} & (F(a) \circ F(t)) \cdot (F(s) \circ F(t)) \\ & = & F(a[t] \cdot (s \circ t)) \end{array}$$

- $v = (a \cdot s) \circ t, \quad w = a \cdot (s \circ t)$, con $t \not\triangleright a$

$$\begin{array}{rcl} F((a \cdot s) \circ t) & = & (F(a) \cdot F(s)) \circ F(t) \\ & \to_{Map} & (F(a) \circ F(t)) \cdot (F(s) \circ F(t)) \\ & \to_{K} & F(a) \cdot (F(s) \circ F(t)) \\ & = & F(a \cdot (s \circ t)) \end{array}$$

- $v = (s \circ t) \circ u, \quad w = s \circ (t \circ u)$

$$F((s \circ t) \circ u) = (F(s) \circ F(t)) \circ F(u) \to_{Ass} F(s) \circ (F(t) \circ F(u)) = F(s \circ (t \circ u))$$

□

Proposición 53. Si σ_{gc0} es SN, entonces σ_{gc} es SN.

Demostración. Los lemas 50 y 52 nos lo garantizan. □

3.3 Presentación del cálculo *AMAs*

Definimos un nuevo cálculo *AMAs* que constará de las reglas[2] *Abs*, *Map*, y *Ass* de σ_{gc0}, y $Emb(\sigma_{\text{gc0}})$. Las reglas de reescritura quedan definidas como se muestra en la figura 3.2.

Proposición 54. Si *AMAs* es SN, entonces σ_{gc0} es SN.

Demostración. Es fácil notar que la función identidad es una función estricta entre σ_{gc0} y *AMAs*. Luego, por lema 50, probando SN de *AMAs* se obtiene SN de σ_{gc0}. □

3.4 *AMAs* es SN

Sea $AMAs_0$ el sub-TRS de *AMAs* conformado por las reglas *Abs*, *Map*, y *Ass*. Por el teorema 16, *AMAs* es totalmente terminante sii $AMAs_0$ es totalmente terminante.

Podemos notar que *Map* es una regla distribuidora para $\cdot$, luego por el teorema 20, $AMAs_0$ es totalmente terminante sii $Erase.(AMAs_0)$ es totalmente terminante.

El TRS $Erase.(AMAs_0)$ queda conformado por las reglas

[2] De allí su nombre: ***A**bs*, ***Map***, ***Ass***

(Abs)	$(\lambda s) \circ t$	$\longrightarrow$	$\lambda(s \circ (1 \cdot (t \circ \uparrow)))$
(Map)	$(s \cdot t) \circ u$	$\longrightarrow$	$(s \circ u) \cdot (t \circ u)$
(Ass)	$(s \circ t) \circ u$	$\longrightarrow$	$s \circ (t \circ u)$
	$s \circ t$	$\longrightarrow$	s
	$s \circ t$	$\longrightarrow$	t
	$s \cdot t$	$\longrightarrow$	s
	$s \cdot t$	$\longrightarrow$	t
	λs	$\longrightarrow$	s

Figura 3.2: Cálculo $AMAs$

(Abs1)	$(\lambda s) \circ t$	$\longrightarrow$	$\lambda(s \circ 1)$
(Abs2)	$(\lambda s) \circ t$	$\longrightarrow$	$\lambda(s \circ (t \circ \uparrow))$
(Ass)	$(s \circ t) \circ u$	$\longrightarrow$	$s \circ (t \circ u)$

Lema 55. *Erase.*$(AMAs_0)$ es totalmente terminante.

Demostración. Vamos a mostrar que este TRS es TT mostrando un álgebra monotona bien fundada y compatible.

Como conjunto soporte utilizaremos el espacio $A = (\mathbb{N} \times \mathbb{N} \times \mathbb{N})$, y como orden $>$ utilizamos el órden lexicográfico. La interpretación de los símbolos de función estará dada por:

$$\begin{aligned} 1_A = \uparrow_A &= (0,0,0) \\ \lambda_A(x_1,x_2,x_3) &= (x_1+1, x_2, x_3) \\ (x_1,x_2,x_3) \circ_A (y_1,y_2,y_3) &= (x_1+y_1, x_1 \cdot (y_1+1) + x_2 + y_2, 2 \cdot x_3 + y_3 + 1) \end{aligned}$$

Hay que ver que $\circ_A$ y λ_A son monótonas en todos sus argumentos. Para la abstracción esto es inmediato. Mostramos que también vale para la composición, es decir:

$$\vec{x} > \vec{z} \implies (\forall \vec{y})\ \vec{x} \circ \vec{y} > \vec{z} \circ \vec{y} \ \wedge\ \vec{y} \circ \vec{x} > \vec{y} \circ \vec{z}$$

Separamos según la definición de órden lexicográfico:

- $x_1 > z_1$: Como $x_1 + y_1 > z_1 + y_1$, es cierto para todo v, w que

  $$(x_1 + y_1, v, w) > (z_1 + y_1, v, w)$$

  y por ende $\vec{x} \circ \vec{y} > \vec{z} \circ \vec{y}$. De forma análoga se prueba para $\vec{y} \circ \vec{x} > \vec{y} \circ \vec{z}$

- $x_1 = z_1 \wedge x_2 > z_2$: Restamos los vectores resultantes y vemos si el resultado es mayor a $(0,0,0)$:

  - $\vec{x} \circ \vec{y} > \vec{z} \circ \vec{y}$
    1. $x_1 + y_1 - (z_1 + y_1) = 0$
    2. $x_1 \cdot (y_1 + 1) + x_2 + y_2 - (z_1 \cdot (y_1 + 1) + z_2 + y_2) = x_2 - z_2 > 0$
  - $\vec{y} \circ \vec{x} > \vec{y} \circ \vec{z}$
    1. $y_1 + x_1 - (y_1 + z_1) = 0$
    2. $y_1 \cdot (x_1 + 1) + y_2 + x_2 - (y_1 \cdot (z_1 + 1) + y_2 + z_2) = x_2 - z_2 > 0$

Notar que no hace falta tener en cuenta la tercer componente del resultado.

- $x_1 = z_1 \wedge x_2 = z_2 \wedge x_3 > z_3$: Es fácil notar que las dos primeras componentes del resultado van a ser iguales, puesto que sólo utilizan las dos primeras componentes de los operandos. Luego, analizamos la tercer componente:
  - $\vec{x} \circ \vec{y} > \vec{z} \circ \vec{y}$

    $$2 \cdot x_3 + y_3 + 1 - (2 \cdot z_3 + y_3 + 1) = 2 \cdot x_3 - 2 \cdot z_3 > 0$$

  - $\vec{y} \circ \vec{x} > \vec{y} \circ \vec{z}$

    $$2 \cdot y_3 + x_3 + 1 - (2 \cdot y_3 + z_3 + 1) = x_3 - z_3 > 0$$

Ahora sólo resta ver que esta interpretación es compatible con el sistema $\mathit{Erase}.(\mathit{AMAs}_0)$. Para ello hay que mostrar que para todas las reglas el lado izquierdo es mayor al lado derecho en esta interpretación. Vamos a calcular regla por regla el valor de cada lado, y compararemos luego el resultado.

Abs1

$$\begin{aligned} l.\quad & (\lambda(x_1, x_2, x_3)) \circ (y_1, y_2, y_3) \\ & = (x_1 + 1, x_2, x_3) \circ (y_1, y_2, y_3) \\ & = (x_1 + 1 + y_1, (x_1 + 1) \cdot (y_1 + 1) + x_2 + y_2, 2 \cdot x_3 + y_3 + 1) \end{aligned}$$

$$\begin{aligned} r.\quad & \lambda((x_1, x_2, x_3) \circ 1) \\ & = \lambda(x_1, x_1 + x_2, 2 \cdot x_3 + 1) \\ & = (x_1 + 1, x_1 + x_2, 2 \cdot x_3 + 1) \end{aligned}$$

Resulta obvio que $l > r$.

Abs2

$$\begin{aligned} l.\quad & (\lambda(x_1, x_2, x_3)) \circ (y_1, y_2, y_3) \\ & = (x_1 + 1 + y_1, (x_1 + 1) \cdot (y_1 + 1) + x_2 + y_2, 2 \cdot x_3 + y_3 + 1) \\ & = (x_1 + 1 + y_1, x_1 \cdot y_1 + x_1 + y_1 + x_2 + y_2 + 1, 2 \cdot x_3 + y_3 + 1) \end{aligned}$$

$$\begin{aligned} r.\quad & \lambda((x_1, x_2, x_3) \circ ((y_1, y_2, y_3) \circ \uparrow)) \\ & = \lambda((x_1, x_2, x_3) \circ (y_1, y_1 + y_2, 2 \cdot y_3 + 1)) \\ & = \lambda(x_1 + y_1, x_1 \cdot (y_1 + 1) + x_2 + y_1 + y_2, 2 \cdot x_3 + 2 \cdot y_3 + 2) \\ & = (x_1 + y_1 + 1, x_1 \cdot (y_1 + 1) + x_2 + y_1 + y_2, 2 \cdot x_3 + 2 \cdot y_3 + 2) \end{aligned}$$

Resulta obvio que $l > r$, puesto que la segunda componente es mayor en l que en r.

Ass

$$\begin{aligned} l.\quad & ((x_1, x_2, x_3) \circ (y_1, y_2, y_3)) \circ (z_1, z_2, z_3) \\ & = (x_1 + y_1, x_1 \cdot (y_1 + 1) + x_2 + y_2, 2x_3 + y_3 + 1) \circ (z_1, z_2, z_3) \\ & = (x_1 + y_1 + z_1, (x_1 + y_1) \cdot (z_1 + 1) + x_1 \cdot (y_1 + 1) + x_2 + y_2 + z_2, 2 \cdot (2x_3 + y_3 + 1) + z_3 + 1) \\ & = (x_1 + y_1 + z_1, x_1 \cdot (2 + y_1 + z_1) + y_1 \cdot (1 + z_1) + x_2 + y_2 + z_2, 4 \cdot x_3 + 2 \cdot y_3 + z_3 + 3) \end{aligned}$$

$$\begin{aligned} r.\quad & (x_1, x_2, x_3) \circ ((y_1, y_2, y_3) \circ (z_1, z_2, z_3)) \\ & = (x_1, x_2, x_3) \circ (y_1 + z_1, y_1 \cdot (z_1 + 1) + y_2 + z_2, 2 \cdot y_3 + z_3 + 1) \\ & = (x_1 + y_1 + z_1, x_1 \cdot (y_1 + z_1 + 1) + x_2 + y_1 \cdot (z_1 + 1) + y_2 + z_2, 2 \cdot x_3 + 2 \cdot y_3 + z_3 + 2) \\ & = (x_1 + y_1 + z_1, x_1 \cdot (1 + y_1 + z_1) + y_1 \cdot (1 + z_1) + x_2 + y_2 + z_2, 2 \cdot x_3 + 2 \cdot y_3 + z_3 + 2) \end{aligned}$$

Resulta obvio que $l > r$, puesto que la segunda componente es mayor o igual, y la tercera componente es mayor en l que en r.

Queda demostrado que $\mathit{Erase}.(AMAs_0)$ tiene un álgebra monótona bien fundada y compatible, y como el orden lexicográfico es total, resulta totalmente terminante. □

Lema 56. *AMAs* es SN.

Demostración. Por lema 55, $\mathit{Erase}.(AMAs_0)$ es TT. Por teorema 20, $AMAs_0$ es TT. Luego, por teorema 16, *AMAs* es TT. Finalmente, por teorema 14, *AMAs* es SN. □

Ya podemos demostrar el punto principal de este capítulo:

Teorema 57. σ_{gc} es fuertemente normalizante.

Demostración. Por lema 56, *AMAs* es SN, luego por la proposición 54 obtenemos SN de σ_{gc0}. Finalmente, la proposición 53 nos permite concluir SN de σ_{gc}. □

Capítulo 4

Lemas útiles

En este capítulo presentaremos distintos lemas utilizados en los próximos capítulos.

Por ejemplo, en el capítulo 5 se presenta la necesidad de encontrar un reducto común entre $a[s]$ y $a[s \circ t]$, para a un término, s y t sustituciones, tales que $s \Rightarrow a$ y $t \not\Rightarrow a[s]$. Este lema, el 78, se llama *ClosGcR*.

Si se intenta demostrar este lema por inducción, se encuentra rapidamente con que no es posible. Si la inducción es en el término a, entonces en el caso inductivo $a = \lambda b$ se tiene

$$(\lambda b)[s] \ \to_{\text{Abs}} \ \lambda b[1 \cdot (s \circ \uparrow)]$$

y

$$(\lambda b)[s \circ t] \ \to_{\text{Abs}} \ \lambda b[1 \cdot ((s \circ t) \circ \uparrow)]$$

con lo que no es posible aplicar la hipótesis inductiva. Algo similar ocurre si se hace inducción en s.

Intuitivamente, si $t \not\Rightarrow a[s]$ es porque no va a modificar a los términos en s, ni va a modificar las variables libres en $a[s]$, con lo que el resultado de sustituir $s \circ t$ en a es el mismo que el de sustituir s en a. Dicho de otra forma, cada variable libre de a será reemplazada por el mismo término ya sea utilizando s como $s \circ t$. Hemos formalizado este concepto en el lema 75, en el cual se establece para un término a, y sustituciones s y t que si para toda variables libre i de a se obtiene que existe un reducto común entre $i[s]$ y $i[t]$, entonces existe un reducto común entre $a[s]$ y $a[t]$.

Luego la demostración del lema *ClosGcR* se reduce a mostrar que para toda variable libre i de a existe un reducto común entre $i[s]$ y $i[s \circ t]$

También se demuestra de igual forma el lema 80, que establece la existencia de un reducto común entre $a[t]$ y $a[s \circ t]$ si $s \not\Rightarrow a$.

Al menos que se especifique lo contrario, notaremos $\to_{\sigma_{\text{gc}}}$ y $\twoheadrightarrow_{\sigma_{\text{gc}}}$ como $\to$ y $\twoheadrightarrow$, respectivamente.

Para algunas demostraciones, vamos a utilizar la siguiente

Definición 58. Dada una estrategia de reducción cualquiera (por ejemplo, *left-most*), llamamos $\sigma_{\text{gc}}{}'(t)$ a la σ_{gc}-f.n. del término t utilizando esa estrategia particular. Por el teorema 57 sabemos que tal forma normal existe.

Lema 59. Sean b un término, s y t sustituciones tales que $|s| = (m, n)$ y $|t| = (p, q)$

$$\mathrm{FV}(b) \neq \emptyset \ \wedge \ s \circ t \not\triangleright b \implies m - n = q - p \ \wedge \ (\forall i \in \mathrm{FV}(b))\ i > m$$

Demostración. Dividimos según la definición de $s \circ t$.

- Si $p \geq n$, $|s \circ t| = (m + p - n, q)$. Por definición de $\not\triangleright$, debe suceder que

  $$m + p - n = q \ \wedge \ (\forall i \in \mathrm{FV}(b))\ i > m + p - n$$

  Luego,

  $$m + p - n = q \quad \iff \quad m - n = q - p$$

  y como $p \geq n$,

  $$(\forall i \in \mathrm{FV}(b))\ i > m + p - n \implies (\forall i \in \mathrm{FV}(b))\ i > m$$

- Si $p < n$, $|s \circ t| = (m, q + n - p)$. Por definición de $\not\triangleright$, debe suceder que

  $$m = q + n - p \ \wedge \ (\forall i \in \mathrm{FV}(b))\ i > m$$

  Luego,

  $$m = q + n - p \quad \iff \quad m - n = q - p$$

□

Lema 60. Sean b un término, s y t sustituciones:

1. Si $s \triangleright b$ y $t \not\triangleright b[s]$, entonces $s \circ t \triangleright b$.
2. Si $s \not\triangleright b$ y $t \triangleright b$, entonces $s \circ t \triangleright b$.
3. Si $s \not\triangleright b$ y $t \not\triangleright b$, entonces $s \circ t \not\triangleright b$.
4. No es cierto que si $s \triangleright b$ y $t \triangleright b[s]$, entonces $s \circ t \triangleright b$.
5. Si $s \triangleright b$ y $s \circ t \not\triangleright b$, entonces $t \triangleright b[s]$.
6. Si $t \triangleright b$ y $s \circ t \not\triangleright b$, entonces $s \triangleright b$.
7. Si $s \not\triangleright b$ y $s \circ t \not\triangleright b$, entonces $t \not\triangleright b$.
8. Si $t \not\triangleright b[s]$ y $s \circ t \not\triangleright b$, entonces $s \not\triangleright b$.
9. Si $s \not\triangleright b$ y $s \circ t \triangleright b$, entonces $t \triangleright b$.

Demostración. Sea $|s| = (m, n)$ y $|t| = (p, q)$.

1. $s \triangleright b \ \wedge \ t \not\triangleright b[s] \implies s \circ t \triangleright b$:
   Suponemos que $s \circ t \not\triangleright b$. Como $s \triangleright b$, entonces

   $$\mathrm{FV}(b) \neq \emptyset \tag{4.1}$$

   Por lema 59,

   $$m - n = q - p \tag{4.2}$$

y

$$(\forall i \in \mathrm{FV}(b))\ i > m \tag{4.3}$$

Si suponemos $m = n$, por ecuación 4.3 sucedería que $s \not\Vdash b$, lo que es una contradicción. Luego, $m \neq n$, y por ecuación 4.2,

$$p \neq q \tag{4.4}$$

Por hipótesis, $t \not\Vdash b[s]$, y por ecuación 4.4, $\mathrm{FV}(b[s]) = \emptyset$. Luego,

$$\begin{aligned} \mathrm{FV}(b[s]) = \emptyset \quad &\implies \quad (\mathrm{FV}(b)_{>m} + n - m) \cup \mathrm{FV}(s) = \emptyset \\ &\implies \quad \mathrm{FV}(b)_{>m} + n - m = \emptyset \\ &\iff \quad \mathrm{FV}(b)_{>m} = \emptyset \end{aligned}$$

Pero por las ecuaciones 4.1 y 4.3, $(\exists i \in \mathrm{FV}(b))\ i > m$, que contradice lo anterior. La contradicción surge de suponer que $s \circ t \not\Vdash b$.

2. $s \not\Vdash b \ \wedge \ t \Vdash b \implies s \circ t \Vdash b$:

   Como $t \Vdash b$, entonces

   $$\mathrm{FV}(b) \neq \emptyset \tag{4.5}$$

   y

   $$p \neq q \ \vee \ (\exists i \in \mathrm{FV}(b))\ i \leq p \tag{4.6}$$

   Como $s \not\Vdash b$, por ecuación 4.5 debe suceder que $m = n$. Suponemos $s \circ t \not\Vdash b$. Por lema 59,

   $$m - n = q - p \iff m - m = q - p \iff p = q$$

   Luego, por ecuación 4.6,

   $$(\exists i \in \mathrm{FV}(b))\ i \leq p \tag{4.7}$$

   Separamos según la definición de peso:

   - Si $p \geq m$, $|s \circ t| = (m + p - m, q) = (p, q)$. Por suposición, $(\forall i \in \mathrm{FV}(b))\ i > p$. Esto contradice la ecuación 4.7.
   - Si $p < m$, $|s \circ t| = (m, m + q - p)$. Por suposición, $(\forall i \in \mathrm{FV}(b))\ i > m > p$. Esto contradice la ecuación 4.7.

   Absurdo, luego $s \circ t \Vdash b$.

3. $s \not\Vdash b \ \wedge \ t \not\Vdash b \implies s \circ t \not\Vdash b$:

   Si $\mathrm{FV}(b) = \emptyset$, entonces $s \circ t \not\Vdash b$. Sino, debe suceder que $m = n$ y que $p = q$ y que toda variable libre de b es mayor que p y que m. Luego, por definición de peso:

   - Si $p \geq m$, $|s \circ t| = (m + p - m, p) = (p, p)$. Como todas las variables libres de b son mayores que p, $s \circ t \not\Vdash b$.
   - Si $p < m$, $|s \circ t| = (m, m + p - p) = (m, m)$. Como todas las variables libres de b son mayores que m, $s \circ t \not\Vdash b$.

4. $s \Vdash b \wedge t \Vdash b[s] \not\Rightarrow s \circ t \Vdash b$: En la sección 2.3 se muestra un contraejemplo.

5. $s \rhd b \ \wedge \ s \circ t \not\rhd b \implies t \rhd b[s]$:

   Si suponemos que $t \not\rhd b[s]$, como $s \rhd b$, por 1 debe suceder que $s \circ t \rhd b$, lo que contradice la hipótesis.

6. $t \rhd b \ \wedge \ s \circ t \not\rhd b \implies s \rhd b$:

   Si suponemos que $s \not\rhd b$, como $t \rhd b$, entonces por 2 debe suceder que $s \circ t \rhd b$, lo que contradice la hipótesis.

7. $s \not\rhd b \ \wedge \ s \circ t \not\rhd b \implies t \not\rhd b$:

   Si suponemos que $t \rhd b$, como $s \not\rhd b$, entonces por 2 debe suceder que $s \circ t \rhd b$, lo que contradice la hipótesis.

8. $t \not\rhd b[s] \ \wedge \ s \circ t \not\rhd b \implies s \not\rhd b$:

   Si suponemos que $s \rhd b$, como $t \not\rhd b[s]$, entonces por 1 debe suceder que $s \circ t \rhd b$, lo que contradice la hipótesis.

9. $s \not\rhd b \ \wedge \ s \circ t \rhd b \implies t \rhd b$:

   Si suponemos que $t \not\rhd b$, como $s \not\rhd b$, entonces por 3 debe suceder que $s \circ t \not\rhd b$, lo que contradice la hipótesis.

□

Lema 61. Sean s y t dos sustituciones de la forma

$$\begin{array}{rcl} s & = & a_1 \cdot \ldots \cdot a_m \cdot \uparrow^n \\ t & = & b_1 \cdot \ldots \cdot b_p \cdot \uparrow^q \end{array}$$

Entonces

$$s \circ t \twoheadrightarrow \begin{cases} a'_1 \cdot \ldots \cdot a'_m \cdot b_{m+1} \cdot \ldots \cdot b_p \cdot \uparrow^q & \text{si } n < p \\ a'_1 \cdot \ldots \cdot a'_m \cdot \uparrow^{q+n-p} & \text{si } n \geq p \end{cases}$$

Con

$$a'_i = \begin{cases} a_i[t] & \text{si } t \rhd a_i \\ a_i & \text{si } t \not\rhd a_i \end{cases}$$

Demostración.

$$\begin{array}{rcl} s \circ t & = & (a_1 \cdot \ldots \cdot a_m \cdot \uparrow^n) \circ t \\ & \to_{(\text{Map o MapGC})^m} & a'_1 \cdot \ldots \cdot a'_m \cdot (\uparrow^n \circ t) \\ & \to_{\text{Ass}^n} & a'_1 \cdot \ldots \cdot a'_m \cdot (\uparrow \circ (\ldots \circ (\uparrow \circ t) \ldots)) \end{array}$$

$$\begin{array}{lll} (\text{si } n < p) & \to_{\text{ShiftCons}^n} & a'_1 \cdot \ldots \cdot a'_m \cdot b_{n+1} \cdot \ldots \cdot b_p \cdot \uparrow^q \\ (\text{si } n \geq p) & \to_{\text{ShiftCons}^p} & a'_1 \cdot \ldots \cdot a'_m \cdot \uparrow^{q+n-p} \end{array}$$

□

Lema 62 (Caracterización de las formas normales)**.** Las σ_{gc}-f.n. son las mismas que las σ-f.n., es decir, están dadas por

$$(\Lambda\sigma^t_{\text{gc}})_{\text{nf}} \ \{ \ a ::= 1 \mid 1[\uparrow^n] \mid a \ a \mid \lambda a$$

$$(\Lambda\sigma^s_{\text{gc}})_{\text{nf}} \ \{ \ s ::= \mathbf{id} \mid \ \uparrow^n \ \mid a \cdot s$$

Demostración. Es claro que los términos generados por la gramática son formas normales. Queda por ver que no hay otras formas normales. Separamos los casos por sort.

- Sea a un término de $\lambda\sigma_{\text{gc}}$ en σ_{gc}-f.n. Vamos a mostrar por inducción que el mismo está generado por la gramática.
  - $a = 1$, es generado por la gramática.
  - $a = b\ c$, con b y c en forma normal. Por h. i., b y c están generados por la gramática. Entonces $b\ c$ está generado por la gramática.
  - $a = \lambda b$, con b y s en forma normal. Por h. i., b está generado por la gramática. Luego λb está generado por la gramática.
  - $a = b[s]$, con b en forma normal. Analizamos según la forma de b:
    - $b = 1$, analizamos s:
      - $s = \mathbf{id}$, se podría aplicar *GC*, con lo que no estaría en forma normal.
      - $s = \uparrow$, la gramática genera $1[\uparrow]$.
      - $s = (c \cdot t)$, se podría aplicar *VarCons*, con lo que no estaría en forma normal.
      - $s = t \circ u$, si t fuera $\mathbf{id}$ se podría aplicar *IdL*; si t fuera $a \cdot s$ se podría aplicar *Map* o *MapGC*; si t fuera $t_1 \circ t_2$ se podría aplicar *Ass*. Con lo que t debe ser $\uparrow$. Luego, $s = \uparrow \circ u$. Probamos que si $\uparrow \circ u$ está en σ_{gc}-f.n., entonces $\exists n \geq 1, u = \uparrow^n$:
        $u = \mathbf{id}$, se podría aplicar *ShiftId*.
        $u = \uparrow$
        $u = a \cdot s$, se podría aplicar *ShiftCons*.
        $u = v \circ w$, como antes, debe suceder que $v = \uparrow$ y por h.i. $w = \uparrow^m$, con lo que $u = \uparrow^{m+1}$
        Luego, si $a = b[s]$, debe ser $b = 1 \wedge s = \uparrow^n$.
    - $b = b_1\ b_2$, se podría aplicar *GC* o alguna *App*, con lo que no estaría en forma normal. Notar que por lema 45 siempre se puede aplicar alguna de las reglas mencionadas.
    - $b = \lambda c$, se podría aplicar *Abs* o *GC*, con lo que no estaría en forma normal. Notar que siempre se puede aplicar alguna de las reglas mencionadas.
    - $b = c[t]$, se podría aplicar *GC* o *Clos* o *ClosGC*, con lo que no estaría en forma normal. Notar que siempre se puede aplicar alguna de las reglas mencionadas.

    Luego sólo puede suceder que $a = 1[\uparrow^n]$.

- Sea s una sustitución de $\lambda\sigma_{\text{gc}}$. Vamos a mostrar por inducción que la misma está generada por la gramática.
  - $s = \mathbf{id}$, es generada por la gramática.
  - $s = \uparrow$, es generada por la gramática.
  - $s = (a \cdot t)$, con a y t en forma normal. Por h. i., a y t son generadas por la gramática, luego $a \cdot t$ es generada por la gramática.

– $s = s_1 \circ s_2$, se sigue el mismo razonamiento que en el item anterior para concluir que $s = \uparrow^n$.

□

De ahora en más, $\forall i \in \mathbb{N}_{>1}, i = 1[\uparrow^{i-1}]$.

Lema 63. Para dos sustituciones s y t,

$$s \to_{\lambda\sigma_{\mathrm{gc}}} t \implies |s| = |t|$$

Demostración. Por inducción en s:

- $s = \mathbf{id}$, $\uparrow$, no hay reducción posible, se satisface vacuamente.
- $s = a \cdot u$, sólo es posible reducir en alguno de los términos.
  - Si $t = a' \cdot u$, el peso se mantiene.
  - Si $t = a \cdot u'$, por h. i. $|u| = |u'|$. Luego debe ocurrir que $|a \cdot u| = |a \cdot u'|$.
- $s = u \circ v$, pueden ocurrir las siguientes situaciones:
  - Si $t = u' \circ v$, por h. i. $|u| = |u'|$, y se concluye que $|s| = |t|$.
  - Si $t = u \circ v'$, por h. i. $|v| = |v'|$, y se concluye que $|s| = |t|$.
  - Si $u = \mathbf{id}$ y se aplica *IdL*, entonces $t = v$, y $|s| = |v| = |t|$.
  - Si $u = \uparrow$ y $v = \mathbf{id}$, y se aplica *ShiftId*, entonces $t = \uparrow$, y $|s| = |\uparrow| = |t|$.
  - Si $u = \uparrow$ y $v = a \cdot w$, y se aplica *ShiftCons*, entonces $t = w$. Si $|w| = (p, q)$, entonces $|v| = (p{+}1, q)$. Entonces $|s| = (0 + (p + 1) - 1, q) = (p, q)$, con lo que $|s| = |t|$.
  - Si $u = a \cdot w$, y se aplica *Map* o *MapGC*, entonces $t = b \cdot (w \circ v)$ (con $b = a$ o $b = a[v]$). Sea $|w| = (m, n)$ y $|v| = (p, q)$.

    $$|s| = |(a \cdot w) \circ v| = \begin{cases} (m + 1 + p - n, q) & \text{si } p \geq n \\ (m + 1, q + n - p) & \text{si } p < n \end{cases}$$

    $$|t| = |b \cdot (w \circ v)| = \begin{cases} (m + p - n + 1, q) & \text{si } p \geq n \\ (m + 1, q + n - p) & \text{si } p < n \end{cases}$$

    En cualquier caso resulta $|s| = |t|$.
  - Si $u = (w \circ z)$, y se aplica *Ass*, entonces $t = w \circ (z \circ v)$.
    Sean $|w| = (m, n) \quad |z| = (p, q) \quad |v| = (i, j)$

    $$|w \circ z| = \begin{cases} (m + p - n, q) & \text{si } p \geq n \\ (m, q + n - p) & \text{si } p < n \end{cases}$$

    $$|z \circ v| = \begin{cases} (p + i - q, j) & \text{si } i \geq q \\ (p, j + q - i) & \text{si } i < q \end{cases}$$

$$|(w \circ z) \circ v| = \begin{cases} p \geq n \begin{cases} (m+p-n+i-q, j) & i \geq q \\ (m+p-n, j+q-i) & i < q \end{cases} \\ p < n \begin{cases} (m+i-(q+n-p), j) & i \geq q+n-p \\ (m, j+q+n-p-i) & i < q+n-p \end{cases} \end{cases} \tag{4.8}$$

$$|w \circ (z \circ v)| = \begin{cases} i \geq q \begin{cases} (m+p+i-q-n, j) & p+i-q \geq n \\ (m, j+n-(p+i-q)) & p+i-q < n \end{cases} \\ i < q \begin{cases} (m+p-n, j+q-i) & p \geq n \\ (m, j+q-i+n-p) & p < n \end{cases} \end{cases} \tag{4.9}$$

Analizamos todos los posibles casos:

* $p \geq n \ \wedge \ i \geq q$: Notar que $p+i-q \geq n$

  (4.8) $(m+p-n+i-q, j) = (m+p+i-q-n, j)$ (4.9)

* $p \geq n \ \wedge \ i < q$:

  (4.8) $(m+p-n, j+q-i) = (m+p-n, j+q-i)$ (4.9)

* $p < n \ \wedge \ i \geq q+n-p$: Notar que $i \geq q$ y $p+i-q \geq n$

  (4.8) $(m+i-(q+n-p), j) = (m+p+i-q-n, j)$ (4.9)

* $p < n \ \wedge \ i < q+n-p$: Notar que $p+i-q < n$.
  1. $i \geq q$

     (4.8) $(m, j+q+n-p-i) = (m, j+n-(p+i-q))$ (4.9)

  2. $i < q$

     (4.8) $(m, j+q+n-p-i) = (m, j+q-i+n-p)$ (4.9)

□

Corolario 64 (Preservación del peso por reducciones)**.** Para dos sustituciones s y t,

$$s \twoheadrightarrow_{\lambda\sigma_{\mathrm{gc}}} t \implies |s| = |t|$$

Demostración. Por inducción en la cantidad de pasos, utilizando el lema anterior. □

Lema 65. Sean a y b términos, s y t sustituciones,

1. $a \to_{\lambda\sigma_{\mathrm{gc}}} b \implies \mathrm{FV}(b) \subseteq \mathrm{FV}(a)$
2. $s \to_{\lambda\sigma_{\mathrm{gc}}} t \implies \mathrm{FV}(t) \subseteq \mathrm{FV}(s)$

Demostración. Por inducción simultánea en a y s. Utilizaremos $\to$ para referirnos a $\to_{\lambda\sigma_{\mathrm{gc}}}$

1.
   - $a = 1$: No hay reducción posible, se satisface vacuamente.
   - $a = a_1\ a_2$
     - $b = b_1\ a_2 \quad a_1 \to b_1$
       Por h. i., $\mathrm{FV}(b_1) \subseteq \mathrm{FV}(a_1)$, luego

$$\begin{aligned} \mathrm{FV}(b) &= \mathrm{FV}(b_1\ a_2) \\ &= \mathrm{FV}(b_1) \cup \mathrm{FV}(a_2) \\ &\subseteq \mathrm{FV}(a_1) \cup \mathrm{FV}(a_2) \\ &= \mathrm{FV}(a_1\ a_2) \\ &= \mathrm{FV}(a) \end{aligned}$$

     - $b = a_1\ b_2 \quad a_2 \to b_2$
       Se sigue el mismo razonamiento anterior.
     - $b = c[a_2 \cdot \mathbf{id}] \quad a_1 = \lambda c$

$$\begin{aligned} \mathrm{FV}(a) &= \mathrm{FV}((\lambda c)\ a_2) \\ &= \mathrm{FV}(\lambda c) \cup \mathrm{FV}(a_2) \\ &= (\mathrm{FV}(c) - 1) \cup \mathrm{FV}(a_2) \end{aligned}$$

$$\begin{aligned} \mathrm{FV}(b) &= \mathrm{FV}(c[a_2 \cdot \mathbf{id}]) \\ &= (\mathrm{FV}(c) + 0 - 1) \cup \mathrm{FV}(a_2 \cdot \mathbf{id}) \\ &= (\mathrm{FV}(c) - 1) \cup \mathrm{FV}(a_2) \end{aligned}$$

   - $a = \lambda c \quad b = \lambda d \quad c \to d$
     Por h.i., $\mathrm{FV}(d) \subseteq \mathrm{FV}(c)$, luego

$$\begin{aligned} \mathrm{FV}(b) &= \mathrm{FV}(\lambda d) \\ &= \mathrm{FV}(d) - 1 \\ &\subseteq \mathrm{FV}(c) - 1 \\ &= \mathrm{FV}(\lambda c) \\ &= \mathrm{FV}(a) \end{aligned}$$

   - $a = c[s]$
     Sea $|s| = (m, n)$
     (a) $b = d[s] \quad c \to d$
     Por h.i., $\mathrm{FV}(d) \subseteq \mathrm{FV}(c)$, luego

$$\begin{aligned} \mathrm{FV}(b) &= \mathrm{FV}(d[s]) \\ &= (\mathrm{FV}(d)_{>m} + n - m) \cup \mathrm{FV}(s) \\ &\subseteq (\mathrm{FV}(c)_{>m} + n - m) \cup \mathrm{FV}(s) \\ &= \mathrm{FV}(c[s]) \\ &= \mathrm{FV}(a) \end{aligned}$$

(b) $b = c[t] \quad s \to t$
Por h.i., $\mathrm{FV}(t) \subseteq \mathrm{FV}(s)$, y por lema 63, $|t| = (m, n)$. Luego,

$$\begin{aligned} \mathrm{FV}(b) &= \mathrm{FV}(c[t]) \\ &= (\mathrm{FV}(c)_{>m} + n - m) \cup \mathrm{FV}(t) \\ &\subseteq (\mathrm{FV}(c)_{>m} + n - m) \cup \mathrm{FV}(s) \\ &= \mathrm{FV}(c[s]) \\ &= \mathrm{FV}(a) \end{aligned}$$

(c) $b = c \quad s \not\rhd c$
Por definición de $\not\rhd$, puede suceder
- $\mathrm{FV}(c) = \emptyset$:
$$\mathrm{FV}(b) = \emptyset \subseteq \mathrm{FV}(a)$$
- $\mathrm{FV}(c) \neq \emptyset$: $m = n \ \wedge \ (\forall i \in \mathrm{FV}(c)) \ i > m$

$$\begin{aligned} \mathrm{FV}(a) &= \mathrm{FV}(c[s]) \\ &= (\mathrm{FV}(c)_{>m} + m - m) \cup \mathrm{FV}(s) \\ &= \mathrm{FV}(c) \cup \mathrm{FV}(s) \\ &\supseteq \mathrm{FV}(c) \\ &= \mathrm{FV}(b) \end{aligned}$$

(d) $a = 1[b \cdot s]$

$$\begin{aligned} \mathrm{FV}(a) &= \mathrm{FV}(1[b \cdot s]) \\ &= (\mathrm{FV}(1)_{>m+1} + n - (m+1)) \cup \mathrm{FV}(b \cdot s) \\ &= \mathrm{FV}(b \cdot s) \\ &= \mathrm{FV}(b) \cup \mathrm{FV}(s) \\ &\supseteq \mathrm{FV}(b) \end{aligned}$$

(e) $a = (c \ d)[s] \quad b = c[s] \ d[s] \quad s \not\rhd c \wedge s \not\rhd d$

$$\begin{aligned} \mathrm{FV}((c \ d)[s]) &= (\mathrm{FV}(c \ d)_{>m} + n - m) \cup \mathrm{FV}(s) \\ &= ((\mathrm{FV}(c) \cup \mathrm{FV}(d))_{>m} + n - m) \cup \mathrm{FV}(s) \\ &= (\mathrm{FV}(c)_{>m} + n - m) \cup (\mathrm{FV}(d)_{>m} + n - m) \cup \mathrm{FV}(s) \\ &= ((\mathrm{FV}(c)_{>m} + n - m) \cup \mathrm{FV}(s)) \cup \\ &\qquad \cup((\mathrm{FV}(d)_{>m} + n - m) \cup \mathrm{FV}(s)) \\ &= \mathrm{FV}(c[s]) \cup \mathrm{FV}(d[s]) \\ &= \mathrm{FV}(c[s] \ d[s]) \end{aligned}$$

(f) $a = (c \ d)[s] \quad b = c[s] \ d \quad s \not\rhd c \wedge s \not\rhd d$

$$\begin{aligned} \mathrm{FV}((c \ d)[s]) &= (\mathrm{FV}(c \ d)_{>m} + n - m) \cup \mathrm{FV}(s) \\ &= ((\mathrm{FV}(c)_{>m} + n - m) \cup \mathrm{FV}(s)) \cup \\ &\qquad \cup((\mathrm{FV}(d)_{>m} + n - m) \cup \mathrm{FV}(s)) \\ &= \mathrm{FV}(c[s]) \cup \mathrm{FV}(d[s]) \\ &\supseteq \mathrm{FV}(c[s]) \cup \mathrm{FV}(d) \quad \text{Por razonamiento análogo al item } c \\ &= \mathrm{FV}(c[s] \ d) \end{aligned}$$

(g) $a = (c\ d)[s] \quad b = c\ d[s] \quad s \not\Rightarrow c \wedge s \Rrightarrow d$
Por razonamiento análogo al anterior.

(h) $a = (\lambda c)[s] \quad b = \lambda c[1 \cdot (s \circ \uparrow)] \quad s \Rrightarrow \lambda c$

$$\begin{aligned}
\mathrm{FV}(a) &= \mathrm{FV}((\lambda c)[s]) \\
&= (\mathrm{FV}(\lambda c)_{>m} + n - m) \cup \mathrm{FV}(s) \\
&= ((\mathrm{FV}(c) - 1)_{>m} + n - m) \cup \mathrm{FV}(s)
\end{aligned}$$

$$\begin{aligned}
\mathrm{FV}(b) &= \mathrm{FV}(\lambda c[1 \cdot (s \circ \uparrow)]) \\
&= \mathrm{FV}(c[1 \cdot (s \circ \uparrow)]) - 1 \\
&= ((\mathrm{FV}(c)_{>m+1} + n + 1 - (m+1)) \cup \mathrm{FV}(1 \cdot (s \circ \uparrow))) - 1 \\
&= ((\mathrm{FV}(c)_{>m+1} + n - m) \cup \mathrm{FV}(1) \cup \mathrm{FV}(s \circ \uparrow)) - 1 \\
&= ((\mathrm{FV}(c)_{>m+1} + n - m) \cup \{1\} \cup (\mathrm{FV}(s) + 1)) - 1 \\
&= (\mathrm{FV}(c)_{>m+1} + n - m - 1) \cup \mathrm{FV}(s) \\
&=_{\text{Obs 39}} ((\mathrm{FV}(c) - 1)_{>m} + n - m) \cup \mathrm{FV}(s)
\end{aligned}$$

(i) $a = c[s][t] \quad b = c[s \circ t] \quad s \Rrightarrow c \ \wedge \ t \Rrightarrow c[s] \ \wedge \ s \circ t \Rrightarrow c$
Por observación 42.

(j) $a = b[s][t] \quad s \Rrightarrow b \ \wedge \ t \Rrightarrow b[s] \ \wedge \ s \circ t \not\Rightarrow b$
Por observación 42 se tiene que $\mathrm{FV}(b[s][t]) = \mathrm{FV}(b[s \circ t])$. Como $s \circ t \not\Rightarrow b$, se sigue el mismo razonamiento que en el item c, y se concluye que $\mathrm{FV}(b) \subseteq \mathrm{FV}(b[s \circ t]) = \mathrm{FV}(b[s][t])$

2. • $a = \mathbf{id}, \uparrow$: No hay reducción posible, se satisface vacuamente.

• $s = a \cdot u$:

(a) $t = b \cdot u \quad a \to b$:

$$\begin{aligned}
\mathrm{FV}(t) &= \mathrm{FV}(b \cdot u) \\
&= \mathrm{FV}(b) \cup \mathrm{FV}(u) \\
&\subseteq_{\text{h.i.}} \mathrm{FV}(a) \cup \mathrm{FV}(u) \\
&= \mathrm{FV}(a \cdot u) \\
&= \mathrm{FV}(s)
\end{aligned}$$

(b) $t = a \cdot v \quad u \to v$:

$$\begin{aligned}
\mathrm{FV}(t) &= \mathrm{FV}(a \cdot v) \\
&= \mathrm{FV}(a) \cup \mathrm{FV}(v) \\
&\subseteq_{\text{h.i.}} \mathrm{FV}(a) \cup \mathrm{FV}(u) \\
&= \mathrm{FV}(a \cdot u) \\
&= \mathrm{FV}(s)
\end{aligned}$$

• $s = u \circ v$:
Sean $|u| = (m, n)$ y $|v| = (p, q)$

(a) $t = w \circ v \quad u \to w$:

$$\begin{array}{rcl} \mathrm{FV}(t) & = & \mathrm{FV}(w \circ v) \\ & = & (\mathrm{FV}(w)_{>p} + q - p) \cup \mathrm{FV}(v) \\ & \subseteq_{\text{h.i.}} & (\mathrm{FV}(u)_{>p} + q - p) \cup \mathrm{FV}(v) \\ & = & \mathrm{FV}(u \circ v) \\ & = & \mathrm{FV}(s) \end{array}$$

(b) $t = u \circ w \quad v \to w$:
Por lema 63, $|w| = (p, q)$

$$\begin{array}{rcl} \mathrm{FV}(t) & = & \mathrm{FV}(u \circ w) \\ & = & (\mathrm{FV}(u)_{>p} + q - p) \cup \mathrm{FV}(w) \\ & \subseteq_{\text{h.i.}} & (\mathrm{FV}(u)_{>p} + q - p) \cup \mathrm{FV}(v) \\ & = & \mathrm{FV}(u \circ v) \\ & = & \mathrm{FV}(s) \end{array}$$

(c) $s = \mathbf{id} \circ t$:

$$\mathrm{FV}(s) \;=\; \mathrm{FV}(\mathbf{id} \circ t) \;=\; \mathrm{FV}(t)$$

(d) $s = \uparrow \circ \mathbf{id} \quad t = \uparrow$:

$$\mathrm{FV}(s) \;=\; \mathrm{FV}(\uparrow \circ \mathbf{id}) \;=\; \emptyset \;=\; \mathrm{FV}(\uparrow) \;=\; \mathrm{FV}(t)$$

(e) $s = \uparrow \circ (a \cdot t)$:

$$\begin{array}{rcl} \mathrm{FV}(s) & = & \mathrm{FV}(\uparrow \circ (a \cdot t)) \\ & = & \mathrm{FV}(a \cdot t) \\ & = & \mathrm{FV}(a) \cup \mathrm{FV}(t) \\ & \supseteq & \mathrm{FV}(t) \end{array}$$

(f) $s = (a \cdot u) \circ v \quad t = a[v] \cdot (u \circ v) \quad v \rhd a$:
Sea $|v| = (p, q)$,

$$\begin{array}{rcl} \mathrm{FV}(s) & = & \mathrm{FV}((a \cdot u) \circ v) \\ & = & (\mathrm{FV}(a \cdot u)_{>p} + q - p) \cup \mathrm{FV}(v) \\ & = & ((\mathrm{FV}(a) \cup \mathrm{FV}(u))_{>p} + q - p) \cup \mathrm{FV}(v) \\ & = & (\mathrm{FV}(a)_{>p} + q - p) \cup (\mathrm{FV}(u)_{>p} + q - p) \cup \mathrm{FV}(v) \\ & = & ((\mathrm{FV}(a)_{>p} + q - p) \cup \mathrm{FV}(v)) \cup ((\mathrm{FV}(u)_{>p} + q - p) \cup \mathrm{FV}(v)) \\ & = & ((\mathrm{FV}(a)_{>p} + q - p) \cup \mathrm{FV}(v)) \cup \mathrm{FV}(u \circ v) \\ & = & \mathrm{FV}(a[v]) \cup \mathrm{FV}(u \circ v) \\ & = & \mathrm{FV}(a[v] \cdot (u \circ v)) \\ & = & \mathrm{FV}(t) \end{array}$$

(g) $s = (a \cdot u) \circ v \quad t = a \cdot (u \circ v) \quad v \not\rhd a$:

Sea $|v| = (p, q)$,

$$\begin{aligned}
\mathrm{FV}(s) &= \mathrm{FV}((a \cdot u) \circ v) \\
&= \mathrm{FV}(a[v]) \cup \mathrm{FV}(u \circ v) \\
&\qquad \text{Por razonamiento análogo al item anterior} \\
&\supseteq \mathrm{FV}(a) \cup \mathrm{FV}(u \circ v) \\
&\qquad \text{Por razonamiento análogo al item } c \\
&= \mathrm{FV}(a \cdot (u \circ v)) \\
&= \mathrm{FV}(t)
\end{aligned}$$

(h) $s = (u \circ v) \circ w \quad t = u \circ (v \circ w)$:
Sean $|v| = (m, n)$ y $|w| = (p, q)$,

$$\begin{aligned}
\mathrm{FV}(s) &= \mathrm{FV}((u \circ v) \circ w) \\
&= (\mathrm{FV}(u \circ v)_{>p} + q - p) \cup \mathrm{FV}(w) \\
&= (((\mathrm{FV}(u)_{>m} + n - m) \cup \mathrm{FV}(v))_{>p} + q - p) \cup \mathrm{FV}(w) \\
&= ((\mathrm{FV}(u)_{>m} + n - m)_{>p} + q - p) \cup \\
&\qquad \cup (\mathrm{FV}(v)_{>p} + q - p) \cup \mathrm{FV}(w) \\
&= ((\mathrm{FV}(u)_{>m} + n - m)_{>p} + q - p) \cup \mathrm{FV}(v \circ w)
\end{aligned}$$

Separamos en casos:

- $p \geq n$: $|v \circ w| = (m + p - n, q)$

$$\begin{aligned}
\mathrm{FV}(s) &= ((\mathrm{FV}(u)_{>m} + n - m)_{>p} + q - p) \cup \mathrm{FV}(v \circ w) \\
&= ((\mathrm{FV}(u)_{>m})_{>p-n+m} + n - m + q - p) \cup \mathrm{FV}(v \circ w) \\
&=_{\text{Obs 39}} ((\mathrm{FV}(u)_{>p-n+m} + n - m + q - p) \cup \mathrm{FV}(v \circ w)
\end{aligned}$$

$$\begin{aligned}
\mathrm{FV}(t) &= \mathrm{FV}(u \circ (v \circ w)) \\
&= (\mathrm{FV}(u)_{>m+p-n} + q - (m + p - n)) \cup \mathrm{FV}(v \circ w) \\
&= (\mathrm{FV}(u)_{>p-n+m} + n - m + q - p) \cup \mathrm{FV}(v \circ w)
\end{aligned}$$

- $p < n$: $|v \circ w| = (m, q + n - p)$

$$\begin{aligned}
\mathrm{FV}(s) &= ((\mathrm{FV}(u)_{>m} + n - m)_{>p} + q - p) \cup \mathrm{FV}(v \circ w) \\
&= (\mathrm{FV}(u)_{>m} + n - m + q - p) \cup \mathrm{FV}(v \circ w)
\end{aligned}$$

$$\begin{aligned}
\mathrm{FV}(t) &= \mathrm{FV}(u \circ (v \circ w)) \\
&= (\mathrm{FV}(u)_{>m} + q + n - p - m) \cup \mathrm{FV}(v \circ w)
\end{aligned}$$

□

Corolario 66 (Preservación de las variables libres por reducción)**.** Sean a y b términos, s y t sustituciones,

1. $a \twoheadrightarrow_{\lambda\sigma_{\mathrm{gc}}} b \implies \mathrm{FV}(b) \subseteq \mathrm{FV}(a)$
2. $s \twoheadrightarrow_{\lambda\sigma_{\mathrm{gc}}} t \implies \mathrm{FV}(t) \subseteq \mathrm{FV}(s)$

Demostración. Por inducción en la cantidad de pasos, aplicando el lema anterior. □

Lema 67. Sean a, b términos y s, t sustituciones tales que

$$a \twoheadrightarrow_{\lambda\sigma_{\mathrm{gc}}} b \quad , \quad s \twoheadrightarrow_{\lambda\sigma_{\mathrm{gc}}} t \quad , \quad s \not\rhd a$$

Entonces $s \not\rhd b$ y $t \not\rhd b$

Demostración. Por el corolario 66, $\mathrm{FV}(b) \subseteq \mathrm{FV}(a)$. Luego, por definición de $\not\rhd$

- o bien $\mathrm{FV}(a) = \emptyset$, con lo que $\mathrm{FV}(b) = \emptyset$, luego $s \not\rhd b$ y $t \not\rhd b$,
- o bien $|s| = (m, m)$ y $(\forall i \in \mathrm{FV}(a))\ i > m$, con lo que $(\forall i \in \mathrm{FV}(b))\ i > m$, luego $s \not\rhd b$. Por lema 63, $|t| = (m, m)$, luego $t \not\rhd b$

□

Lema 68. Sea s una sustitución tal que $|s| = (m, n)$. Entonces existen términos $a_1, \ldots, a_m$ tales que

$$\sigma_{\mathrm{gc}}{}'(s) = a_1 \cdot a_2 \cdot \ldots \cdot a_m \cdot \uparrow^n$$

con el caso especial $\sigma_{\mathrm{gc}}{}'(s) = \mathbf{id}$ cuando $m = 0$ y $n = 0$.

Demostración. Por lema 62, debe ocurrir que

$$\sigma_{\mathrm{gc}}{}'(s) = a_1 \cdot a_2 \cdot \ldots \cdot a_p \cdot \uparrow^q \quad \vee \quad \sigma_{\mathrm{gc}}{}'(s) = \mathbf{id}$$

Claramente, dicha sustitución tiene peso (p, q), pero como por corolario 64 el peso se mantiene, debe ocurrir que $p = m$ y que $q = n$. □

Lema 69 (IdR)**.** Sea s una sustitución, entonces $s \circ \mathbf{id} \twoheadrightarrow s$.

Demostración. Por inducción estructural en la sustitución s:

- $s = \mathbf{id}$, tenemos $\mathbf{id} \circ \mathbf{id} \ \to_{\mathrm{IdL}} \ \mathbf{id}$
- $s = \uparrow$, tenemos $\uparrow \circ \mathbf{id} \ \to_{\mathrm{ShiftId}} \uparrow$
- $s = a \cdot t$, tenemos $(a \cdot t) \circ \mathbf{id} \ \to_{\mathrm{MapGC}} \ a \cdot (t \circ \mathbf{id}) \ \overset{h.i.}{\twoheadrightarrow} \ a \cdot t$
- $s = s_1 \circ s_2$, tenemos $(s_1 \circ s_2) \circ \mathbf{id} \ \to_{\mathrm{Ass}} \ s_1 \circ (s_2 \circ \mathbf{id}) \ \overset{h.i.}{\twoheadrightarrow} \ s_1 \circ s_2$

□

Lema 70. Sea s una sustitución en σ_{gc}-forma normal, tal que $|s| = (m, n)$. Por lema 68, existen términos $a_1, \ldots, a_m$ tales que

$$s = a_1 \cdot \ldots \cdot a_m \cdot \uparrow^n \quad \vee \quad s = \mathbf{id}$$

Sea i una variable.

1. Si $i > m$, entonces $i[s] \twoheadrightarrow i + n - m$.
2. Si $i \leq m$, entonces $i[s] \twoheadrightarrow a_i$.

Además, dicha derivación es única.

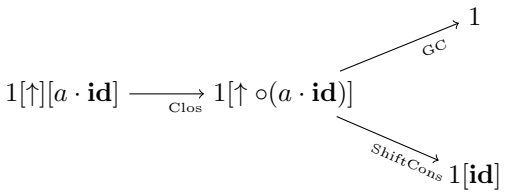
Demostración. Separamos en casos:

- Si $s = \mathbf{id}$, $i[s] \ \to_{\mathrm{GC}} \ i = i + 0 - 0$
- Si $i = 1$,
  1. Si $1 > m$, entonces $m = 0$. $1[\uparrow^n]$ está en forma normal, y corresponde a $1 + n - m$
  2. Si $1 \leq m$, entonces $i[a_1 \cdot ... \cdot a_m \cdot \uparrow^n] \ \to_{\mathrm{VarCons}} \ a_1$
- Si $i = 1[\uparrow^{i-1}]$ y $s \not\Rightarrow i$. Por definición de $\Rightarrow$, debe ocurrir que $m = n$, y que $i > m$. Luego
$$i[s] \ \to_{\mathrm{GC}} \ i = i + n - m$$
- Si $i = 1[\uparrow^{i-1}]$ y $s \Rightarrow i$, y $\uparrow^{i-1} \circ s \Rightarrow 1$.

$$\begin{array}{lrl}
1[\uparrow^{i-1}][s] & \to_{\mathrm{Clos}} & 1[\uparrow^{i-1} \circ s] \\
& = & 1[(\uparrow \circ(\uparrow \circ(...(\uparrow \circ \uparrow)...))) \circ s] \\
& \to_{\mathrm{Ass}} & 1[\uparrow \circ((\uparrow \circ(...(\uparrow \circ \uparrow)...)) \circ s)] \\
& \to_{\mathrm{Ass}^{i-2}} & 1[\uparrow \circ(\uparrow \circ(...(\uparrow \circ(\uparrow \circ s))...))] \\
& = & 1[\uparrow \circ(\uparrow \circ(...(\uparrow \circ(\uparrow \circ(a_1 \cdot ... \cdot a_m \cdot \uparrow^n)))...))] \\
1. & \to_{\mathrm{ShiftCons}^m} & 1[\uparrow^{i-1-m} \circ \uparrow^n] \\
& \to_{\mathrm{Ass}^{i-1-m}} & 1[\uparrow^{i+n-m-1}] = i + n - m \\
2. & \to_{\mathrm{ShiftCons}^{i-1}} & 1[a_i \cdot ... \cdot a_m \uparrow^n] \\
& \to_{\mathrm{VarShift}} & a_i
\end{array}$$

- Si $i = 1[\uparrow^{i-1}]$ y $s \Rightarrow i$, pero $\uparrow^{i-1} \circ s \not\Rightarrow 1$. Por definición de $\Rightarrow$, debe ocurrir que $|\uparrow^{i-1} \circ s| = (0, 0)$. Luego, como $|\uparrow^{i-1}| = (0, i-1)$,
$$|\uparrow^{i-1} \circ s| = (0 + m - (i-1), n) = (0, 0)$$
entonces $n = 0$, y $m = i - 1$, con lo que
$$1[\uparrow^{i-1}][s] \ \to_{\mathrm{ClosGC}} \ 1 = i + n - m$$

Notar que en cada paso no se pudo aplicar otra regla. La regla *ClosGC* es imprescindible para lograr esta unicidad en la reducción, como lo muestra el siguiente ejemplo:

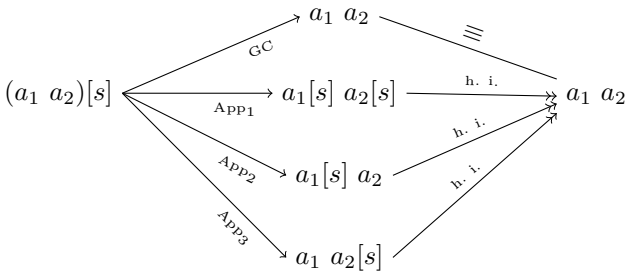


□

Lema 71. Sea a un término en σ_{gc}-f. n., y sea s una sustitución en σ_{gc}-f. n. Si $(\forall i \in \text{FV}(a))\ i[s] \twoheadrightarrow i$ entonces $a[s] \twoheadrightarrow a$.

Demostración. Por inducción en la σ_{gc}-f. n. de a:

- Caso $a = i$: Por hipótesis vale.
- Caso $a = a_1\ a_2$: Como $\text{FV}(a_i) \subseteq \text{FV}(a)$ para $i \in \{1, 2\}$, vale por h.i. que $a_i[s] \twoheadrightarrow a_i$. Luego,



- Caso $a = \lambda b$:

  Si $s \not\Vdash a$,

  $$(\lambda b)[s] \ \to_{\text{GC}}\ \lambda b = a$$

  Sino,

  $$(\lambda b)[s] \ \to_{\text{Abs}}\ \lambda b[1 \cdot (s \circ \uparrow)]$$

  Por estar s en σ_{gc}-f. n., $s = a_1 \cdot ... \cdot a_m \cdot \uparrow^n$ (notar que $s \neq \mathbf{id}$ porque $s \Vdash a$). Luego,

  $$\lambda b[1 \cdot ((a_1 \cdot ... \cdot a_m \cdot \uparrow^n) \circ \uparrow)] \quad \twoheadrightarrow_{\text{Map o MapGC}} \quad \lambda b[1 \cdot a'_1 \cdot ... \cdot a'_m \cdot \uparrow^{n+1}]$$

  Con $a'_i = a_i$ o $a'_i = a_i[\uparrow]$. Por hipótesis, sabemos que si $i \in \text{FV}(a)$, $i[s] \twoheadrightarrow i$. Por definición de FV, para $i > 1$, $i \in \text{FV}(\lambda b) \Leftrightarrow i + 1 \in \text{FV}(b)$. Luego puede ocurrir

  - Si $i + 1 \leq m + 1$, por lema 70

    $$(i + 1)[1 \cdot a'_1 \cdot ... \cdot a'_m \cdot \uparrow^{n+1}] \twoheadrightarrow a'_i$$

    Por lema 70, hay una única derivación

    $$i[s] = i[a_1 \cdot ... \cdot a_m \cdot \uparrow^n] \twoheadrightarrow a_i$$

    Por hipótesis, $i[s] \twoheadrightarrow i$, con lo que $a_i = i$ (pues a_i está en σ_{gc}-f.n.). Luego,

    $$a'_i = a_i[\uparrow] = i[\uparrow] \twoheadrightarrow i + 1$$

– Si $i+1 > m+1$, por lema 70

$$(i+1)[1 \cdot a'_1 \cdot ... \cdot a'_m \cdot \uparrow^{n+1}] \twoheadrightarrow i+1+(n+1)-(m+1)$$

Como $i > m$, por lema 70 existe una única derivación de

$$i[s] \twoheadrightarrow i+n-m$$

Por hipótesis, $i[s] \twoheadrightarrow i$, luego $n = m$. Con lo que queda

$$(i+1)[1 \cdot a'_1 \cdot ... \cdot a'_m \cdot \uparrow^{n+1}] \twoheadrightarrow i+1$$

En ambos casos, queda $(i+1)[1 \cdot (s \circ \uparrow)] \twoheadrightarrow i+1$.
Si $1 \in \mathrm{FV}(b)$, $1[1 \cdot (s \circ \uparrow)] \ \to_{\mathrm{VarCons}} \ 1$.
Luego, $(\forall i \in \mathrm{FV}(b))\ i[1 \cdot (s \circ \uparrow)] \twoheadrightarrow i$, con lo que se puede aplicar h. i.
Luego,

$$\lambda b[1 \cdot (s \circ \uparrow)] \twoheadrightarrow \lambda b$$

□

Lema 72. Sean s y t sustituciones en σ_{gc}-f. n. y sea i una variable. Si ocurre que

$$|s| = (m,n) \quad \wedge \quad |t| = (p,p) \quad \wedge \quad i > p$$

y

$$(\exists c)\ i[s] \twoheadrightarrow c \twoheadleftarrow i[t]$$

entonces

$$i[s] \twoheadrightarrow i \twoheadleftarrow i[t]$$

Demostración. Por el lema 70 sabemos que existe una única derivación a partir de $i[t]$

$$i[t] \ \to \ c_1 \ \to \ ... \ \to \ c_n = i$$

Luego, c debe pertenecer a ese camino, es decir

$$i[t] \ \to \ c_1 \ \to \ ... \ \to \ c \ \to \ ... \ \to \ c_n = i$$

Con lo que si $i[s] \twoheadrightarrow c$, entonces

$$i[s] \ \twoheadrightarrow \ c \ \to \ ... \ \to \ c_n = i$$

□

Lema 73. Sean s y t sustituciones en σ_{gc}-f. n. y sea a un término en σ_{gc}-f. n. Si ocurre que

- $s \Rrightarrow a$
- $t \not\Rrightarrow a$
- $(\forall i \in \mathrm{FV}(a))(\exists c_i)\ i[s] \ \twoheadrightarrow \ c_i \ \twoheadleftarrow \ i[t]$

Entonces $a[s] \twoheadrightarrow a$, más aún,

$$a[s] \twoheadrightarrow a \twoheadleftarrow a[t]$$

Demostración. Como $s \Subset a$, entonces $\mathrm{FV}(a) \neq \emptyset$. Luego, con $p \in \mathbb{N}$

$$|t| = (p, p) \ \wedge \ (\forall i \in \mathrm{FV}(a)) \ i > p$$

Luego por lema 72 sabemos que

$$(\forall i \in \mathrm{FV}(a)) \ i[s] \twoheadrightarrow i \twoheadleftarrow i[t]$$

y por lema 71 concluimos que

$$a[s] \quad \twoheadrightarrow \quad a \quad \leftarrow_{\mathrm{GC}} \quad a[t]$$

□

Lema 74. Sean s y t sustituciones en σ_{gc}-f. n. y sea i una variable. Si ocurre que

$$(\exists d) \ i[s] \twoheadrightarrow d \twoheadleftarrow i[t]$$

Entonces, con

$$\begin{aligned} s &= a_1 \cdot \ldots \cdot a_m \cdot \uparrow^n \\ t &= b_1 \cdot \ldots \cdot b_p \cdot \uparrow^q \end{aligned}$$

Ocurre

1. $i \leq m \wedge i \leq p \implies d \twoheadrightarrow a_i = b_i$
2. $i \leq m \wedge i > p \implies d \twoheadrightarrow a_i = i + q - p$
3. $i > m \wedge i \leq p \implies d \twoheadrightarrow b_i = i + n - m$
4. $i > m \wedge i > p \implies d \twoheadrightarrow i + n - m = i + q - p$

Demostración.

1. Por el lema 70 sabemos que $i[s] \twoheadrightarrow a_i$ y $i[t] \twoheadrightarrow b_i$. Además, ambas derivaciones son únicas, luego debe suceder que

   $$i[s] \ \to \ c_1 \ \to \ \ldots \ \to \ c_l = d \ \to \ \ldots \ \to \ c_j = a_i$$

   y

   $$i[t] \ \to \ e_1 \ \to \ \ldots \ \to \ e_h = d \ \to \ \ldots \ \to \ e_k = b_i$$

   Al existir una única única derivación desde d hasta a_i, sucede que

   $$e_{h+1} = c_{l+1} \ \wedge \ \ldots \ \wedge \ e_j = c_k$$

   Luego $a_i = b_i$. Notar que a_i y b_i se encuentran en σ_{gc}-f. n.

2, 3, 4. Misma demostración, pero cambiando los términos c_j y e_k por lo que corresponda según el lema 70.

□

Lema 75 (Reemplazo de variables libres)**.** Sea a un término en σ_{gc}-f. n., y sean s y t dos sustituciones en σ_{gc}-f. n. Si

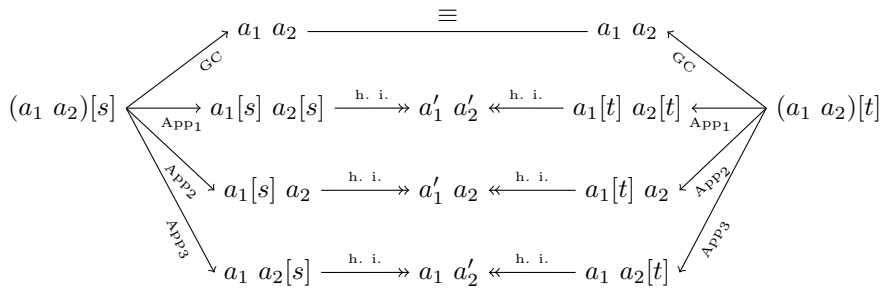
$$(\forall i \in \text{FV}(a))(\exists b_i)\ i[s] \twoheadrightarrow b_i \twoheadleftarrow i[t]$$

entonces

$$(\exists c)\ a[s] \twoheadrightarrow c \twoheadleftarrow a[t]$$

Demostración. Por inducción estructural en a:

- Caso $a = i$: Por hipótesis vale.
- Caso $a = a_1\ a_2$:



  El resto de las combinaciones corresponden a casos en los que una sustitución afecta a alguno de los subtérminos, pero la otra no. En ese caso, por hipótesis y por lema 73, se cierra el diagrama. Mostramos un ejemplo:

  Suponemos que $s \rhd a_1 \ \wedge\ s \not\rhd a_2 \ \wedge\ t \not\rhd a_1 \ \wedge\ t \rhd a_2$

$$\begin{array}{lll} (a_1\ a_2)[s] & \to_{\text{App}_2} & a_1[s]\ a_2 \\ (a_1\ a_2)[t] & \to_{\text{App}_3} & a_1\ a_2[t] \end{array}$$

  Por hipótesis,

$$(\forall i \in \text{FV}(a_1))(\exists b_i)\ i[s] \twoheadrightarrow b_i \twoheadleftarrow i[t]$$
$$(\forall j \in \text{FV}(a_2))(\exists c_j)\ j[s] \twoheadrightarrow c_j \twoheadleftarrow j[t]$$

  Luego por lema 73, $a_1[s] \twoheadrightarrow a_1$ y $a_2[t] \twoheadrightarrow a_2$. Con lo que

$$a_1[s]\ a_2 \twoheadrightarrow a_1\ a_2 \twoheadleftarrow a_1\ a_2[t]$$

- Caso $a = \lambda b$:
  Para el caso en que alguna de las sustituciones no afecte a a, por el lema 73, o simplemente aplicando GC, sabemos que

$$a[s] \quad \twoheadrightarrow \quad a \quad \twoheadleftarrow \quad a[t]$$

  Para el caso en que ambas sustituciones afecten a a:

Sean s y t

$$\begin{array}{rcl} s & = & c_1 \cdot \ldots \cdot c_m \cdot \uparrow^n \\ t & = & d_1 \cdot \ldots \cdot d_p \cdot \uparrow^q \end{array}$$

Entonces

$$\begin{array}{ll} (\lambda b)[s] & \to_{\text{Abs}} \ \lambda b[1 \cdot (s \circ \uparrow)] = \lambda b[1 \cdot ((c_1 \cdot \ldots \cdot c_m \cdot \uparrow^n) \circ \uparrow)] \\ & \twoheadrightarrow_{(\text{Map o MapGC})^m + \text{Ass}^*} \ \lambda b[1 \cdot c'_1 \cdot \ldots \cdot c'_m \cdot \uparrow^{n+1}] \end{array}$$

$$\begin{array}{ll} (\lambda b)[t] & \to_{\text{Abs}} \ \lambda b[1 \cdot (t \circ \uparrow)] = \lambda b[1 \cdot ((d_1 \cdot \ldots \cdot d_p \cdot \uparrow^q) \circ \uparrow)] \\ & \twoheadrightarrow_{(\text{Map o MapGC})^p + \text{Ass}^*} \ \lambda b[1 \cdot d'_1 \cdot \ldots \cdot d'_p \cdot \uparrow^{q+1}] \end{array}$$

Con $\quad c'_i = c_i[\uparrow] \ \vee \ c'_i = c_i$
$\quad d'_i = d_i[\uparrow] \ \vee \ d'_i = d_i$

Queremos ver que

$$(\forall i \in \text{FV}(b))(\exists e_i) \ i[1 \cdot c'_1 \cdot \ldots \cdot c'_m \cdot \uparrow^{n+1}] \twoheadrightarrow e_i \twoheadleftarrow i[1 \cdot d'_1 \cdot \ldots \cdot d'_p \cdot \uparrow^{q+1}]$$

Si $i \in \text{FV}(b)$, entonces o $i = 1$, o $i - 1 \in \text{FV}(a)$. Por hipótesis,

$$(i-1)[s] \twoheadrightarrow b_{i-1} \twoheadleftarrow (i-1)[t]$$

Si $i = 1$,

$$1[1 \cdot c'_1 \cdot \ldots \cdot c'_m \cdot \uparrow^{n+1}] \twoheadrightarrow 1 \twoheadleftarrow 1[1 \cdot d'_1 \cdot \ldots \cdot d'_p \cdot \uparrow^{q+1}]$$

Si $i > 1$,

$$i[1 \cdot c'_1 \cdot \ldots \cdot c'_m \cdot \uparrow^{n+1}] \twoheadrightarrow \begin{cases} c'_{i-1} & \text{si } i \leq m+1 \\ i + (n+1) - (m+1) & \text{si } i > m+1 \end{cases}$$

$$i[1 \cdot d'_1 \cdot \ldots \cdot d'_p \cdot \uparrow^{q+1}] \twoheadrightarrow \begin{cases} d'_{i-1} & \text{si } i \leq p+1 \\ i + (q+1) - (p+1) & \text{si } i > p+1 \end{cases}$$

Consideramos los distintos casos:

- $i \leq m+1 \ \wedge \ i \leq p+1$: Por hipótesis y por lema 74 $\ c_{i-1} = d_{i-1}$. Luego $\uparrow \ \triangleright \ c_{i-1} \iff \uparrow \ \triangleright \ d_{i-1}$, con lo que $c'_{i-1} = d'_{i-1}$. Entonces,

$$i[1 \cdot (s \circ \uparrow)] \ \twoheadrightarrow \ c'_{i-1} = d'_{i-1} \ \twoheadleftarrow \ i[1 \cdot (t \circ \uparrow)]$$

- $i \leq m+1 \wedge i > p+1$: Por hipótesis y por lema 74 $\ c_{i-1} = i-1+q-p$. Luego,

$$i[1 \cdot (s \circ \uparrow)] \ \twoheadrightarrow \ c'_{i-1} = (i-1+q-p)[\uparrow] \ \twoheadrightarrow \ i+q-p \ \twoheadleftarrow \ i[1 \cdot (t \circ \uparrow)]$$

- $i > m+1 \wedge i \leq p+1$: Por hipótesis y por lema 74 $\ d_{i-1} = i-1+n-m$. Luego,

$$i[1 \cdot (s \circ \uparrow)] \ \twoheadrightarrow \ i+n-m \ \twoheadleftarrow \ (i-1+n-m)[\uparrow] = d'_{i-1} \ \twoheadleftarrow \ i[1 \cdot (t \circ \uparrow)]$$

– $i > m+1 \ \wedge \ i > p+1$: Por hipótesis y por lema 74 $\ i-1+n-m = i-1+q-p$. Luego,

$$\begin{array}{rcl} i[1\cdot(s\circ\uparrow)] & \twoheadrightarrow & (i-1+n-m)[\uparrow] \twoheadrightarrow i+n-m = i+q-p \\ & \twoheadleftarrow & (i-1+q-p)[\uparrow] \twoheadleftarrow i[1\cdot(t\circ\uparrow)] \end{array}$$

Encontramos que

$$(\forall i \in \mathrm{FV}(b))(\exists e_i)\ i[1\cdot(s\circ\uparrow)] \twoheadrightarrow e_i \twoheadleftarrow i[1\cdot(t\circ\uparrow)]$$

con lo que por h. i.

$$(\exists b')\ \lambda b[1\cdot(s\circ\uparrow)] \twoheadrightarrow b' \twoheadleftarrow \lambda b[1\cdot(t\circ\uparrow)]$$

□

Observación 76. Sean s y t dos sustituciones, a un término, con $s = a_1 \cdot \ldots \cdot a_m \cdot \uparrow^n$ y $t \Vdash a[s]$. Entonces $(\forall i)\ 0 < i \leq m \implies t \Vdash a_i$

Demostración. Sea $|t| = (p, q)$. Como $t \Vdash a[s]$,

$$\mathrm{FV}(a[s]) = \emptyset \quad \vee \quad p = q \ \wedge \ (\forall i \in \mathrm{FV}(a[s]))\ i > p$$

Por definición de FV,

$$\mathrm{FV}(a[s]) = (\mathrm{FV}(a)_{>m} + n - m) \cup \mathrm{FV}(s) = (\mathrm{FV}(a)_{>m} + n - m) \cup \bigcup_{j=1}^{m} \mathrm{FV}(a_j)$$

Luego, para cada a_j, o $\mathrm{FV}(a_j) = \emptyset$ o $(\forall i \in \mathrm{FV}(a_j))\ i > p$. Luego, $t \Vdash a_j$. □

Lema 77. Sean s y t sustituciones en σ_{gc}-f. n., a un término en σ_{gc}-f. n. Si $t \Vdash a[s]$, entonces $(\exists c)\ a[s \circ t] \twoheadrightarrow c \twoheadleftarrow a[s]$

Demostración. Si $\mathrm{FV}(a) = \emptyset$, entonces $a[s \circ t] \ \to_{\mathrm{GC}} \ a \quad \leftarrow_{\mathrm{GC}} \quad a[s]$. Vamos a ver qué sucede en el otro caso.
Queremos ver que

$$(\forall i \in \mathrm{FV}(a))\ (\exists d_i)\ i[s \circ t] \ \twoheadrightarrow d_i \ \twoheadleftarrow \ i[s]$$

para concluir con el lema 75 que

$$(\exists c)\ a[s \circ t] \twoheadrightarrow c \twoheadleftarrow a[s]$$

Por estar en forma normal,

$$s = b_1 \cdot \ldots \cdot b_m \cdot \uparrow^n \qquad t = c_1 \cdot \ldots \cdot c_p \cdot \uparrow^q$$

Por observación 76, como $t \Vdash a[s]$, $(\forall i)\ 0 < i \leq m \implies t \Vdash b_i$. Entonces

$$s \circ t \ \to_{\mathrm{MapGC}^m} \ b_1 \cdot \ldots \cdot b_m \cdot (\uparrow^n \circ t)$$

Luego,

$$b_1 \cdot ... \cdot b_m \cdot (\uparrow^n \circ t) \twoheadrightarrow_{\text{Ass + ShiftCons}} \begin{cases} b_1 \cdot ... \cdot b_m \cdot c_{n+1} \cdot ... \cdot c_p \cdot \uparrow^q & \text{si } n < p \\ b_1 \cdot ... \cdot b_m \cdot \uparrow^{n+q-p} & \text{si } n \geq p \end{cases}$$

En el caso que $\text{FV}(a[s]) = \emptyset$, por definición de FV sucede que $\text{FV}(a)_{>m} = \emptyset$. Luego, $(\forall i \in \text{FV}(a))\ i \leq m$, con lo que $i[s \circ t] \twoheadrightarrow b_i \twoheadleftarrow i[s]$.

Si $\text{FV}(a[s]) \neq \emptyset$, entonces $p = q$. Analizamos los casos:

1. Si $n \geq p$, entonces $s \circ t \twoheadrightarrow b_1 \cdot ... \cdot b_m \cdot \uparrow^n = s$, con lo que $a[s \circ t] \twoheadrightarrow a[s]$.
2. Si $n < p$, sea $i \in \text{FV}(a)$.
   - Si $i \leq m$, entonces $i[s] \twoheadrightarrow b_i \twoheadleftarrow i[s \circ t]$.
   - Si $i > m$, entonces $i[s] \twoheadrightarrow i + n - m$. Como $t \not\Vdash a[s]$, significa que

     $$(\forall j \in \text{FV}(a[s]))\ j > p \iff (\forall j \in (\text{FV}(a)_{>m} + n - m) \cup \text{FV}(s)))\ j > p$$

     Con lo que en particular, al ser $i \in \text{FV}(a)$, $i > m$,

     $$i + n - m > p \iff i > p + m - n$$

     Como hay $p + m - n$ términos en $b_1 \cdot ... \cdot b_m \cdot c_{n+1} \cdot ... \cdot c_p \cdot \uparrow^q$, por lema 70 tenemos que

     $$i[s \circ t] \twoheadrightarrow i + q - (p + m - n) = i + n - m \twoheadleftarrow i[s]$$

   Luego, $(\forall i \in \text{FV}(a))(\exists d_i)\ i[s \circ t] \twoheadrightarrow d_i \twoheadleftarrow i[s]$. Luego por lema 75 obtenemos que $(\exists d)\ a[s \circ t] \twoheadrightarrow d \twoheadleftarrow a[s]$

En ambos casos se tiene que existe d tal que $a[s \circ t] \twoheadrightarrow d \twoheadleftarrow a[s]$. □

Lema 78 (ClosGcR)**.** Sean s y t sustituciones, a un término. Si $t \not\Vdash a[s]$, entonces $(\exists c)\ a[s \circ t] \twoheadrightarrow c \twoheadleftarrow a[s]$.

Demostración. Sean

$$\begin{aligned} s' &= {\sigma_{\text{gc}}}'(s) \\ t' &= {\sigma_{\text{gc}}}'(t) \\ a' &= {\sigma_{\text{gc}}}'(a) \end{aligned}$$

Luego

$$\begin{aligned} a[s \circ t] &\twoheadrightarrow a'[s' \circ t'] \\ a[s] &\twoheadrightarrow a'[s'] \end{aligned}$$

Por el lema 67, $t' \not\Vdash a'[s']$. Luego, por lema 77,

$$(\exists c)\ a'[s' \circ t'] \twoheadrightarrow c \twoheadleftarrow a'[s']$$

Luego

$$(\exists c)\ a[s \circ t] \twoheadrightarrow c \twoheadleftarrow a[s]$$

□

Lema 79. Sean s y t sustituciones en σ_{gc}-f. n., a un término en σ_{gc}-f. n. Si $s \Rsh a$, entonces $(\exists c)\ a[s \circ t] \twoheadrightarrow c \twoheadleftarrow a[t]$.

Demostración. Si $\text{FV}(a) = \emptyset$, entonces $a[s \circ t] \ \to_{\text{GC}}\ a \quad \leftarrow_{\text{GC}}\quad a[t]$. Vamos a ver qué sucede en el otro caso.
Queremos ver que

$$(\forall i \in a)\ (\exists d_i)\ i[s \circ t] \ \twoheadrightarrow d_i \ \twoheadleftarrow\ i[t]$$

para concluir con el lema 75 que

$$(\exists c)\ a[s \circ t] \twoheadrightarrow c \twoheadleftarrow a[t]$$

Por estar en forma normal,

$$s = b_1 \cdot \ldots \cdot b_m \cdot \uparrow^n \qquad t = c_1 \cdot \ldots \cdot c_p \cdot \uparrow^q$$

Como estamos en el caso en el que $\text{FV}(a) \neq \emptyset$, debe suceder que $m = n$ y $(\forall i \in \text{FV}(a))\ i > m$. De ahora en más, utilizaremos m en vez de n.
Sea $b'_i = b_i[t] \ \vee\ b'_i = b_i$ si $t \Rsh b_i$ o no, respectivamente. Entonces

$$s \circ t \ \to_{(\text{Map o MapGC})^m}\ b'_1 \cdot \ldots \cdot b'_m \cdot (\uparrow^m \circ t)$$

Luego,

$$b'_1 \cdot \ldots \cdot b'_m \cdot (\uparrow^m \circ t) \twoheadrightarrow_{\text{Ass + ShiftCons}} \begin{cases} b'_1 \cdot \ldots \cdot b'_m \cdot c_{m+1} \cdot \ldots \cdot c_p \cdot \uparrow^q & \text{si } m < p \\ b'_1 \cdot \ldots \cdot b'_m \cdot \uparrow^{m+q-p} & \text{si } m \geq p \end{cases}$$

Sea $i \in \text{FV}(a)$, sabiendo que $i > m$, analizamos según el caso:

1. Si $m < p$, entonces,

$$i[s \circ t] \ \twoheadrightarrow\ i[b'_1 \cdot \ldots \cdot b'_m \cdot c_{m+1} \cdot \ldots \cdot c_p \cdot \uparrow^q] \ \twoheadrightarrow_{\text{L 70}}\ c_i \ \twoheadleftarrow_{\text{L 70}}\ i[t]$$

2. Si $m \geq p$, entonces $i > p$, con lo que

$$i[s \circ t] \ \twoheadrightarrow\ i[b'_1 \cdot \ldots \cdot b'_m \cdot \uparrow^{m+q-p}] \ \twoheadrightarrow_{\text{L 70}}\ i + (m+q-p) - m = i + q - p \ \twoheadleftarrow_{\text{L 70}}\ i[t]$$

Luego, $(\forall i \in \text{FV}(a))(\exists d_i)\ i[s \circ t] \twoheadrightarrow d_i \twoheadleftarrow i[t]$. Luego por lema 75 obtenemos que $(\exists d)\ a[s \circ t] \twoheadrightarrow d \twoheadleftarrow a[t]$ □

Lema 80 (ClosGcL)**.** Sean s y t sustituciones, a un término. Si $s \Rsh a$, entonces $(\exists c)\ a[s \circ t] \twoheadrightarrow c \twoheadleftarrow a[t]$.

Demostración. Sean

$$\begin{aligned} s' &= {\sigma_{\text{gc}}}'(s) \\ t' &= {\sigma_{\text{gc}}}'(t) \\ a' &= {\sigma_{\text{gc}}}'(a) \end{aligned}$$

Luego

$$\begin{aligned} a[s \circ t] &\twoheadrightarrow a'[s' \circ t'] \\ a[t] &\twoheadrightarrow a'[t'] \end{aligned}$$

Por el lema 67, $s' \not\Rightarrow a'$. Luego, por lema 79,

$$(\exists c)\ a'[s' \circ t'] \twoheadrightarrow c \twoheadleftarrow a'[t']$$

Luego

$$(\exists c)\ a[s \circ t] \twoheadrightarrow c \twoheadleftarrow a[t]$$

□

Lema 81. Sean a un término y s una sustitución tal que $s \Rrightarrow \lambda a$, entonces $s' = 1 \cdot (s \circ \uparrow) \Rrightarrow a$.

Demostración. Sea $|s| = (m, n)$. Como $s \Rrightarrow \lambda a$, entonces $\mathrm{FV}(\lambda a) \neq \emptyset$ y, o bien $m \neq n$, o existe $i \in \mathrm{FV}(\lambda a)$ tal que $i \leq m$. Notar que $\mathrm{FV}(a) \neq \emptyset$. Luego,

1. Si $m \neq n$, entonces $|s'| = (m+1, n+1)$ cumple con $m+1 \neq n+1$, luego $s' \Rrightarrow a$.
2. Si existe $i \in \mathrm{FV}(\lambda a)$ tal que $i \leq m$, por definición de FV vale que $i+1 \in \mathrm{FV}(a)$, con lo que $i+1 \leq m+1$, y nuevamente $s' \Rrightarrow a$.

□

Lema 82. Sean s una sustitución y a un término,

$$s \not\Rightarrow \lambda a \implies (\exists c)\ \lambda a[1 \cdot (s \circ \uparrow)] \twoheadrightarrow c \twoheadleftarrow \lambda a$$

Demostración. Sean:

$$|s| = (m, n) \qquad a' = {\sigma_{\mathrm{gc}}}'(a) \qquad t' = {\sigma_{\mathrm{gc}}}'(1 \cdot (s \circ \uparrow))$$

Vamos a ver que

$$(\forall i \in \mathrm{FV}(a'))\ i[t'] \twoheadrightarrow i$$

Para concluir por lema 71 que

$$\lambda a[1 \cdot (s \circ \uparrow)] \twoheadrightarrow \lambda a'[t'] \twoheadrightarrow \lambda a' \twoheadleftarrow \lambda a$$

Notar que $t' = 1 \cdot {\sigma_{\mathrm{gc}}}'(s \circ \uparrow)$
Por definición de $\not\Rightarrow$, tenemos dos casos:

- $\mathrm{FV}(\lambda a) = \emptyset$: Como $\mathrm{FV}(\lambda a) = \mathrm{FV}(a) \setminus 1 = \emptyset$, se pueden dar dos casos:
  - $\mathrm{FV}(a) = \emptyset$: En este caso, se llega directamente a la conclusión sin utilizar el lema:
    $$\lambda a[1 \cdot (s \circ \uparrow)] \ \to_{\mathrm{GC}}\ \lambda a$$
  - $\mathrm{FV}(a) = \{1\}$: Mostramos que se aplican las hipótesis del lema 71. Como
    $$1[t'] \ \to_{\mathrm{VarCons}}\ 1$$
    y por corolario 66 $\mathrm{FV}(a') \subseteq \mathrm{FV}(a)$, se tiene que
    $$(\forall i \in \mathrm{FV}(a'))\ i[t'] \twoheadrightarrow i$$

- $\mathrm{FV}(\lambda a) \neq \emptyset$: Entonces $m = n$ y $(\forall i \in \mathrm{FV}(\lambda a))\ i > m$. Luego, como $|1 \cdot (s \circ \uparrow)| = (m+1, m+1)$, para toda variable $i \in \mathrm{FV}(a)$ tenemos

$$\begin{array}{lll} \text{si } i > 1 & \Longrightarrow & i > m+1 \ \wedge\ i[t'] \twoheadrightarrow_{\mathrm{L}\ 70} i \\ \text{si } i = 1 & \Longrightarrow & 1[1 \cdot (s \circ \uparrow)] \ \rightarrow_{\mathrm{VarCons}}\ 1 \end{array}$$

  Luego, como por corolario 66 $\mathrm{FV}(a') \subseteq \mathrm{FV}(a)$,

$$(\forall i \in \mathrm{FV}(a'))\ i[t'] \twoheadrightarrow i$$

□

Corolario 83. Sean s, t sustituciones y a un término,

$$s \circ t \not\Vdash \lambda a \implies (\exists c)\ \lambda a[1 \cdot (s \circ (t \circ \uparrow))] \twoheadrightarrow c \twoheadleftarrow \lambda a$$

Demostración. La demostración es igual a la anterior, puesto que para $|s \circ t| = (m, n)$,

$$|1 \cdot (s \circ (t \circ \uparrow))| = |1 \cdot ((s \circ t) \circ \uparrow)| = (m+1, n+1)$$

□

Capítulo 5

Confluencia de $\sigma_{\mathbf{gc}}$

En el presente capítulo se muestra que para términos cerrados (*i.e.* sin metavariables) se cumple la propiedad de confluencia para el cálculo asociado. Para ello, vamos a mostrar en 5.1 que el cálculo es *localmente confluente* (WCR), para concluir por SN (ver capítulo 3) que el cálculo es *confluente* (CR).

5.1 Confluencia local de $\sigma_{\mathbf{gc}}$

En la presente sección se muestra que para términos cerrados, se cumple la propiedad de confluencia débil o local para el cálculo asociado. Para ello, utilizaremos el Critical Pair Lemma (lema 11), mostrando que todos los pares críticos del cálculo pueden cerrarse, es decir, si (t', t'') es un par crítico, entonces existe t''' tal que $t' \twoheadrightarrow t''' \twoheadleftarrow t''$

En el contexto de un TRS infinito, se tendrán infinitos pares críticos. Por ello, los pares críticos conformados por reglas-esquemas en realidad serán esquemas de pares críticos. Al mostrar la convergencia de un par crítico-esquema, se estará mostrando la convergencia de todos los pares críticos definidos por él. De aquí en adelante no se hará distinción entre par crítico y par crítico-esquema.

A continuación mostramos la lista de todos los posibles pares críticos de σ_{gc}, sin tener en cuenta las condiciones de las reglas. Caso por caso, analizamos cuando dichas condiciones no permiten la superposición. Como notación utilizaremos $t \rightrightarrows (t_1, t_2)$ para indicar al par crítico (t_1, t_2) que nace del término t.

1. GC + VarCons: $1[a \cdot s] \rightrightarrows (1, a)$. No es posible, puesto que $a \cdot s \Rsh 1$.
2. GC + App$_1$: $(a\ b)[s] \rightrightarrows (a\ b, a[s]\ b[s])$. No es posible, puesto que por lema 45 $s \not\Rsh a\ b \iff s \not\Rsh a \wedge s \not\Rsh b$.
3. GC + App$_2$: $(a\ b)[s] \rightrightarrows (a\ b, a[s]\ b)$. No es posible por mismo razonamiento que en el item 2.
4. GC + App$_3$: $(a\ b)[s] \rightrightarrows (a\ b, a\ b[s])$. No es posible por mismo razonamiento que en el item 2.
5. GC + Abs: $(\lambda a)[s] \rightrightarrows (\lambda a, \lambda a[1 \cdot (s \circ \uparrow)])$. No es posible pues las condiciones son excluyentes.

6. GC + Clos: $a[s][t] \rightrightarrows (a[t], a[s \circ t])$. No es posible pues las condiciones son excluyentes.
7. GC + Clos: $a[s][t] \rightrightarrows (a[s], a[s \circ t])$. No es posible pues las condiciones son excluyentes.
8. GC + ClosGC: $a[s][t] \rightrightarrows (a[s], a)$. No es posible pues las condiciones son excluyentes.
9. GC + ClosGC: $a[s][t] \rightrightarrows (a[t], a)$. No es posible pues las condiciones son excluyentes.
10. VarCons + Clos: $1[a \cdot s][t] \rightrightarrows (a[t], 1[(a \cdot s) \circ t])$. Será analizado más adelante.
11. VarCons + ClosGC: $1[a \cdot s][t] \rightrightarrows (a[t], 1)$. No es posible pues las condiciones son excluyentes.
12. App_i + App_j: para $i, j \in [1, 3]$. No es posible pues las condiciones son excluyentes.
13. App_i + Clos: para $i \in [1, 3]$. Serán analizados más adelante.
14. App_i + ClosGC: para $i \in [1, 3]$. Serán analizados más adelante.
15. Abs + Clos: $(\lambda a)[s][t] \rightrightarrows ((\lambda a[1 \cdot (s \circ \uparrow)])[t], (\lambda a)[s \circ t])$. Será analizado más adelante.
16. Abs + ClosGC: $(\lambda a)[s][t] \rightrightarrows ((\lambda a[1 \cdot (s \circ \uparrow)])[t], \lambda a)$. Será analizado más adelante.
17. Clos + Clos: $a[s][t][u] \rightrightarrows (a[s \circ t][u], a[s][t \circ u])$. Será analizado más adelante.
18. Clos + ClosGC: $a[s][t][u] \rightrightarrows (a[s \circ t][u], a[s])$. Será analizado más adelante.
19. ClosGC + Clos: $a[s][t][u] \rightrightarrows (a[u], a[s][t \circ u])$. Será analizado más adelante.
20. ClosGC + ClosGC: $a[s][t][u] \rightrightarrows (a[u], a[s])$. Será analizado más adelante.
21. IdL + Ass: $(\mathbf{id} \circ s) \circ t \rightrightarrows (s \circ t, \mathbf{id} \circ (s \circ t))$. Será analizado más adelante.
22. ShiftId + Ass: $(\uparrow \circ \mathbf{id}) \circ t \rightrightarrows (\uparrow \circ t, \uparrow \circ (\mathbf{id} \circ t))$. Será analizado más adelante.
23. ShiftCons + Ass: $(\uparrow \circ (a \cdot s)) \circ t \rightrightarrows (s \circ t, \uparrow \circ ((a \cdot s) \circ t))$. Será analizado más adelante.
24. Map + Ass: $((a \cdot s) \circ t) \circ u \rightrightarrows ((a[t] \cdot (s \circ t)) \circ u, (a \cdot s) \circ (t \circ u))$. Será analizado más adelante.
25. Map + MapGC: $(a \cdot s) \circ t \rightrightarrows (a[t] \cdot (s \circ t), a \cdot (s \circ t))$. No es posible pues las condiciones son excluyentes.
26. MapGC + Ass: $((a \cdot s) \circ t) \circ u \rightrightarrows ((a \cdot (s \circ t)) \circ u, (a \cdot s) \circ (t \circ u))$. Será analizado más adelante.
27. Ass + Ass: $((s \circ t) \circ u) \circ v \rightrightarrows ((s \circ (t \circ u)) \circ v, (s \circ t) \circ (u \circ v))$. Será analizado más adelante.

Para presentar a los pares críticos vamos a utilizar la siguiente notación:

Regla1 + Regla2

$condicion_1$, $condicion_2$, $\ldots$, $condicion_n$

para indicar el análisis del par crítico generado por las reglas *Regla1* y *Regla2*, bajo la condición $condicion_1 \wedge condicion_2 \wedge \ldots \wedge condicion_n$

Luego, analizaremos las derivaciones por separado, marcando con un rectángulo numerado cuando las derivaciones encuentran un término en común. Según las reglas que se puedan aplicar, puede ser que las derivaciones se bifurquen según determinadas condiciones. Estos casos serán anotados como subíndices de la derivación original, con las condiciones remarcadas en cada caso:

$$
\begin{array}{lll}
1. & t & \to_{\text{Regla1}}\ t' \\
\quad i. & (\text{si } condicion_i) & \to_{\text{Regla}i}\ \ldots\ \to_{\text{Regla}k}\ \boxed{t'''_i}^{1} \\
\quad ii. & (\text{si } condicion_{ii}) & \to_{\text{Regla}ii}\ \ldots\ \to_{\text{Regla}kk}\ \boxed{t'''_{ii}}^{2} \\
2. & t & \to_{\text{Regla2}}\ t'' \\
\quad i. & (\text{si } condicion_i) & \to_{\text{Regla}j}\ \ldots\ \to_{\text{Regla}l}\ \boxed{t'''_i}^{1} \\
\quad ii. & (\text{si } condicion_{ii}) & \to_{\text{Regla}jj}\ \ldots\ \to_{\text{Regla}ll}\ \boxed{t'''_{ii}}^{2}
\end{array}
$$

VarCons + Clos

$t \rhd 1[a \cdot s]$

$$
\begin{array}{lll}
1. & 1[a \cdot s][t] & \to_{\text{VarCons}}\ \boxed{a[t]}^{1} \\
\quad i. & (\text{si } t \not\rhd a) & \to_{\text{GC}}\ \boxed{a}^{2} \\
2. & 1[a \cdot s][t] & \to_{\text{Clos}}\ 1[(a \cdot s) \circ t] \\
\quad i. & (\text{si } t \rhd a) & \to_{\text{Map}}\ 1[a[t] \cdot (s \circ t)] \\
 & & \to_{\text{VarCons}}\ \boxed{a[t]}^{1} \\
\quad ii. & (\text{si } t \not\rhd a) & \to_{\text{MapGC}}\ 1[a \cdot (s \circ t)] \\
 & & \to_{\text{VarCons}}\ \boxed{a}^{2}
\end{array}
$$

App$_1$ + Clos

$s \rhd a_1$, $s \rhd a_2$, $t \rhd (a_1\ a_2)[s]$, $s \circ t \rhd a_1\ a_2$

$$
\begin{array}{lll}
1. & (a_1\ a_2)[s][t] & \to_{\text{Clos}}\ (a_1\ a_2)[s \circ t] \\
2. & (a_1\ a_2)[s][t] & \to_{\text{App}_1}\ (a_1[s]\ a_2[s])[t]
\end{array}
$$

Vemos como se cierran las distintas derivaciones, separando por casos. Notar que

$$t \Rrightarrow (a_1\ a_2)[s] \implies t \Rrightarrow (a_1[s]\ a_2[s])$$

pues $\mathrm{FV}((a\ b)[s]) = \mathrm{FV}(a[s]\ b[s])$

- $s \circ t \Rrightarrow a_1, \quad s \circ t \Rrightarrow a_2$

  1. $(a_1\ a_2)[s \circ t] \quad \to_{\mathrm{App}_1} \boxed{a_1[s \circ t]\ a_2[s \circ t]}$

  2. $(a_1[s]\ a_2[s])[t]$

     i. (si $t \Rrightarrow a_1[s] \wedge t \Rrightarrow a_2[s]$)
     $\to_{\mathrm{App}_1} a_1[s][t]\ a_2[s][t]$
     $\to_{\mathrm{Clos}\ \times 2} \boxed{a_1[s \circ t]\ a_2[s \circ t]}$

     ii. (si $t \Rrightarrow a_1[s] \wedge t \not\Rightarrow a_2[s]$)
     $\to_{\mathrm{App}_2} a_1[s][t]\ a_2[s]$
     $\to_{\mathrm{Clos}} a_1[s \circ t]\ a_2[s]$

     iii. (si $t \not\Rightarrow a_1[s] \wedge t \Rrightarrow a_2[s]$)
     $\to_{\mathrm{App}_3} a_1[s]\ a_2[s][t]$
     $\to_{\mathrm{Clos}} a_1[s]\ a_2[s \circ t]$

  Los casos *2.ii* y *2.iii* se cierran con el lema 78.

- $s \circ t \Rrightarrow a_1, \quad s \circ t \not\Rightarrow a_2$, entonces por lema 60.5 $t \Rrightarrow a_2[s]$

  1. $(a_1\ a_2)[s \circ t] \quad \to_{\mathrm{App}_2} \boxed{a_1[s \circ t]\ a_2}$

  2. $(a_1[s]\ a_2[s])[t]$

     i. (si $t \Rrightarrow a_1[s]$)
     $\to_{\mathrm{App}_1} a_1[s][t]\ a_2[s][t]$
     $\to_{\mathrm{Clos} + \mathrm{ClosGC}} \boxed{a_1[s \circ t]\ a_2}$

     ii. (si $t \not\Rightarrow a_1[s]$)
     $\to_{\mathrm{App}_3} a_1[s]\ a_2[s][t]$
     $\to_{\mathrm{ClosGC}} a_1[s]\ a_2$

  El caso *2.ii* se cierra con el lema 78.

- $s \circ t \not\Rightarrow a_1, \quad s \circ t \Rrightarrow a_2$, entonces por lema 60.5 $t \Rrightarrow a_1[s]$. Luego se cierra de forma análoga al caso anterior.

$\mathbf{App_1}$ + ClosGC

$s \Rrightarrow a_1, \quad s \Rrightarrow a_2, \quad t \Rrightarrow (a_1\ a_2)[s], \quad s \circ t \not\Rightarrow a_1\ a_2$
Por lema 45, $s \circ t \not\Rightarrow a_1$ y $s \circ t \not\Rightarrow a_2$. Luego, por lema 60.5, $t \Rrightarrow a_1[s]$ y $t \Rrightarrow a_2[s]$.

1. $(a_1\ a_2)[s][t] \quad \to_{\text{ClosGC}} \boxed{a_1\ a_2}$

2. $(a_1\ a_2)[s][t] \quad \to_{\text{App}_1} (a_1[s]\ a_2[s])[t]$
   $\to_{\text{App}_1} a_1[s][t]\ a_2[s][t]$
   $\to_{\text{ClosGC}\ \times 2} \boxed{a_1\ a_2}$

App_2 + Clos

$s \Rightarrow a_1, \quad s \not\Rightarrow a_2, \quad t \Rightarrow (a_1\ a_2)[s], \quad s \circ t \Rightarrow a_1\ a_2.$

1. $(a_1\ a_2)[s][t] \quad \to_{\text{App}_2} (a_1[s]\ a_2)[t]$

2. $(a_1\ a_2)[s][t] \quad \to_{\text{Clos}} (a_1\ a_2)[s \circ t]$

Analizamos los distintos casos:

1. $t \Rightarrow a_1[s], \quad t \Rightarrow a_2$. Por lema 60.2, $s \circ t \Rightarrow a_2$.

   1. $(a_1[s]\ a_2)[t] \quad \to_{\text{App}_1} a_1[s][t]\ a_2[t]$
      i. (si $s \circ t \Rightarrow a_1$) $\to_{\text{Clos}} a_1[s \circ t]\ a_2[t]$

      ii. (si $s \circ t \not\Rightarrow a_1$) $\to_{\text{ClosGC}} a_1\ a_2[t]$

   2. $(a_1\ a_2)[s \circ t]$
      i. (si $s \circ t \Rightarrow a_1$) $\to_{\text{App}_1} a_1[s \circ t]\ a_2[s \circ t]$

      ii. (si $s \circ t \not\Rightarrow a_1$) $\to_{\text{App}_3} a_1\ a_2[s \circ t]$

   Los casos *1.i* y *2.i* se cierran con el lema 80. Los mismo sucede con los casos *1.ii* y *2.ii*. No hay otra combinación posible.

2. $t \Rightarrow a_1[s], \quad t \not\Rightarrow a_2$. Por lema 60.3, $s \circ t \not\Rightarrow a_2$. Por hipótesis, $s \circ t \Rightarrow a_1\ a_2$, luego $s \circ t \Rightarrow a_1$.

   1. $(a_1[s]\ a_2)[t] \quad \to_{\text{App}_2} a_1[s][t]\ a_2$
      $\to_{\text{Clos}} \boxed{a_1[s \circ t]\ a_2}$

   2. $(a_1\ a_2)[s \circ t] \quad \to_{\text{App}_2} \boxed{a_1[s \circ t]\ a_2}$

3. $t \not\Rightarrow a_1[s], \quad t \Rightarrow a_2$. Por lema 60.1, $s \circ t \Rightarrow a_1$ y por lema 60.2, $s \circ t \Rightarrow a_2$.

   1. $(a_1[s]\ a_2)[t] \quad \to_{\text{App}_3} a_1[s]\ a_2[t]$

   2. $(a_1\ a_2)[s \circ t] \quad \to_{\text{App}_1} a_1[s \circ t]\ a_2[s \circ t]$

El diagrama se cierra con los lemas 80 y 78.

4. $t \Rsh a_1[s], \quad t \Rsh a_2$. Vamos a mostrar que este caso no puede suceder.

   Vamos a mostrar que $\mathrm{FV}((a_1\ a_2)[s]) = \mathrm{FV}(a_1[s]\ a_2)$, para concluir que si $t \Rsh a_1[s]$ y $t \Rsh a_2$, entonces $t \Rsh (a_1\ a_2)[s]$, lo que es absurdo.

   Sea $|s| = (m, n)$ y $|t| = (p, q)$. Por definición de $\Rsh$, como $s \Rsh a_2$ debe suceder que $m = n$ y $(\forall i \in \mathrm{FV}(a_2))\ i > m$, o $\mathrm{FV}(a_2) = \emptyset$.

   $$\begin{aligned}\mathrm{FV}((a_1\ a_2)[s]) &= (\mathrm{FV}(a_1\ a_2)_{>m} + n - m) \cup \mathrm{FV}(s)\\ &= (\mathrm{FV}(a_1)_{>m} + n - m) \cup \mathrm{FV}(s) \cup (\mathrm{FV}(a_2)_{>m} + n - m)\\ &= \mathrm{FV}(a_1[s]) \cup \mathrm{FV}(a_2)\\ &= \mathrm{FV}(a_1[s]\ a_2)\end{aligned}$$

   Como $t \Rsh a_1[s]$ y $t \Rsh a_2$, se tiene que dar uno de los siguientes casos:

   - $\mathrm{FV}(a_1[s]) = \mathrm{FV}(a_2) = \emptyset$: entonces $\mathrm{FV}(a_1[s]\ a_2) = \mathrm{FV}((a_1\ a_2)[s]) = \emptyset$, luego $t \Rsh (a_1\ a_2)[s]$, absurdo.
   - $p = q$ y $(\forall i \in \mathrm{FV}(a_1[s]) \cup \mathrm{FV}(a_2))\ i > p$. Como $\mathrm{FV}(a_1[s]) \cup \mathrm{FV}(a_2) = \mathrm{FV}(a_1[s]\ a_2) = \mathrm{FV}((a_1\ a_2)[s])$ se tiene $t \Rsh (a_1\ a_2)[s]$, absurdo.

   En ambos casos se llega al absurdo, entonces no puede suceder que $t \Rsh a_1[s]$ y $t \Rsh a_2$.

$\mathbf{App_2}$ + ClosGC

$s \Rtr a_1, \quad s \Rsh a_2, \quad t \Rtr (a_1\ a_2)[s], \quad s \circ t \Rsh a_1\ a_2$
Por lema 45, $s \circ t \Rsh a_1\ a_2 \implies s \circ t \Rsh a_1 \ \wedge\ s \circ t \Rsh a_2$. Luego, por lema 60.5, $t \Rtr a_1[s]$ y por lema 60.7, $t \Rsh a_2$.

1. $(a_1\ a_2)[s][t] \quad \to_{\mathrm{ClosGC}} \quad \boxed{a_1\ a_2}$

2. $$\begin{aligned}(a_1\ a_2)[s][t] &\to_{\mathrm{App_2}} (a_1[s]\ a_2)[t]\\ &\to_{\mathrm{App_2}} a_1[s][t]\ a_2\\ &\to_{\mathrm{ClosGC}} \boxed{a_1\ a_2}\end{aligned}$$

$\mathbf{App_3}$ + Clos

$s \Rsh a_1, \quad s \Rtr a_2, \quad t \Rtr (a_1\ a_2)[s], \quad s \circ t \Rtr a_1\ a_2$
Demostración análoga al par crítico $\mathrm{App_2}$ + Clos

$\mathbf{App_3}$ + ClosGC

$s \Rsh a_1, \quad s \Rtr a_2, \quad t \Rtr (a_1\ a_2)[s], \quad s \circ t \Rsh a_1\ a_2$
Demostración análoga al par crítico $\mathrm{App_2}$ + ClosGC

Abs + Clos

$s \Rrightarrow \lambda a, \quad t \Rrightarrow (\lambda a)[s], \quad s \circ t \Rrightarrow \lambda a$

1. $(\lambda a)[s][t] \quad \to_{\text{Abs}} \ (\lambda a[1 \cdot (s\circ \uparrow)])[t]$
   $\to_{\text{Abs}} \ \lambda a[1 \cdot (s\circ \uparrow)][1 \cdot (t\circ \uparrow)]$
   $\to_{\text{Clos}} \ \lambda a[(1 \cdot (s\circ \uparrow)) \circ (1 \cdot (t\circ \uparrow))]$
   $\to_{\text{Map}} \ \lambda a[1[1 \cdot (t\circ \uparrow)] \cdot ((s\circ \uparrow) \circ (1 \cdot (t\circ \uparrow)))]$
   $\to_{\text{VarCons}} \ \lambda a[1 \cdot ((s\circ \uparrow) \circ (1 \cdot (t\circ \uparrow)))]$
   $\to_{\text{Ass}} \ \lambda a[1 \cdot (s \circ (\uparrow \circ (1 \cdot (t\circ \uparrow))))]$
   $\to_{\text{ShiftCons}} \ \boxed{\lambda a[1 \cdot (s \circ (t\circ \uparrow))]}$

2. $(\lambda a)[s][t] \quad \to_{\text{Clos}} \ (\lambda a)[s \circ t]$
   $\to_{\text{Abs}} \ \lambda a[1 \cdot ((s \circ t)\circ \uparrow)]$
   $\to_{\text{Ass}} \ \boxed{\lambda a[1 \cdot (s \circ (t\circ \uparrow))]}$

Notar que en la primera derivación el segundo paso de *Abs* se puede realizar puesto que $\text{FV}((\lambda a)[s]) = \text{FV}(\lambda a[1 \cdot (s\circ \uparrow)])$
Notar que para aplicar *Clos* debemos asegurar que

- $1 \cdot (s\circ \uparrow) \Rrightarrow a$. El lema 81 asegura esta condición.
- $1 \cdot (t\circ \uparrow) \Rrightarrow a[1 \cdot (s\circ \uparrow)]$. Anteriormente vimos que $t \Rrightarrow \lambda a[1 \cdot (s\circ \uparrow)]$. Luego, por lema 81 se tiene que $1 \cdot (t\circ \uparrow) \Rrightarrow a[1 \cdot (s\circ \uparrow)]$
- Sea $u = (1 \cdot (s\circ \uparrow)) \circ (1 \cdot (t\circ \uparrow))$, $u \Rrightarrow a$. Como $s \circ t \Rrightarrow \lambda a$, por lema 81 obtenemos que $1 \cdot ((s \circ t)\circ \uparrow) \Rrightarrow a$. Fácilmente se comprueba que

  $$u \quad \twoheadrightarrow \quad 1 \cdot (s \circ (t\circ \uparrow))$$

  y como

  $$1 \cdot ((s \circ t)\circ \uparrow) \ \to_{\text{Ass}} \ 1 \cdot (s \circ (t\circ \uparrow))$$

  por corolario 64 se tiene que

  $$|u| = |1 \cdot (s \circ (t\circ \uparrow))| = |1 \cdot ((s \circ t)\circ \uparrow)|$$

  Luego $u \Rrightarrow a \iff 1 \cdot ((s \circ t)\circ \uparrow) \Rrightarrow a$, luego $u \Rrightarrow a$.

Abs + ClosGC

$s \Rrightarrow \lambda a, \quad t \Rrightarrow (\lambda a)[s], \quad s \circ t \not\Rrightarrow \lambda a$

1.	$(\lambda a)[s][t]$	$\to_{\text{Abs}}\ (\lambda a[1\cdot(s\circ\uparrow)])[t]$
		$\to_{\text{Abs}}\ \lambda a[1\cdot(s\circ\uparrow)][1\cdot(t\circ\uparrow)]$
	i. (si $(1\cdot(s\circ\uparrow))\circ(1\cdot(t\circ\uparrow))\Rrightarrow a$)	$\to_{\text{Clos}}\ \lambda a[(1\cdot(s\circ\uparrow))\circ(1\cdot(t\circ\uparrow))]$
		$\to_{\text{Map}}\ \lambda a[1[1\cdot(t\circ\uparrow)]\cdot((s\circ\uparrow)\circ(1\cdot(t\circ\uparrow)))]$
		$\to_{\text{VarCons}}\ \lambda a[1\cdot((s\circ\uparrow)\circ(1\cdot(t\circ\uparrow)))]$
		$\to_{\text{Ass}}\ \lambda a[1\cdot(s\circ(\uparrow\circ(1\cdot(t\circ\uparrow))))]$
		$\to_{\text{ShiftCons}}\ \lambda a[1\cdot(s\circ(t\circ\uparrow))]$
	ii. (si $(1\cdot(s\circ\uparrow))\circ(1\cdot(t\circ\uparrow))\not\Rrightarrow a$)	$\to_{\text{ClosGC}}\ \boxed{\lambda a}$
2.	$(\lambda a)[s][t]$	$\to_{\text{ClosGC}}\ \boxed{\lambda a}$

Las condiciones $t\Rrightarrow\lambda a[1\cdot(s\circ\uparrow)]$, $1\cdot(s\circ\uparrow)\Rrightarrow a$, y $1\cdot(t\circ\uparrow)\Rrightarrow a[1\cdot(s\circ\uparrow)]$ necesarias en la primera derivación se justifican de igual manera que en la demostración anterior.

Un reducto común entre la derivación *1.i* y *2* puede encontrarse gracias al corolario 83.

Clos + Clos

$s\Rrightarrow a,\quad t\Rrightarrow a[s],\quad u\Rrightarrow a[s][t],\quad s\circ t\Rrightarrow a,\quad t\circ u\Rrightarrow a[s]$

1.	$a[s][t][u]$	$\to_{\text{Clos}}\ a[s\circ t][u]$
	i. (si $(s\circ t)\circ u\Rrightarrow a$)	$\to_{\text{Clos}}\ a[(s\circ t)\circ u]$
		$\to_{\text{Ass}}\ \boxed{a[s\circ(t\circ u)]}^{1}$
	ii. (si $(s\circ t)\circ u\not\Rrightarrow a$)	$\to_{\text{ClosGC}}\ \boxed{a}^{2}$
2.	$a[s][t][u]$	$\to_{\text{Clos}}\ a[s][t\circ u]$
	i. (si $s\circ(t\circ u)\Rrightarrow a$)	$\to_{\text{Clos}}\ \boxed{a[s\circ(t\circ u)]}^{1}$
	ii. (si $s\circ(t\circ u)\not\Rrightarrow a$)	$\to_{\text{ClosGC}}\ \boxed{a}^{2}$

Consideraciones:
Las derivaciones se corresponden como está indicado pues

$$((s\circ t)\circ u)\Rrightarrow a\iff(s\circ(t\circ u))\Rrightarrow a$$

Esto resulta como corolario del lema 63.

Clos + ClosGC

$s\Rrightarrow a,\quad t\Rrightarrow a[s],\quad u\Rrightarrow a[s][t],\quad s\circ t\Rrightarrow a,\quad t\circ u\not\Rrightarrow a[s]$

Notar que $\mathrm{FV}(a[s][t]) = \mathrm{FV}(a[s \circ t])$, con lo que $u \Rightarrow a[s \circ t]$.

$$\begin{array}{lll} 1. & a[s][t][u] & \to_{\mathrm{Clos}} \ a[s \circ t][u] \\ & & \to_{\mathrm{Clos}} \ a[(s \circ t) \circ u] \\ & & \to_{\mathrm{Ass}} \ a[s \circ (t \circ u)] \end{array}$$

$$2. \quad a[s][t][u] \quad \to_{\mathrm{ClosGC}} \ a[s]$$

En la primera derivación, para aplicar la regla *Clos* por segunda vez, es necesario mostrar que $(s \circ t) \circ u \Rightarrow a$. Como $s \Rightarrow a$ y $t \circ u \not\Rightarrow a[s]$, por lema 60.1 obtenemos $s \circ (t \circ u) \Rightarrow a$. Como $|(s \circ t) \circ u| = |s \circ (t \circ u)|$, entonces $(s \circ t) \circ u \Rightarrow a$.

El diagrama se cierra con el lema 78.

ClosGC + Clos

$s \Rightarrow a, \quad t \Rightarrow a[s], \quad u \Rightarrow a[s][t], \quad s \circ t \not\Rightarrow a, \quad t \circ u \Rightarrow a[s]$

$$\begin{array}{lll} 1. & a[s][t][u] & \to_{\mathrm{ClosGC}} \ a[u] \\ & i. \ (\text{si } u \not\Rightarrow a) & \to_{\mathrm{GC}} \ \boxed{a} \end{array}$$

$$\begin{array}{lll} 2. & a[s][t][u] & \to_{\mathrm{Clos}} \ a[s][t \circ u] \\ & i. \ (\text{si } s \circ (t \circ u) \not\Rightarrow a) & \to_{\mathrm{ClosGC}} \ \boxed{a} \\ & ii. \ (\text{si } s \circ (t \circ u) \Rightarrow a) & \to_{\mathrm{Clos}} \ a[s \circ (t \circ u)] \end{array}$$

Mostramos que $u \Rightarrow a \iff s \circ (t \circ u) \Rightarrow a$:

- Si $u \Rightarrow a$, como $s \circ t \not\Rightarrow a$, por lema 60.2 se tiene que $(s \circ t) \circ u \Rightarrow a$.
- Si $u \not\Rightarrow a$, como $s \circ t \not\Rightarrow a$, por lema 60.3 se tiene que $(s \circ t) \circ u \not\Rightarrow a$.

Luego, como $|(s \circ t) \circ u| = |s \circ (t \circ u)|$, entonces $(s \circ t) \circ u \Rightarrow a \iff s \circ (t \circ u) \Rightarrow a$.

Con esto queda demostrado la correspondencia entre las derivaciones *1.i* y *2.i*.

Nos queda mostrar que existe un reducto común entre las derivaciones *1* y *2.ii*. Tomamos una σ_{gc}-forma normal de cada término o sustitución que interviene:

$$\begin{array}{rcl} a' = {\sigma_{\mathrm{gc}}}'(a) & & \\ s' = {\sigma_{\mathrm{gc}}}'(s) & = & s_1 \cdot \ldots \cdot s_m \cdot \uparrow^n \\ t' = {\sigma_{\mathrm{gc}}}'(t) & = & t_1 \cdot \ldots \cdot t_p \cdot \uparrow^q \\ u' = {\sigma_{\mathrm{gc}}}'(u) & = & u_1 \cdot \ldots \cdot u_j \cdot \uparrow^k \end{array}$$

Con lo que

$$\begin{array}{lll} 1. & a[u] & \twoheadrightarrow a'[u'] \\ 2.ii & a[s \circ (t \circ u)] & \twoheadrightarrow a'[s' \circ (t' \circ u')] \end{array}$$

Queremos ver que

$$(\forall i \in \mathrm{FV}(a'))\ (\exists c_i)\ i[u'] \twoheadrightarrow c_i \twoheadleftarrow i[s' \circ (t' \circ u')]$$

para concluir con el lema 75 que

$$(\exists d)\ a'[u'] \twoheadrightarrow d \twoheadleftarrow a'[s' \circ (t' \circ u')]$$

Por lema 70, sabemos que

$$\begin{aligned} i[u'] &\twoheadrightarrow u_i \quad \text{si} \quad i \leq j \\ i[u'] &\twoheadrightarrow i + k - j \quad \text{si} \quad i > j \end{aligned}$$

Analizamos la relación entre las componentes del peso.

$$\begin{aligned} |s \circ t| &= (m + p - n, q) \quad \text{con } p \geq n \\ |s \circ t| &= (m, q + n - p) \quad \text{con } p < n \end{aligned}$$

Como $s \circ t \not\Vdash a$, pero $\mathrm{FV}(a) \neq \emptyset$, entonces debe suceder que

$$m + p - n = q \iff m = q + n - p \iff m - n = q - p$$

y $\forall i \in \mathrm{FV}(a)$

$$\begin{aligned} i &> m + p - n \quad \text{si} \quad p \geq n \\ i &> m \quad \text{si} \quad p < n \end{aligned}$$

Analizamos los distintos casos, según la relación entre los pesos de las sustituciones:

- $j \geq q$: $\quad s' \circ (t' \circ u') \twoheadrightarrow_{\mathrm{L}\ 61} s' \circ (t'_1 \cdot \ldots \cdot t'_p \cdot u_{q+1} \cdot \ldots \cdot u_j \cdot \uparrow^k)$
  - $p \geq n$: Notar que $n \leq p + j - q$

    $(s_1 \cdot \ldots \cdot s_m \cdot \uparrow^n) \circ (t'_1 \cdot \ldots \cdot t'_p \cdot u_{q+1} \cdot \ldots \cdot u_j \cdot \uparrow^k) \twoheadrightarrow_{\mathrm{L}\ 61} s'_1 \cdot \ldots \cdot s'_m \cdot t'_{n+1} \cdot \ldots \cdot t'_p \cdot u_{q+1} \cdot \ldots \cdot u_j \cdot \uparrow^k$

    Como $(\forall i \in \mathrm{FV}(a'))\ i > m + p - n = q$,

    $$\begin{aligned} i[s' \circ (t' \circ u')] &\twoheadrightarrow u_i \quad \text{si} \quad i \leq j \\ i[s' \circ (t' \circ u')] &\twoheadrightarrow i + k - j \quad \text{si} \quad i > j \end{aligned}$$

  - $j + p - q > n > p$

    $(s_1 \cdot \ldots \cdot s_m \cdot \uparrow^n) \circ (t'_1 \cdot \ldots \cdot t'_p \cdot u_{q+1} \cdot \ldots \cdot u_j \cdot \uparrow^k) \twoheadrightarrow_{\mathrm{L}\ 61} s'_1 \cdot \ldots \cdot s'_m \cdot u_{q+1+n-p} \cdot \ldots \cdot u_j \cdot \uparrow^k$

    Como $(\forall i \in \mathrm{FV}(a'))\ i > m = q + n - p$,

    $$\begin{aligned} i[s' \circ (t' \circ u')] &\twoheadrightarrow u_i \quad \text{si} \quad i \leq j \\ i[s' \circ (t' \circ u')] &\twoheadrightarrow i + k - j \quad \text{si} \quad i > j \end{aligned}$$

  - $n \geq j + p - q$

    $(s'_1 \cdot \ldots \cdot s'_m \cdot \uparrow^n) \circ (t'_1 \cdot \ldots \cdot t'_p \cdot u_{q+1} \cdot \ldots \cdot u_j \cdot \uparrow^k) \twoheadrightarrow_{\mathrm{L}\ 61} s'_1 \cdot \ldots \cdot s'_m \cdot \uparrow^{k+n-(j+p-q)}$

    Como $(\forall i \in \mathrm{FV}(a'))\ i > m = q + n - p$,

    $$i[s' \circ (t' \circ u')] \twoheadrightarrow i + (k + m - j) - m = i + k - j$$

    Notar que $q + n - p \geq j$, con lo que $i > j$.

- $j < q$: $s' \circ (t' \circ u') \twoheadrightarrow_{\mathrm{L}\ 61} s' \circ (t'_1 \cdot \ldots \cdot t'_p \cdot \uparrow^{k+q-j})$
  - $p \geq n$

    $(s'_1 \cdot \ldots \cdot s'_m \cdot \uparrow^n) \circ (t'_1 \cdot \ldots \cdot t'_p \cdot \uparrow^{k+q-j}) \twoheadrightarrow_{\mathrm{L}\ 61} s'_1 \cdot \ldots \cdot s'_m \cdot t'_{n+1} \cdot \ldots \cdot t'_p \cdot \uparrow^{k+q-j}$

    Como $(\forall i \in \mathrm{FV}(a'))\ i > m + p - n = q$,

    $$i[s' \circ (t' \circ u')] \twoheadrightarrow i + k + q - j - q = i + k - j$$

    Notar que $i > q > j$.
  - $p < n$

    $(s'_1 \cdot \ldots \cdot s'_m \cdot \uparrow^n) \circ (t'_1 \cdot \ldots \cdot t'_p \cdot \uparrow^{k+q-j}) \twoheadrightarrow s'_1 \cdot \ldots \cdot s'_m \cdot \uparrow^{k+q-j+n-p}$

    Como $(\forall i \in \mathrm{FV}(a'))\ i > m$, y $\ n - p - m = -q$,

    $$i[s' \circ (t' \circ u')] \twoheadrightarrow i + k + q - j + n - p - m = i + k - j$$

    Notar que $i > m = q + n - p > q > j$.

Hemos probado que

$$(\forall i \in \mathrm{FV}(a'))(\exists c_i)\ i[u'] \twoheadrightarrow c_i \twoheadleftarrow i[s' \circ (t' \circ u')]$$

Por lema 75, entonces

$$(\exists d)\ a'[u'] \twoheadrightarrow d \twoheadleftarrow a'[s' \circ (t' \circ u')]$$

Luego ambos términos convergen.

ClosGC + ClosGC

$s \Rrightarrow a$, $\quad t \Rrightarrow a[s]$, $\quad u \Rrightarrow a[s][t]$, $\quad s \circ t \not\Rrightarrow a$, $\quad t \circ u \not\Rrightarrow a[s]$
Notar que $\mathrm{FV}(a) \neq \emptyset$

1. $a[s][t][u] \quad \rightarrow_{\mathrm{ClosGC}} \ a[u]$

2. $a[s][t][u] \quad \rightarrow_{\mathrm{ClosGC}} \ a[s]$

Sean

$$\begin{aligned} |s| &= (m, n) \\ |t| &= (p, q) \\ |u| &= (j, k) \end{aligned}$$

Analizamos los pesos de $s \circ t$ y $t \circ u$ para ver las condiciones en las que esto sucede:

$$\begin{aligned} |s \circ t| &= (m + p - n, q) \quad \text{con } p \geq n \\ |s \circ t| &= (m, q + n - p) \quad \text{con } p < n \end{aligned}$$

En cualquier caso tenemos:

$$s \circ t \Vdash a \implies (\forall i \in \mathrm{FV}(a))\ i > m \ \wedge\ m - n = q - p \tag{5.1}$$

$$\begin{array}{lll} |t \circ u| & = (p + j - q, k) & \text{con } j \geq q \\ |t \circ u| & = (p, k + q - j) & \text{con } j < q \end{array}$$

Como $t \circ u \Vdash a[s]$, tenemos

$$p - q = k - j \tag{5.2}$$

y

$$\begin{array}{lll} (\forall i \in \mathrm{FV}(a[s])) & i > p + j - q & \text{con } j \geq q \\ (\forall i \in \mathrm{FV}(a[s])) & i > p & \text{con } j < q \end{array}$$

Por definición de FV:

$$\begin{array}{lll} (\forall i \in \mathrm{FV}(a)_{>m}) & i + n - m > p + j - q & \text{con } j \geq q \\ (\forall i \in \mathrm{FV}(a)_{>m}) & i + n - m > p & \text{con } j < q \end{array}$$

Entonces, por 5.1,

$$\begin{array}{lll} (\forall i \in \mathrm{FV}(a)) & i + n - m > p + j - q & \text{con } j \geq q \qquad (5.3) \\ (\forall i \in \mathrm{FV}(a)) & i + n - m > p & \text{con } j < q \qquad (5.4) \end{array}$$

De 5.1 y 5.2 se deduce que

$$m - n = j - k \tag{5.5}$$

Para ver que existe un reducto común, vamos a reducir a una forma normal cada término o sustitución que interviene. Sean

$$\begin{array}{lll} a' & = & {\sigma_{\mathrm{gc}}}'(a) \\ s' & = & {\sigma_{\mathrm{gc}}}'(s) \\ u' & = & {\sigma_{\mathrm{gc}}}'(u) \end{array}$$

Luego

$$\begin{array}{lll} a[s] & \twoheadrightarrow & a'[s'] \\ a[u] & \twoheadrightarrow & a'[u'] \end{array}$$

Queremos ver que

$$(\forall i \in \mathrm{FV}(a'))\ (\exists c_i)\ i[s'] \twoheadrightarrow c_i \twoheadleftarrow i[u']$$

para concluir con el lema 75 que

$$(\exists d)\ a'[s'] \twoheadrightarrow d \twoheadleftarrow a'[u']$$

De 5.1 y el lema 70, se deduce que

$$(\forall i \in \mathrm{FV}(a'))\ i[s'] \twoheadrightarrow i + n - m$$

Si mostramos que

$$(\forall i \in \mathrm{FV}(a'))\ i > j$$

entonces tendríamos que, por lema 70 y ecuación 5.5

$$(\forall i \in \mathrm{FV}(a'))\ i[u'] \twoheadrightarrow i + k - j = i + n - m$$

Separamos en los dos casos del peso de $t \circ u$:

- $j \geq q$: por 5.1 y 5.3,

$$i + n - m > p + j - q \quad \Longleftrightarrow \quad i + p - q > j + p - q \quad \Longrightarrow \quad i > j$$

- $j < q$: por 5.1 y 5.4,

$$i + n - m > p = q - m + n \quad \Longrightarrow \quad i > q \quad \Longrightarrow \quad i > j$$

Hemos probado que

$$(\forall i \in \mathrm{FV}(a'))(\exists c_i)\ i[u'] \twoheadrightarrow c_i \twoheadleftarrow i[s']$$

Por lema 75, entonces

$$(\exists d)\ a'[u'] \twoheadrightarrow d \twoheadleftarrow a'[s']$$

Luego ambos términos convergen.

IdL + Ass

1. $(\mathbf{id} \circ s) \circ t \quad \to_{\mathrm{IdL}} \boxed{s \circ t}$

2. $(\mathbf{id} \circ s) \circ t \quad \to_{\mathrm{Ass}} \mathbf{id} \circ (s \circ t)$
   $\to_{\mathrm{IdL}} \boxed{s \circ t}$

ShiftId + Ass

1. $(\uparrow \circ \mathbf{id}) \circ t \quad \to_{\mathrm{ShiftId}} \boxed{\uparrow \circ t}$

2. $(\uparrow \circ \mathbf{id}) \circ t \quad \to_{\mathrm{Ass}} \uparrow \circ (\mathbf{id} \circ t)$
   $\to_{\mathrm{IdL}} \boxed{\uparrow \circ t}$

ShiftCons + Ass

1. $(\uparrow \circ (a \cdot s)) \circ t \quad \to_{\text{ShiftCons}} \boxed{s \circ t}$

2. $(\uparrow \circ (a \cdot s)) \circ t \quad \to_{\text{Ass}} \uparrow \circ ((a \cdot s) \circ t)$
$\to_{\text{Map o MapGC}} \uparrow \circ (a' \cdot (s \circ t))$
$\to_{\text{ShiftCons}} \boxed{s \circ t}$

Map + Ass

$t \rhd a$

1. $((a \cdot s) \circ t) \circ u \quad \to_{\text{Map}} (a[t] \cdot (s \circ t)) \circ u$
i. (si $u \rhd a[t]$) $\to_{\text{Map}} a[t][u] \cdot ((s \circ t) \circ u)$
i.1. (si $t \circ u \rhd a$) $\to_{\text{Clos + Ass}} \boxed{a[t \circ u] \cdot (s \circ (t \circ u))}^{1}$
i.2. (si $t \circ u \not\rhd a$) $\to_{\text{ClosGC + Ass}} \boxed{a \cdot (s \circ (t \circ u))}^{2}$
ii. (si $u \not\rhd a[t]$) $\to_{\text{MapGC}} a[t] \cdot ((s \circ t) \circ u)$
$\to_{\text{Ass}} a[t] \cdot (s \circ (t \circ u))$

2. $((a \cdot s) \circ t) \circ u \quad \to_{\text{Ass}} (a \cdot s) \circ (t \circ u)$
i. (si $t \circ u \rhd a$) $\to_{\text{Map}} \boxed{a[t \circ u] \cdot (s \circ (t \circ u))}^{1}$
ii. (si $t \circ u \not\rhd a$) $\to_{\text{MapGC}} \boxed{a \cdot (s \circ (t \circ u))}^{2}$

Si $u \rhd a[t]$, entonces resulta evidente que 1.*i*.1 $\iff$ 2.*i* y 1.*i*.2 $\iff$ 2.*ii*
Si $u \not\rhd a[t]$, entonces no se puede dar el caso *2.ii*, pues si $u \not\rhd a[t]$ y $t \rhd a$, por lema 60.1, $t \circ u \rhd a$. Luego 1.*ii* $\iff$ 2.*i*. Un reducto común entre estas derivaciones se encuentra gracias al lema 78.

MapGC + Ass

$t \not\rhd a$

$$
\begin{array}{lll}
1. & ((a \cdot s) \circ t) \circ u & \to_{\text{MapGC}} \ (a \cdot (s \circ t)) \circ u \\
& i.\ (\text{si } u \rhd a) & \to_{\text{Map}} \ a[u] \cdot ((s \circ t) \circ u) \\
& & \to_{\text{Ass}} \ a[u] \cdot (s \circ (t \circ u)) \\
& & \\
& ii.\ (\text{si } u \not\rhd a) & \to_{\text{MapGC}} \ a \cdot ((s \circ t) \circ u) \\
& & \to_{\text{Ass}} \ \boxed{a \cdot (s \circ (t \circ u))} \\
& & \\
2. & ((a \cdot s) \circ t) \circ u & \to_{\text{Ass}} \ (a \cdot s) \circ (t \circ u) \\
& i.\ (\text{si } t \circ u \rhd a) & \to_{\text{Map}} \ a[t \circ u] \cdot (s \circ (t \circ u)) \\
& & \\
& ii.\ (\text{si } t \circ u \not\rhd a) & \to_{\text{MapGC}} \ \boxed{a \cdot (s \circ (t \circ u))}
\end{array}
$$

El paso *1.i* cierra con el *2.i* con el lema 80. Vemos que $1.i \iff 2.i$ y $1.ii \iff 2.ii$:

- $1.i \implies 2.i$. Como $t \not\rhd a$ y $u \rhd a$, por lema 60.2, $t \circ u \rhd a$.
- $1.ii \implies 2.ii$. Como $t \not\rhd a$ y $u \not\rhd a$, por lema 60.3, $t \circ u \not\rhd a$.
- $2.i \implies 1.i$. Como $t \not\rhd a$ y $t \circ u \rhd a$, por lema 60.9, $u \rhd a$.
- $2.ii \implies 1.ii$. Como $t \not\rhd a$ y $t \circ u \not\rhd a$, por lema 60.7, $u \not\rhd a$.

Ass + Ass

$$
\begin{array}{lll}
1. & ((s \circ t) \circ u) \circ w & \to_{\text{Ass}} \ (s \circ (t \circ u)) \circ w \\
& & \to_{\text{Ass}} \ \boxed{s \circ (t \circ (u \circ w))} \\
& & \\
2. & ((s \circ t) \circ u) \circ w & \to_{\text{Ass}} \ (s \circ t) \circ (u \circ w) \\
& & \to_{\text{Ass}} \ \boxed{s \circ (t \circ (u \circ w))}
\end{array}
$$

Teorema 84. σ_{gc} es WCR

Demostración. Hemos dedicado este capítulo a verificar que todos los pares críticos se cierran, y se conlcuye por lema 11. □

5.2 Confluencia de σ_{gc}

Estamos en condiciones de presentar el resultado principal del capítulo:

Teorema 85. [Confluencia σ_{gc}] σ_{gc} es CR

Demostración. Por teorema 84, σ_{gc} es WCR. Por teorema 57 vimos que σ_{gc} es SN. Luego, por Newman (lema 5), σ_{gc} es CR. □

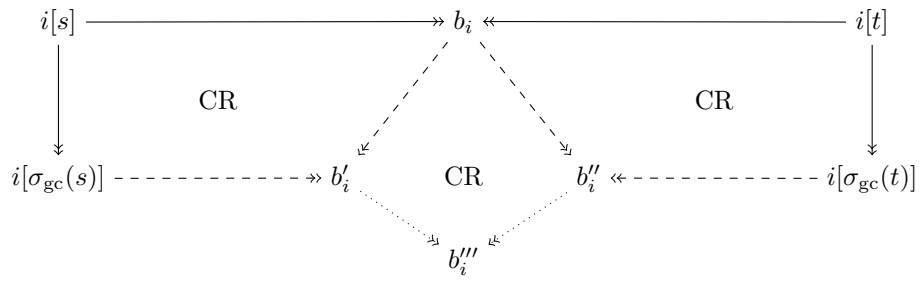
Notación. $\sigma_{\text{gc}}(t)$ denota la única σ_{gc}-f. n. de t.

Sabemos que dicha forma normal es única por CR y SN de σ_{gc} (lema 6).

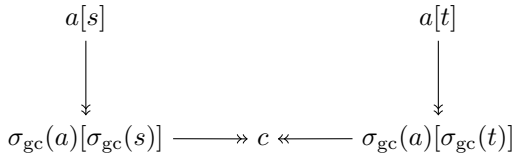
Lema 86. Sean $a \in \Lambda\sigma^t_{\text{gc}}$ y $s, t \in \Lambda\sigma^s_{\text{gc}}$

$$(\forall i \in \text{FV}(a))(\exists b_i)\ i[s] \twoheadrightarrow b_i \twoheadleftarrow i[t] \implies (\exists c)\ a[s] \twoheadrightarrow c \twoheadleftarrow a[t]$$

Demostración. Por confluencia de σ_{gc}, para cada $i \in \text{FV}(\sigma_{\text{gc}}(a)) \subseteq \text{FV}(a)$ se cumple



Luego, por lema 75,



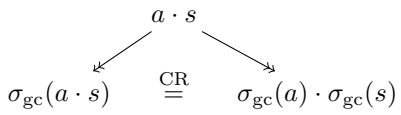
□

Lema 87. Sean $a, b \in \Lambda\sigma^t_{\text{gc}}$, $s, t \in \Lambda\sigma^s_{\text{gc}}$, entonces

1. $\sigma_{\text{gc}}(a[s]) = \sigma_{\text{gc}}(\sigma_{\text{gc}}(a)[\sigma_{\text{gc}}(s)])$
2. $\sigma_{\text{gc}}(a \cdot s) = \sigma_{\text{gc}}(a) \cdot \sigma_{\text{gc}}(s)$
3. $\sigma_{\text{gc}}(a\ b) = \sigma_{\text{gc}}(a)\ \sigma_{\text{gc}}(b)$
4. $\sigma_{\text{gc}}(\lambda a) = \lambda\sigma_{\text{gc}}(a)$
5. $\sigma_{\text{gc}}(s \circ t) = \sigma_{\text{gc}}(\sigma_{\text{gc}}(s) \circ \sigma_{\text{gc}}(t))$

Demostración.

1. Trivial por CR de σ_{gc}
2. No hay reglas que se puedan aplicar en la raíz del término, luego



3. Se sigue el mismo razonamiento de la demostración anterior.
4. Se sigue el mismo razonamiento de la demostración anterior.
5. Trivial por CR de σ_{gc}

□

Capítulo 6

Confluencia de $\lambda\sigma_{\mathbf{gc}}$

En el presente capítulo vamos a mostrar la confluencia de $\lambda\sigma_{\mathrm{gc}}$ por medio del lema de interpretación (lema 7). Utilizaremos el resultado del lema 62, y la convención $(\forall i \in \mathbb{N}_{>1})\ i = 1[\uparrow^{i-1}]$. Notar que, con dicha convención, las σ_{gc}-f. n. son los términos Λ_{DB}.

Para poder cumplir con las hipótesis del lema de interpretación, demostraremos que el cálculo tiene las propiedades *Simulación* (sección 6.1) y *Corrección* (sección 6.2).

Sea $\to_{\beta'}$ la menor reducción compatible generada por

$$(\lambda a)\ b \quad \to_{\beta'} \quad a\{1 \leftarrow b\}$$

Sobre $(\Lambda\sigma^t_{\mathrm{gc}})_{\mathrm{nf}}$, $\to_{\beta'}$ coincide con $\to_{\beta}$ sobre Λ_{DB}.

Sobre $(\Lambda\sigma^s_{\mathrm{gc}})_{\mathrm{nf}}$, $a_1 \cdot \ldots \cdot a_i \cdot \ldots \cdot a_n \cdot \uparrow^n \to_{\beta'} t$ ssi $t = a_1 \cdot \ldots \cdot a'_i \cdot \ldots \cdot a_n \cdot \uparrow^n$ con $a_i \to_{\beta} a'_i$

Lema 88. $\to_{\beta'}$ es confluente sobre $(\Lambda\sigma_{\mathrm{gc}})_{\mathrm{nf}}$.

Demostración. Vemos que $\forall w \in (\Lambda\sigma_{\mathrm{gc}})_{\mathrm{nf}}\ \ w' \twoheadleftarrow w \twoheadrightarrow w'' \quad \exists w'''\ \ w' \twoheadrightarrow w''' \twoheadleftarrow w''$

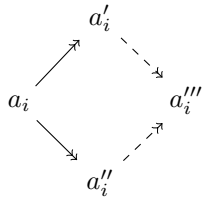
1. $w \in (\Lambda\sigma^t_{\mathrm{gc}})_{\mathrm{nf}}$: como β' coincide con β y β es confluente, es inmediato.
2. $w \in (\Lambda\sigma^s_{\mathrm{gc}})_{\mathrm{nf}}$: $w = a_1 \cdot \ldots \cdot a_n \cdot \uparrow^n$

   $$w \twoheadrightarrow w' \implies w' = a'_1 \cdot \ldots \cdot a'_n \cdot \uparrow^n \text{ con } a_i \twoheadrightarrow a'_i$$

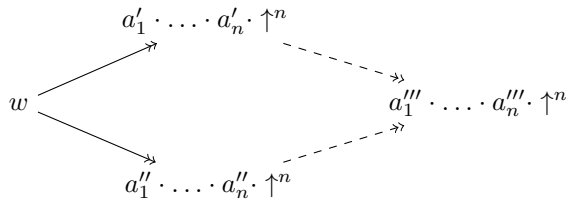
   y

   $$w \twoheadrightarrow w'' \implies w'' = a''_1 \cdot \ldots \cdot a''_n \cdot \uparrow^n \text{ con } a_i \twoheadrightarrow a''_i$$

   Por el punto anterior, para los términos a_i vale la confluencia. Es decir,



Luego,



□

6.1 Simulación

En la presente sección mostraremos que $\lambda\sigma_{\text{gc}}$ simula λ_{DB}, es decir

$$w, v \in (\Lambda\sigma_{\text{gc}})_{\text{nf}} \;\; w \to_{\beta'} v \implies w \twoheadrightarrow_{\lambda\sigma_{\text{gc}}} v$$

Notación. Salvo indicación contraria, utilizaremos $\to$ y $\twoheadrightarrow$ para indicar $\to_{\sigma_{\text{gc}}}$ y $\twoheadrightarrow_{\sigma_{\text{gc}}}$, respectivamente.

Definición 89. Sean $i \geq 0, n \geq 1$, se define $s_{i\,n}$ como

$$s_{i\,n} = 1 \cdot 2 \cdot \ldots \cdot i \cdot \uparrow^{i+n-1} \qquad s_{0\,n} = \uparrow^{n-1} \qquad s_{0\,1} = \mathbf{id}$$

Notar que $|s_{i\,n}| = (i, i+n-1)$

Observación 90.

1. $n[\uparrow] = 1[\uparrow^{n-1}][\uparrow] \to 1[\uparrow^{n-1} \circ \uparrow] \twoheadrightarrow 1[\uparrow^n] = n+1$
2. $$\begin{aligned} 1 \cdot (s_{i\,n} \circ \uparrow) &= 1 \cdot ((1 \cdot \ldots \cdot i \cdot \uparrow^{i+n-1}) \circ \uparrow) \\ &\twoheadrightarrow 1 \cdot ((1[\uparrow] \cdot \ldots \cdot i[\uparrow] \cdot (\uparrow^{i+n-1} \circ \uparrow))) \\ &\twoheadrightarrow 1 \cdot 2 \cdot \ldots \cdot i+1 \cdot \uparrow^{i+n} \\ &= s_{i+1\,n} \end{aligned}$$

Lema 91. Sea $j \in \mathbb{N}$, entonces $\mathrm{U}^1_j(a) = a$

Demostración. Por inducción en el término, como se menciona en [Rio93]. □

Lema 92. Sean $n, j \in \mathbb{N}$, entonces

$$(\forall i \in \mathrm{FV}(a))\; i \leq j \implies \mathrm{U}^n_j(a) = a$$

Demostración. Por inducción en el término.

- $a = m$: Por hipótesis, $m \leq j$. $\mathrm{U}^n_j(m) = m$
- $a = c\ d$: $\mathrm{U}^n_j(b\ c) = \mathrm{U}^n_j(b)\ \mathrm{U}^n_j(c) =_{\text{h. i.}} b\ c$

- $a = \lambda b$: $\mathrm{U}_j^n(\lambda b) = \lambda \mathrm{U}_{j+1}^n(b) =_{\text{h. i.}} \lambda b$
  La h. i. se puede aplicar pues

$$(\forall i \in \mathrm{FV}(\lambda b))\ i \leq j \implies (\forall i \in \mathrm{FV}(b))\ i \leq j+1$$

□

Lema 93. Sean $i, n \in \mathbb{N}$, $a \in \Lambda_{\mathrm{DB}}$, entonces

$$s_{i\,n} \not\Subset a \implies \mathrm{U}_i^n(a) = a$$

Demostración. Por definición de $\Subset$ y de $s_{i\,n}$, puede ocurrir:

1. $\mathrm{FV}(a) = \emptyset$: Por lema 92, $\mathrm{U}_i^n(a) = a$.
2. $n = 1$: Por lema 91, $\mathrm{U}_i^1(a) = a$.

□

Lema 94. Sean $b \in \Lambda_{\mathrm{DB}}$, $i \geq 0$, $n \geq 1$, entonces

$$b[s_{i\,n}] \twoheadrightarrow_{\sigma_{\mathrm{gc}}} \mathrm{U}_i^n(b)$$

Demostración. Por inducción en b:

- $b = 1$
  - $i = 0$
    - $n = 1 \quad 1[s_{0\,1}] = 1[\mathbf{id}] \to_{\mathrm{GC}} 1 = \mathrm{U}_0^1(1)$
    - $n > 1 \quad 1[s_{0\,n}] = 1[\uparrow^{n-1}] = n = \mathrm{U}_0^n(1)$
  - $i > 0$

    $1[s_{i\,n}] \to_{\mathrm{VarCons}} 1 = \mathrm{U}_i^n(1)$
- $b = c\ d$
  - $(c\ d)[s_{i\,n}] \to_{\mathrm{App}_1} c[s_{i\,n}]\ d[s_{i\,n}] \twoheadrightarrow_{\text{h. i.}} \mathrm{U}_i^n(c)\ \mathrm{U}_i^n(d) = \mathrm{U}_i^n(c\ d)$
  - $(c\ d)[s_{i\,n}] \to_{\mathrm{App}_2} c[s_{i\,n}]\ d \twoheadrightarrow_{\text{h. i.}} \mathrm{U}_i^n(c)\ d =_{\mathrm{L}\ 93} \mathrm{U}_i^n(c)\ \mathrm{U}_i^n(d) = \mathrm{U}_i^n(c\ d)$
  - $(c\ d)[s_{i\,n}] \to_{\mathrm{App}_3} c\ d[s_{i\,n}] \twoheadrightarrow_{\text{h. i.}} c\ \mathrm{U}_i^n(d) =_{\mathrm{L}\ 93} \mathrm{U}_i^n(c)\ \mathrm{U}_i^n(d) = \mathrm{U}_i^n(c\ d)$
  - $(c\ d)[s_{i\,n}] \to_{\mathrm{GC}} c\ d =_{\mathrm{L}\ 93} \mathrm{U}_i^n(c\ d)$
- $b = \lambda c$
  - $(\lambda c)[s_{i\,n}] \to_{\mathrm{Abs}} \lambda c[1 \cdot (s_{i\,n} \circ \uparrow)] \twoheadrightarrow_{\mathrm{Obs}\ 90} \lambda c[s_{i+1\,n}] \twoheadrightarrow_{\text{h. i.}} \lambda \mathrm{U}_{i+1}^n(c) = \mathrm{U}_i^n(\lambda c)$
  - $(\lambda c)[s_{i\,n}] \to_{\mathrm{GC}} \lambda c =_{\mathrm{L}\ 93} \mathrm{U}_i^n(\lambda c)$
- $b = k > 1$
  - $s_{i\,n} \Subset k \quad k[s_{i\,n}] = 1[\uparrow^{k-1}][s_{i\,n}] \to_{\mathrm{Clos}} 1[\uparrow^{k-1} \circ s_{i\,n}]$

$$\uparrow^{k-1} \circ 1 \cdot 2 \cdot \ldots \cdot i \cdot \uparrow^{n+i-1} \twoheadrightarrow \begin{cases} k \cdot \ldots \cdot i \cdot \uparrow^{n+i-1} & k-1 < i \\ \uparrow^{n+k-2} & k-1 \geq i \end{cases}$$

$$\therefore \quad 1[\uparrow^{k-1} \circ s_{i\,n}] \twoheadrightarrow \left\{ \begin{array}{ll} k & k-1 < i \\ n+k-1 & k-1 \geq i \end{array} \right\} = \mathrm{U}_i^n(k)$$

– $s_{i\,n} \not\Vdash k \quad k[s_{i\,n}] \ \to_{\mathrm{GC}} \ k =_{\mathrm{L\ 93}} \mathrm{U}_i^n(k)$

□

Definición 95. Para $b \in \Lambda\sigma^t_{\mathrm{gc}}$ se define $b_i \in \Lambda\sigma^s_{\mathrm{gc}}$ /

$$\begin{array}{llll} \mathrm{FV}(b) \neq \emptyset & b_1 = b \cdot \mathbf{id} & b_2 = 1 \cdot b[\uparrow] \cdot \uparrow & b_{k+1} = 1 \cdot \ldots \cdot k \cdot b[\uparrow^k] \cdot \uparrow^k \\ \mathrm{FV}(b) = \emptyset & b_1 = b \cdot \mathbf{id} & b_2 = 1 \cdot b \cdot \uparrow & b_{k+1} = 1 \cdot \ldots \cdot k \cdot b \cdot \uparrow^k \end{array}$$

Notar que $|b_i| = (i, i-1)$

Observación 96. $1 \cdot (b_k \circ \uparrow) \twoheadrightarrow b_{k+1}$

Demostración.

1. $\mathrm{FV}(b) \neq \emptyset$
   - $k = 1$
     $$1 \cdot (b_1 \circ \uparrow) = 1 \cdot ((b \cdot \mathbf{id}) \circ \uparrow) \twoheadrightarrow 1 \cdot (b[\uparrow] \cdot \uparrow) = b_2$$
   - $k > 1$
     $$\begin{array}{rcl} 1 \cdot (b_k \circ \uparrow) & = & 1 \cdot ((1 \cdot \ldots \cdot k-1 \cdot b[\uparrow^{k-1}] \cdot \uparrow^{k-1}) \circ \uparrow) \\ & \twoheadrightarrow & 1 \cdot 2 \cdot \ldots \cdot k \cdot b[\uparrow^k] \cdot \uparrow^k = b_{k+1} \end{array}$$
2. $\mathrm{FV}(b) = \emptyset$
   - $k = 1$
     $$1 \cdot (b_1 \circ \uparrow) = 1 \cdot ((b \cdot \mathbf{id}) \circ \uparrow) \twoheadrightarrow 1 \cdot (b \cdot \uparrow) = b_2$$
   - $k > 1$
     $$\begin{array}{rcl} 1 \cdot (b_k \circ \uparrow) & = & 1 \cdot ((1 \cdot \ldots \cdot k-1 \cdot b \cdot \uparrow^{k-1}) \circ \uparrow) \\ & \twoheadrightarrow & 1 \cdot 2 \cdot \ldots \cdot k \cdot b \cdot \uparrow^k = b_{k+1} \end{array}$$

□

Lema 97. Sean $k > 0$, $a, b \in \Lambda_{\mathrm{DB}}$, entonces

$$(\forall i \in \mathrm{FV}(a))\ i < k \implies a\{k \leftarrow b\} = a$$

Demostración. Por inducción en a.

- $i\{k \leftarrow b\} = i$
- $(c\ d)\{k \leftarrow b\} = c\{k \leftarrow b\}\ d\{k \leftarrow b\} =_{\mathrm{h.\ i.}} c\ d$
- $(\lambda c)\{k \leftarrow b\} = \lambda c\{k+1 \leftarrow b\} =_{\mathrm{h.\ i.}} \lambda c$

□

Lema 98. Sean $k > 0$, $a, b \in \Lambda_{\mathrm{DB}}$, entonces

$$b_k \not\Vdash a \implies a\{k \leftarrow b\} = a$$

Demostración. Por definición de b_k y $\rhd$, $\mathrm{FV}(a) = \emptyset$. Luego, por lema 97, $a\{k \leftarrow b\} = a$ □

Lema 99. Sean $a, b \in \Lambda_{\mathrm{DB}}$, $k \geq 1$, entonces

$$a[b_k] \twoheadrightarrow a\{k \leftarrow b\}$$

Demostración. Inducción en a

- $a = 1$
  - $k = 1$
    $$1[b_1] = 1[b \cdot \mathbf{id}] \to b =_{\mathrm{L}\ 91} \mathrm{U}_0^1(b) = 1\{1 \leftarrow b\}$$
  - $k > 1$
    $$1[b_k] \to 1 = 1\{k \leftarrow b\}$$
- $a = c\ d$
  - $(c\ d)[b_k] \ \to_{\mathrm{App}_1} \ c[b_k]\ d[b_k] \twoheadrightarrow_{\text{h. i.}} \ c\{k \leftarrow b\}\ d\{k \leftarrow b\} = (c\ d)\{k \leftarrow b\}$
  - $(c\ d)[b_k] \ \to_{\mathrm{App}_2} \ c[b_k]\ d \twoheadrightarrow_{\text{h. i.}} \ c\{k \leftarrow b\}\ d =_{\mathrm{L}\ 98} \ c\{k \leftarrow b\}\ d\{k \leftarrow b\} = (c\ d)\{k \leftarrow b\}$
  - $(c\ d)[b_k] \ \to_{\mathrm{App}_3} \ c\ d[b_k] \twoheadrightarrow_{\text{h. i.}} \ c\ d\{k \leftarrow b\} =_{\mathrm{L}\ 98} \ c\{k \leftarrow b\}\ d\{k \leftarrow b\} = (c\ d)\{k \leftarrow b\}$
  - $(c\ d)[b_k] \ \to_{\mathrm{GC}} \ c\ d =_{\mathrm{L}\ 98} \ (c\ d)\{k \leftarrow b\}$
- $a = \lambda c$
  - $(\lambda c)[b_k] \ \to_{\mathrm{Abs}} \ \lambda c[1 \cdot (b_k \circ \uparrow)] \twoheadrightarrow_{\mathrm{Obs}\ 96} \lambda c[b_{k+1}] \twoheadrightarrow_{\text{h. i.}} \lambda c\{k+1 \leftarrow b\} = (\lambda c)\{k \leftarrow b\}$
  - $(\lambda c)[b_k] \ \to_{\mathrm{GC}} \ \lambda c =_{\mathrm{L}\ 98} \ (\lambda c)\{k \leftarrow b\}$
- $a = m > 1 \quad m[b_k] = 1[\uparrow^{m-1}][b_k] \to 1[\uparrow^{m-1} \circ b_k]$
  - $k = 1$
    $$1[\uparrow^{m-1} \circ (b \cdot \mathbf{id})] \twoheadrightarrow 1[\uparrow^{m-2}] = m - 1 = m\{1 \leftarrow b\}$$
  - $k > 1$
    $$\uparrow^{m-1} \circ (1 \cdot \ldots \cdot k-1 \cdot b' \cdot \uparrow^{k-1}) \twoheadrightarrow \begin{cases} m \cdot \ldots \cdot k-1 \cdot b' \cdot \uparrow^{k-1} & m < k \\ b' \cdot \uparrow^{k-1} & m = k \\ \uparrow^{m-2} & m > k \end{cases}$$
    $$1[\uparrow^{m-1} \circ b_k] \twoheadrightarrow \begin{cases} m & m < k \\ b' & m = k \\ m-1 & m > k \end{cases} \twoheadrightarrow_{\mathrm{L}\ 94\ \mathrm{o}\ 92} \begin{cases} m & m < k \\ \mathrm{U}_0^k(b) & m = k \\ m-1 & m > k \end{cases} = m\{k \leftarrow b\}$$

  Con $b' = b$ si $\mathrm{FV}(b) = \emptyset$ o $b' = b[\uparrow^{k-1}]$ si no.

□

Corolario 100. Sean $a, b \in \Lambda_{\mathrm{DB}}$, $k \geq 1$, entonces $\sigma_{\mathrm{gc}}(a[b \cdot \mathbf{id}]) = a\{1 \leftarrow b\}$

Demostración. Por CR de σ_{gc} y por estar $a\{1 \leftarrow b\}$ en σ_{gc}-f.n. □

Teorema 101. [Simulación] Sean $w, v \in (\Lambda\sigma_{\mathrm{gc}})_{\mathrm{nf}}$, entonces

$$w \to_{\beta'} v \implies w \twoheadrightarrow_{\lambda\sigma_{\mathrm{gc}}} v$$

En particular, $\quad a, b \in \Lambda_{\mathrm{DB}} \;\; a \to_{\beta} b \implies a \twoheadrightarrow_{\lambda\sigma_{\mathrm{gc}}} b$

Demostración. Por inducción en w

- $w = 1, n, \mathbf{id}, \uparrow^n$: se satisface vacuamente.
- $w = a\ b$
  - $v = a'\ b \quad$ con $a \to_{\beta'} a'$

    $$a \overset{\text{h. i.}}{\twoheadrightarrow}_{\lambda\sigma_{\mathrm{gc}}} a' \implies a\ b \twoheadrightarrow_{\lambda\sigma_{\mathrm{gc}}} a'\ b$$

  - $v = a\ b' \quad$ con $b \to_{\beta'} b'$

    $$b \overset{\text{h. i.}}{\twoheadrightarrow}_{\lambda\sigma_{\mathrm{gc}}} b' \implies a\ b \twoheadrightarrow_{\lambda\sigma_{\mathrm{gc}}} a\ b'$$

  - $a = \lambda c \quad v = c\{1 \leftarrow b\}$

    $$(\lambda c)\ b \;\to_{\mathrm{Beta}}\; c[b \cdot \mathbf{id}] \twoheadrightarrow_{\sigma_{\mathrm{gc}}} \sigma_{\mathrm{gc}}(c[b \cdot \mathbf{id}]) =_{\text{Coro 100}} c\{1 \leftarrow b\}$$

- $w = \lambda a \quad v = \lambda a' \quad$ con $a \to_{\beta'} a'$

  $$a \overset{\text{h. i.}}{\twoheadrightarrow}_{\lambda\sigma_{\mathrm{gc}}} a' \implies \lambda a \twoheadrightarrow_{\lambda\sigma_{\mathrm{gc}}} \lambda a'$$

- $w = a \cdot s$
  - $v = a' \cdot s \quad$ con $a \to_{\beta'} a'$

    $$a \overset{\text{h. i.}}{\twoheadrightarrow}_{\lambda\sigma_{\mathrm{gc}}} a' \implies a \cdot s \twoheadrightarrow_{\lambda\sigma_{\mathrm{gc}}} a' \cdot s$$

  - $v = a \cdot s' \quad$ con $s \to_{\beta'} s'$

    $$s \overset{\text{h. i.}}{\twoheadrightarrow}_{\lambda\sigma_{\mathrm{gc}}} s' \implies a \cdot s \twoheadrightarrow_{\lambda\sigma_{\mathrm{gc}}} a \cdot s'$$

□

6.2 Corrección y confluencia de $\lambda\sigma_{\mathbf{gc}}$

En la presente sección mostraremos que el cálculo $\lambda\sigma_{\mathrm{gc}}$ cumple la propiedad de Corrección (también conocida como *Soundness*), es decir, para $w, v \in (\Lambda\sigma_{\mathrm{gc}})_{\mathrm{nf}}$, entonces

$$w \twoheadrightarrow_{\lambda\sigma_{\mathrm{gc}}} v \implies w \twoheadrightarrow_{\beta'} v$$

y concluiremos probando que se cumple la propiedad de confluencia para $\lambda\sigma_{\mathrm{gc}}$, mediante el lema de interpretación.

Lema 102. Sean $a \in \Lambda\sigma^t_{\mathrm{gc}}$, $s \in \Lambda\sigma^s_{\mathrm{gc}}$, entonces

$$\sigma_{\mathrm{gc}}((\lambda a)[s]) = \sigma_{\mathrm{gc}}(\lambda a[1 \cdot (s \circ \uparrow)])$$

Demostración.

- $s \rhd (\lambda a)$: es trivial.
- $s \not\rhd (\lambda a)$: Por lema 82 sabemos que existe c tal que

  $$(\lambda a)[s] \;\twoheadrightarrow\; c \;\twoheadleftarrow\; \lambda a[1 \cdot (s \circ \uparrow)]$$

  Entonces, por CR de σ_{gc}, debe ocurrir que su forma normal sea la misma.

□

Lema 103. Sean $a, b \in \Lambda\sigma^t_{\mathrm{gc}}$, $s \in \Lambda\sigma^s_{\mathrm{gc}}$, entonces

$$\sigma_{\mathrm{gc}}((a\ b)[s]) = \sigma_{\mathrm{gc}}(a[s])\ \sigma_{\mathrm{gc}}(b[s])$$

Demostración. Por CR de σ_{gc} y aplicación de *GC* o algún *App*. □

Lema 104. Sean $a, b \in \Lambda\sigma^t_{\mathrm{gc}}$, $s \in \Lambda\sigma^s_{\mathrm{gc}}$, entonces

$$\sigma_{\mathrm{gc}}(a[1 \cdot (s \circ \uparrow)][b \cdot \mathbf{id}]) = \sigma_{\mathrm{gc}}(a[b \cdot s])$$

Demostración. Separamos en casos:

- $1 \cdot (s \circ \uparrow) \rhd a$: Notar que $1 \in \mathrm{FV}(a[1 \cdot (s \circ \uparrow)])$, entonces $b \cdot \mathbf{id} \rhd a[1 \cdot (s \circ \uparrow)]$
  - $(1 \cdot (s \circ \uparrow)) \circ (b \cdot \mathbf{id}) \rhd a$

    $$\begin{array}{lll} a[1 \cdot (s \circ \uparrow)][b \cdot \mathbf{id}] & \to_{\mathrm{Clos}} & a[(1 \cdot (s \circ \uparrow)) \circ (b \cdot \mathbf{id})] \\ & \to_{\mathrm{Map}} & a[1[b \cdot \mathbf{id}] \cdot ((s \circ \uparrow) \circ (b \cdot \mathbf{id}))] \\ & \to_{\mathrm{VarCons}} & a[b \cdot ((s \circ \uparrow) \circ (b \cdot \mathbf{id}))] \\ & \to_{\mathrm{Ass}} & a[b \cdot (s \circ (\uparrow \circ (b \cdot \mathbf{id})))] \\ & \to_{\mathrm{ShiftCons}} & a[b \cdot (s \circ \mathbf{id})] \\ & \twoheadrightarrow_{\mathrm{IdR}} & a[b \cdot s] \end{array}$$

  - $(1 \cdot (s \circ \uparrow)) \circ (b \cdot \mathbf{id}) \not\rhd a$: Notar que

    $$|(1 \cdot (s \circ \uparrow)) \circ (b \cdot \mathbf{id})| = |b \cdot s|$$

    Luego,

    $$a[1 \cdot (s \circ \uparrow)][b \cdot \mathbf{id}] \;\to_{\mathrm{ClosGC}}\; a \quad \leftarrow_{\mathrm{GC}} \quad a[b \cdot s]$$

- $1 \cdot (s \circ \uparrow) \not\rhd a$:
  - $\mathrm{FV}(a) = \emptyset$:

    $$a[1 \cdot (s \circ \uparrow)][b \cdot \mathbf{id}] \;\to_{\mathrm{GC}}\; a[b \cdot \mathbf{id}] \;\to_{\mathrm{GC}}\; a \quad \leftarrow_{\mathrm{GC}} \quad a[b \cdot s]$$

– $\mathrm{FV}(a) \neq \emptyset$:

$|s| = (m, m) \quad |1 \cdot (s \circ \uparrow)| = (m+1, m+1) \quad (\forall i \in \mathrm{FV}(a))\ i > m+1$

Luego

$$|b \cdot s| = (m+1, m)$$
$$a[1 \cdot (s \circ \uparrow)][b \cdot \mathbf{id}] \ \to_{\mathrm{GC}}\ a[b \cdot \mathbf{id}]$$

Como, por lema 70,

$$(\forall i \in \mathrm{FV}(a))\ i[b \cdot \mathbf{id}] \quad \twoheadrightarrow \quad i-1$$

y

$$(\forall i \in \mathrm{FV}(a))\ i[b \cdot s] \quad \twoheadrightarrow \quad i-1$$

Por lema 86 es cierto

$$\sigma_{\mathrm{gc}}(a[b \cdot \mathbf{id}]) = \sigma_{\mathrm{gc}}(a[b \cdot s])$$

□

Lema 105. Sean $a, b \in \Lambda\sigma^t_{\mathrm{gc}}$, $s \in \Lambda\sigma^s_{\mathrm{gc}}$, entonces

$$\sigma_{\mathrm{gc}}(a[b[s] \cdot s]) = \sigma_{\mathrm{gc}}(a[b \cdot \mathbf{id}][s])$$

Demostración. Separamos en casos:

- $b \cdot \mathbf{id} \Rsh a$
  - $s \Rsh a[b \cdot \mathbf{id}]$
    - $(b \cdot \mathbf{id}) \circ s \Rsh a$

$$\begin{array}{lcl} a[b \cdot \mathbf{id}][s] & \to_{\mathrm{Clos}} & a[(b \cdot \mathbf{id}) \circ s] \\ & \to_{\mathrm{Map\ o\ MapGC}} & a[b' \cdot (\mathbf{id} \circ s)] \\ & \to_{\mathrm{IdL}} & a[b' \cdot s] \end{array}$$

      Con $b' = b[s]$ o $b' = b$ según si $s \Rsh b$ o $s \nRsh b$. Luego,

$$\sigma_{\mathrm{gc}}(a[b[s] \cdot s]) = \sigma_{\mathrm{gc}}(a[b \cdot \mathbf{id}][s])$$

    - $(b \cdot \mathbf{id}) \circ s \nRsh a$ Como $|(b \cdot \mathbf{id}) \circ s| = |b[s] \cdot s|$

$$a[b \cdot \mathbf{id}][s] \ \to_{\mathrm{ClosGC}}\ a \quad \leftarrow_{\mathrm{GC}} \quad a[b[s] \cdot s]$$

  - $s \nRsh a[b \cdot \mathbf{id}]$
    - $\mathrm{FV}(a[b \cdot \mathbf{id}]) = \emptyset$:

$$\mathrm{FV}(a[b \cdot \mathbf{id}]) = (\mathrm{FV}(a)_{>1} + 0 - 1) \cup \mathrm{FV}(b) = \emptyset$$
$$\implies (\forall i \in \mathrm{FV}(a))\ i \leq 1 \ \wedge\ \mathrm{FV}(b) = \emptyset$$

Como $b \cdot \mathbf{id} \rhd a$, entonces $\mathrm{FV}(a) \neq \emptyset$. Luego $\mathrm{FV}(a) = \{1\}$. Vemos que para la única variable libre de a ambas sustituciones convergen a un mismo término, para concluir por lema 86 que ambos términos convergen.

$$1[b \cdot \mathbf{id}][s] \ \to_{\mathrm{VarCons}} \ b[s] \quad \leftarrow_{\mathrm{VarCons}} \quad 1[b[s] \cdot s]$$

Luego por lema 86, entonces

$$\sigma_{\mathrm{gc}}(a[b[s] \cdot s]) = \sigma_{\mathrm{gc}}(a[b \cdot \mathbf{id}][s])$$

$*$ $\mathrm{FV}(a[b \cdot \mathbf{id}]) \neq \emptyset$:
Sea $|s| = (m, m) \quad |b[s] \cdot s| = (m+1, m)$

$$\mathrm{FV}(a[b \cdot \mathbf{id}]) = (\mathrm{FV}(a)_{>1} + 0 - 1) \cup \mathrm{FV}(b)$$

Luego,

$$(\forall i \in \mathrm{FV}(a[b \cdot \mathbf{id}]))\ i > m \implies (\forall i \in \mathrm{FV}(a)_{>1})\ i - 1 > m$$

Luego, sea $i \in \mathrm{FV}(a)$

· $i = 1$:

$$1[b[s] \cdot s] \ \to_{\mathrm{VarCons}} \ b[s] \quad \leftarrow_{\mathrm{VarCons}} \quad 1[b \cdot \mathbf{id}][s]$$

· $i > 1$: Es cierto que $i > m + 1$, luego por lema 70

$$i[b[s] \cdot s] \quad \twoheadrightarrow \quad i - 1 \quad \twoheadleftarrow \quad (i-1)[s] \quad \twoheadleftarrow \quad i[b \cdot \mathbf{id}][s]$$

Por lema 86 es cierto

$$\sigma_{\mathrm{gc}}(a[b[s] \cdot s]) = \sigma_{\mathrm{gc}}(a[b \cdot \mathbf{id}][s])$$

- $b \cdot \mathbf{id} \not\rhd a$: Sólo puede ocurrir si $\mathrm{FV}(a) = \emptyset$. Luego

$$a[b \cdot \mathbf{id}][s] \ \to_{\mathrm{GC}} \ a[s] \ \to_{\mathrm{GC}} \ a \quad \leftarrow_{\mathrm{GC}} \quad a[b[s] \cdot s]$$

□

Observación 106. Sea $a \in \Lambda\sigma^t_{\mathrm{gc}}$, $s, t \in \Lambda\sigma^s_{\mathrm{gc}}$, entonces

$$\sigma_{\mathrm{gc}}((a \cdot s) \circ t) = \sigma_{\mathrm{gc}}(a[t] \cdot (s \circ t))$$

Demostración. Se aplica *Map* o *MapGC* en el término de la izquierda, y si corresponde *GC* en el de la derecha. □

Lema 107. Sean $a, b \in \Lambda_{\mathrm{DB}}$, $t, u, s \in (\Lambda\sigma^s_{\mathrm{gc}})_{\mathrm{nf}}$, entonces

1. $a \to_{\beta'} b \implies \sigma_{\mathrm{gc}}(a[s]) \to_{\beta'} \sigma_{\mathrm{gc}}(b[s])$
2. $t \to_{\beta'} u \implies \sigma_{\mathrm{gc}}(t \circ s) \to_{\beta'} \sigma_{\mathrm{gc}}(u \circ s)$

Demostración.

1. Inducción en a:

   - $a = 1, n$ se satisface vacuamente.
   - $a = \lambda c \quad c \to_{\beta'} c' \quad b = \lambda c'$

$$\begin{array}{rll}
\sigma_{\mathrm{gc}}((\lambda c)[s]) & =_{\text{L 102}} & \sigma_{\mathrm{gc}}(\lambda c[1 \cdot (s \circ \uparrow)]) \\
& =_{\text{L 87}} & \lambda\sigma_{\mathrm{gc}}(c[1 \cdot (s \circ \uparrow)]) \\
& =_{\text{L 87}} & \lambda\sigma_{\mathrm{gc}}(\sigma_{\mathrm{gc}}(c)[\sigma_{\mathrm{gc}}(1 \cdot (s \circ \uparrow))]) \\
& = & \lambda\sigma_{\mathrm{gc}}(c[\sigma_{\mathrm{gc}}(1 \cdot (s \circ \uparrow))]) \\
& \overset{\text{h.i.}}{\to_{\beta'}} & \lambda\sigma_{\mathrm{gc}}(c'[\sigma_{\mathrm{gc}}(1 \cdot (s \circ \uparrow))]) \\
& = & \lambda\sigma_{\mathrm{gc}}(\sigma_{\mathrm{gc}}(c')[\sigma_{\mathrm{gc}}(1 \cdot (s \circ \uparrow))]) \\
& =_{\text{L 87}} & \lambda\sigma_{\mathrm{gc}}(c'[1 \cdot (s \circ \uparrow)]) \\
& =_{\text{L 87}} & \sigma_{\mathrm{gc}}(\lambda c'[1 \cdot (s \circ \uparrow)]) \\
& =_{\text{L 102}} & \sigma_{\mathrm{gc}}((\lambda c')[s])
\end{array}$$

   - $a = a_1\ a_2$
     - $a_1 \to_{\beta'} a_1' \quad b = a_1'\ a_2$

$$\begin{array}{rll}
\sigma_{\mathrm{gc}}((a_1\ a_2)[s]) & =_{\text{L 103}} & \sigma_{\mathrm{gc}}(a_1[s])\ \sigma_{\mathrm{gc}}(a_2[s]) \\
& \to_{\beta'\text{h.i.}} & \sigma_{\mathrm{gc}}(a_1'[s])\ \sigma_{\mathrm{gc}}(a_2[s]) \\
& =_{\text{L 103}} & \sigma_{\mathrm{gc}}(a_1'[s]\ a_2[s])
\end{array}$$

     - $a_2 \to_{\beta'} a_2' \quad b = a_1\ a_2'$

$$\begin{array}{rll}
\sigma_{\mathrm{gc}}((a_1\ a_2)[s]) & =_{\text{L 103}} & \sigma_{\mathrm{gc}}(a_1[s])\ \sigma_{\mathrm{gc}}(a_2[s]) \\
& \to_{\beta'\text{h.i.}} & \sigma_{\mathrm{gc}}(a_1[s])\ \sigma_{\mathrm{gc}}(a_2'[s]) \\
& =_{\text{L 103}} & \sigma_{\mathrm{gc}}(a_1[s]\ a_2'[s])
\end{array}$$

     - $a_1 = \lambda c \quad b = c\{1 \leftarrow a_2\}$

$$\begin{array}{rll}
\sigma_{\mathrm{gc}}(((\lambda c)\ a_2)[s]) & =_{\text{L 103}} & \sigma_{\mathrm{gc}}((\lambda c)[s])\ \sigma_{\mathrm{gc}}(a_2[s]) \\
& =_{\text{L 102}} & \sigma_{\mathrm{gc}}(\lambda c[1 \cdot (s \circ \uparrow)])\ \sigma_{\mathrm{gc}}(a_2[s]) \\
& =_{\text{L 87}} & \lambda\sigma_{\mathrm{gc}}(c[1 \cdot (s \circ \uparrow)])\ \sigma_{\mathrm{gc}}(a_2[s]) \\
& \to_{\beta'} & \sigma_{\mathrm{gc}}(c[1 \cdot (s \circ \uparrow)])\{1 \leftarrow \sigma_{\mathrm{gc}}(a_2[s])\} \\
& =_{\text{Coro 100}} & \sigma_{\mathrm{gc}}(\sigma_{\mathrm{gc}}(c[1 \cdot (s \circ \uparrow)])[\sigma_{\mathrm{gc}}(a_2[s]) \cdot \mathbf{id}]) \\
& =_{\text{L 87}} & \sigma_{\mathrm{gc}}(c[1 \cdot (s \circ \uparrow)][a_2[s] \cdot \mathbf{id}]) \\
& =_{\text{L 104}} & \sigma_{\mathrm{gc}}(c[a_2[s] \cdot s]) \\
& =_{\text{L 105}} & \sigma_{\mathrm{gc}}(c[a_2 \cdot \mathbf{id}][s]) \\
& =_{\text{L 87}} & \sigma_{\mathrm{gc}}(\sigma_{\mathrm{gc}}(c[a_2 \cdot \mathbf{id}])[s]) \\
& =_{\text{Coro 100}} & \sigma_{\mathrm{gc}}((c\{1 \leftarrow a_2\})[s])
\end{array}$$

2. Inducción en t:

   - $t = \mathbf{id}, \uparrow^n$ se satisface vacuamente.
   - $t = a \cdot t_1$

– $a \to_{\beta'} a' \quad u = a' \cdot t_1$

$$\begin{array}{rll} \sigma_{\mathrm{gc}}(t \circ s) & =_{\text{Obs 106}} & \sigma_{\mathrm{gc}}(a[s] \cdot (t_1 \circ s)) \\ & =_{\text{L 87}} & \sigma_{\mathrm{gc}}(a[s]) \cdot \sigma_{\mathrm{gc}}(t_1 \circ s) \\ & \to_{\beta'\text{item 1}} & \sigma_{\mathrm{gc}}(a'[s]) \cdot \sigma_{\mathrm{gc}}(t_1 \circ s) \\ & =_{\text{L 87}} & \sigma_{\mathrm{gc}}(a'[s] \cdot (t_1 \circ s)) \\ & =_{\text{Obs 106}} & \sigma_{\mathrm{gc}}(u \circ s) \end{array}$$

– $t_1 \to_{\beta'} t_1' \quad u = a \cdot t_1'$

$$\begin{array}{rll} \sigma_{\mathrm{gc}}(t \circ s) & =_{\text{Obs 106}} & \sigma_{\mathrm{gc}}(a[s] \cdot (t_1 \circ s)) \\ & =_{\text{L 87}} & \sigma_{\mathrm{gc}}(a[s]) \cdot \sigma_{\mathrm{gc}}(t_1 \circ s) \\ & \to_{\beta'\text{h. i.}} & \sigma_{\mathrm{gc}}(a[s]) \cdot \sigma_{\mathrm{gc}}(t_1' \circ s) \\ & =_{\text{L 87}} & \sigma_{\mathrm{gc}}(a[s] \cdot (t_1' \circ s)) \\ & =_{\text{Obs 106}} & \sigma_{\mathrm{gc}}(u \circ s) \end{array}$$

□

Corolario 108. Sean $a, b \in \Lambda_{\mathrm{DB}}$, $t, u, s \in (\Lambda\sigma^s_{\mathrm{gc}})_{\mathrm{nf}}$, entonces

1. $a \twoheadrightarrow_{\beta'} b \implies \sigma_{\mathrm{gc}}(a[s]) \twoheadrightarrow_{\beta'} \sigma_{\mathrm{gc}}(b[s])$
2. $t \twoheadrightarrow_{\beta'} u \implies \sigma_{\mathrm{gc}}(t \circ s) \twoheadrightarrow_{\beta'} \sigma_{\mathrm{gc}}(u \circ s)$

Demostración. Por inducción en la longitud de la derivación. □

Lema 109. Sean $a \in \Lambda_{\mathrm{DB}}$, $t, u, s \in (\Lambda\sigma^s_{\mathrm{gc}})_{\mathrm{nf}}$, con $s \to_{\beta'} t$, entonces

1. $\sigma_{\mathrm{gc}}(a[s]) \twoheadrightarrow_{\beta'} \sigma_{\mathrm{gc}}(a[t])$
2. $\sigma_{\mathrm{gc}}(u \circ s) \twoheadrightarrow_{\beta'} \sigma_{\mathrm{gc}}(u \circ t)$

Demostración. $s = a_1 \cdot \ldots \cdot a_i \cdot \ldots \cdot a_k \cdot \uparrow^n \quad t = a_1 \cdot \ldots \cdot a_i' \cdot \ldots \cdot a_k \cdot \uparrow^n \quad$ con $a_i \to_{\beta'} a_i'$

1. Inducción en a. Notar que $i \geq 1$, pues si no no existiría un redex.
   - $a = 1$
     - $i = 1$
       $$\sigma_{\mathrm{gc}}(1[s]) = a_1 \to_{\beta'} a_1' = \sigma_{\mathrm{gc}}(1[t])$$
     - $i > 1$
       $$\sigma_{\mathrm{gc}}(1[s]) = a_1 = \sigma_{\mathrm{gc}}(1[t])$$
   - $a = m > 1$
     - $m \leq k \quad i = m$
       $$\sigma_{\mathrm{gc}}(1[\uparrow^{m-1}][s]) = a_i \to_{\beta'} a_i' = \sigma_{\mathrm{gc}}(1[\uparrow^{m-1}][t])$$
     - $m \leq k \quad i \neq m$
       $$\sigma_{\mathrm{gc}}(1[\uparrow^{m-1}][s]) = a_m = \sigma_{\mathrm{gc}}(1[\uparrow^{m-1}][t])$$

- $m > k$

$$\sigma_{\mathrm{gc}}(1[\uparrow^{m-1}][s]) = n + m - k = \sigma_{\mathrm{gc}}(1[\uparrow^{m-1}][t])$$

- $a = b\ c$

$$\begin{array}{rll} \sigma_{\mathrm{gc}}((b\ c)[s]) & =_{\text{L 103}} & \sigma_{\mathrm{gc}}(b[s])\ \sigma_{\mathrm{gc}}(c[s]) \\ & \twoheadrightarrow_{\beta'} \text{ h.i.} & \sigma_{\mathrm{gc}}(b[t])\ \sigma_{\mathrm{gc}}(c[t]) \\ & =_{\text{L 103}} & \sigma_{\mathrm{gc}}((b\ c)[t]) \end{array}$$

- $a = \lambda b$

$$\begin{array}{rll} \sigma_{\mathrm{gc}}((\lambda b)[s]) & =_{\text{L 102}} & \sigma_{\mathrm{gc}}(\lambda b[1 \cdot (s \circ \uparrow)]) \\ & =_{\text{L 87}} & \lambda\sigma_{\mathrm{gc}}(b[1 \cdot (s \circ \uparrow)]) \\ & =_{\text{L 87}} & \lambda\sigma_{\mathrm{gc}}(\sigma_{\mathrm{gc}}(b)[\sigma_{\mathrm{gc}}(1 \cdot (s \circ \uparrow))]) \\ & = & \lambda\sigma_{\mathrm{gc}}(b[\sigma_{\mathrm{gc}}(1 \cdot (s \circ \uparrow))]) \\ & =_{\text{L 87}} & \lambda\sigma_{\mathrm{gc}}(b[1 \cdot (\sigma_{\mathrm{gc}}(s \circ \uparrow))]) \\ & \overset{\text{L 107+h.i.}}{\twoheadrightarrow_{\beta'}} & \lambda\sigma_{\mathrm{gc}}(b[1 \cdot (\sigma_{\mathrm{gc}}(t \circ \uparrow))]) \\ & =_{\text{L 87}} & \lambda\sigma_{\mathrm{gc}}(b[1 \cdot (t \circ \uparrow)]) \\ & =_{\text{L 102}} & \sigma_{\mathrm{gc}}((\lambda b)[t]) \end{array}$$

2. Inducción en u:

- $u = \mathbf{id}$

$$\sigma_{\mathrm{gc}}(\mathbf{id} \circ s) = s \to_{\beta'} t = \sigma_{\mathrm{gc}}(\mathbf{id} \circ t)$$

- $u = \uparrow^m$
  - $m < k\ \wedge\ i > m$

$$\sigma_{\mathrm{gc}}(\uparrow^m \circ s) = a_{m+1} \cdot \ldots \cdot a_i \cdot \ldots \cdot a_k \cdot \uparrow^n\ \to_{\beta'} a_{m+1} \cdot \ldots \cdot a'_i \cdot \ldots \cdot a_k \cdot \uparrow^n = \sigma_{\mathrm{gc}}(\uparrow^m \circ t)$$

  - $m < k\ \wedge\ i \leq m$

$$\sigma_{\mathrm{gc}}(\uparrow^m \circ s) = a_{m+1} \cdot \ldots \cdot a_k \cdot \uparrow^n = \sigma_{\mathrm{gc}}(\uparrow^m \circ t)$$

  - $m \geq k$

$$\sigma_{\mathrm{gc}}(\uparrow^m \circ s) = \uparrow^{m+n-k} = \sigma_{\mathrm{gc}}(\uparrow^m \circ t)$$

- $u = a \cdot u_1$

$$\begin{array}{rll} \sigma_{\mathrm{gc}}((a \cdot u_1) \circ s) & =_{\text{L 87}} & \sigma_{\mathrm{gc}}(a[s]) \cdot \sigma_{\mathrm{gc}}(u_1 \circ s) \\ & \twoheadrightarrow_{\beta'} \text{item 1 y h. i.} & \sigma_{\mathrm{gc}}(a[t]) \cdot \sigma_{\mathrm{gc}}(u_1 \circ t) \\ & =_{\text{L 87}} & \sigma_{\mathrm{gc}}((a \cdot u_1) \circ t) \end{array}$$

□

Corolario 110. Sean $a \in \Lambda_{\mathrm{DB}}$, $t, u, s \in (\Lambda\sigma^s_{\mathrm{gc}})_{\mathrm{nf}}$, con $s \twoheadrightarrow_{\beta'} t$, entonces

1. $\sigma_{\mathrm{gc}}(a[s]) \twoheadrightarrow_{\beta'} \sigma_{\mathrm{gc}}(a[t])$
2. $\sigma_{\mathrm{gc}}(u \circ s) \twoheadrightarrow_{\beta'} \sigma_{\mathrm{gc}}(u \circ t)$

Demostración. Por inducción en la longitud de la derivación $s \twoheadrightarrow_{\beta'} t$ □

Lema 111. Sean $w, v \in \Lambda\sigma_{\mathrm{gc}}$, entonces

$$w \to_{\mathrm{Beta}} v \implies \sigma_{\mathrm{gc}}(w) \twoheadrightarrow_{\beta'} \sigma_{\mathrm{gc}}(v)$$

Demostración. Inducción en w:

- $w = 1, \mathbf{id}, \uparrow$ se satisface vacuamente.
- $w = a\ b$
  - $v = a'\ b, \quad a \to_{\mathrm{Beta}} a'$

    $$\sigma_{\mathrm{gc}}(a\ b) =_{\text{L 87}} \sigma_{\mathrm{gc}}(a)\ \sigma_{\mathrm{gc}}(b) \twoheadrightarrow_{\text{h. i.}} \sigma_{\mathrm{gc}}(a')\ \sigma_{\mathrm{gc}}(b) =_{\text{L 87}} \sigma_{\mathrm{gc}}(a'\ b)$$

  - $v = a\ b', \quad b \to_{\mathrm{Beta}} b'$

    $$\sigma_{\mathrm{gc}}(a\ b) =_{\text{L 87}} \sigma_{\mathrm{gc}}(a)\ \sigma_{\mathrm{gc}}(b) \twoheadrightarrow_{\text{h. i.}} \sigma_{\mathrm{gc}}(a)\ \sigma_{\mathrm{gc}}(b') =_{\text{L 87}} \sigma_{\mathrm{gc}}(a\ b')$$

  - $a = \lambda c, \quad v = c[b \cdot \mathbf{id}]$

    $$\begin{aligned} \sigma_{\mathrm{gc}}((\lambda c)\ b) &=_{\text{L 87}} \lambda\sigma_{\mathrm{gc}}(c)\ \sigma_{\mathrm{gc}}(b) \\ &\to_{\beta'} \sigma_{\mathrm{gc}}(c)\{1 \leftarrow \sigma_{\mathrm{gc}}(b)\} \\ &=_{\text{Coro 100}} \sigma_{\mathrm{gc}}(\sigma_{\mathrm{gc}}(c)[\sigma_{\mathrm{gc}}(b) \cdot \mathbf{id}]) \\ &=_{\text{L 87}} \sigma_{\mathrm{gc}}(c[b \cdot \mathbf{id}]) \end{aligned}$$

- $w = \lambda a, \quad v = \lambda a', \quad a \to_{\mathrm{Beta}} a'$

  $$\begin{aligned} \sigma_{\mathrm{gc}}(\lambda a) &=_{\text{L 87}} \lambda\sigma_{\mathrm{gc}}(a) \\ &\twoheadrightarrow_{\text{h. i.}} \lambda\sigma_{\mathrm{gc}}(a') \\ &=_{\text{L 87}} \sigma_{\mathrm{gc}}(\lambda a') \end{aligned}$$

- $w = a[s]$
  - $v = a'[s], \quad a \to_{\mathrm{Beta}} a'$

    $$\begin{aligned} \sigma_{\mathrm{gc}}(a[s]) &=_{\text{L 87}} \sigma_{\mathrm{gc}}(\sigma_{\mathrm{gc}}(a)[\sigma_{\mathrm{gc}}(s)]) \\ &\twoheadrightarrow_{\text{h. i. y Coro 108}} \sigma_{\mathrm{gc}}(\sigma_{\mathrm{gc}}(a')[\sigma_{\mathrm{gc}}(s)]) \\ &=_{\text{L 87}} \sigma_{\mathrm{gc}}(a'[s]) \end{aligned}$$

  - $v = a[s'], \quad s \to_{\mathrm{Beta}} s'$

    $$\begin{aligned} \sigma_{\mathrm{gc}}(a[s]) &=_{\text{L 87}} \sigma_{\mathrm{gc}}(\sigma_{\mathrm{gc}}(a)[\sigma_{\mathrm{gc}}(s)]) \\ &\twoheadrightarrow_{\text{h. i. y Coro 110}} \sigma_{\mathrm{gc}}(\sigma_{\mathrm{gc}}(a)[\sigma_{\mathrm{gc}}(s')]) \\ &=_{\text{L 87}} \sigma_{\mathrm{gc}}(a[s']) \end{aligned}$$

- $w = a \cdot s$
  - $v = a' \cdot s, \quad a \to_{\mathrm{Beta}} a'$

    $$\sigma_{\mathrm{gc}}(a \cdot s) =_{\text{L 87}} \sigma_{\mathrm{gc}}(a) \cdot \sigma_{\mathrm{gc}}(s) \twoheadrightarrow_{\text{h. i.}} \sigma_{\mathrm{gc}}(a') \cdot \sigma_{\mathrm{gc}}(s) =_{\text{L 87}} \sigma_{\mathrm{gc}}(a' \cdot s)$$

- $v = a \cdot s', \quad s \to_{\text{Beta}} s'$

  $\sigma_{\text{gc}}(a \cdot s) =_{\text{L 87}} \sigma_{\text{gc}}(a) \cdot \sigma_{\text{gc}}(s) \twoheadrightarrow_{\text{h. i.}} \sigma_{\text{gc}}(a) \cdot \sigma_{\text{gc}}(s') =_{\text{L 87}} \sigma_{\text{gc}}(a \cdot s')$

- $w = s \circ t$
  - $v = s' \circ t, \quad s \to_{\text{Beta}} s'$

$$\begin{array}{lll} \sigma_{\text{gc}}(s \circ t) & =_{\text{L 87}} & \sigma_{\text{gc}}(\sigma_{\text{gc}}(s) \circ \sigma_{\text{gc}}(t)) \\ & \twoheadrightarrow_{\text{h. i. y Coro 108}} & \sigma_{\text{gc}}(\sigma_{\text{gc}}(s') \circ \sigma_{\text{gc}}(t)) \\ & =_{\text{L 87}} & \sigma_{\text{gc}}(s' \circ t) \end{array}$$

  - $v = s \circ t', \quad t \to_{\text{Beta}} t'$

$$\begin{array}{lll} \sigma_{\text{gc}}(s \circ t) & =_{\text{L 87}} & \sigma_{\text{gc}}(\sigma_{\text{gc}}(s) \circ \sigma_{\text{gc}}(t)) \\ & \twoheadrightarrow_{\text{h. i. y Coro 110}} & \sigma_{\text{gc}}(\sigma_{\text{gc}}(s) \circ \sigma_{\text{gc}}(t')) \\ & =_{\text{L 87}} & \sigma_{\text{gc}}(s \circ t') \end{array}$$

□

Corolario 112. Sean $w, v \in \Lambda\sigma_{\text{gc}}$, entonces

$$w \to_{\lambda\sigma_{\text{gc}}} v \implies \sigma_{\text{gc}}(w) \twoheadrightarrow_{\beta'} \sigma_{\text{gc}}(v)$$

Demostración. Según el paso que se realice, si es *Beta* entonces se aplica el lema 111. Para cualquier otro paso, $\sigma_{\text{gc}}(w) = \sigma_{\text{gc}}(v)$. □

Corolario 113. Sean $w, v \in \Lambda\sigma_{\text{gc}}$, entonces

$$w \twoheadrightarrow_{\lambda\sigma_{\text{gc}}} v \implies \sigma_{\text{gc}}(w) \twoheadrightarrow_{\beta'} \sigma_{\text{gc}}(v)$$

Demostración. Por inducción en la longitud de la derivación. □

Corolario 114 (Corrección (Soundness))**.** Sean $w, v \in (\Lambda\sigma_{\text{gc}})_{\text{nf}}$, entonces

$$w \twoheadrightarrow_{\lambda\sigma_{\text{gc}}} v \implies w \twoheadrightarrow_{\beta'} v$$

En particular, para $a, b \in \Lambda_{\text{DB}}$,

$$a \twoheadrightarrow_{\lambda\sigma_{\text{gc}}} b \implies a \twoheadrightarrow_{\beta} b$$

Teorema 115. $\lambda\sigma_{\text{gc}}$ es confluente sobre $\Lambda\sigma_{\text{gc}}$.

Demostración. Se prueba con el lema de interpretación (ver lema 7).
Sean $R_1 = \sigma_{\text{gc}} \quad R_2 = \to_{\text{Beta}} \quad R' = \to_{\beta'}$
Las hipótesis de dicho método son:

1. σ_{gc} es SN. Se probó en el capítulo 3.
2. $\to_{\beta'}$ es una relación en las σ_{gc}-f. n. tal que $\to_{\beta'} \subseteq \lambda\sigma_{\text{gc}}{}^*$. El teorema 101 asegura esta hipótesis.
3. $a \to_{\text{Beta}} b \implies \sigma_{\text{gc}}(a) \twoheadrightarrow_{\beta'} \sigma_{\text{gc}}(b)$ Esto se mostró en el lema 111.

Luego, $\lambda\sigma_{\text{gc}}$ es CR $\iff$ $\to_{\beta'}$ es CR. Por lema 88 sabemos que $\to_{\beta'}$ es CR, luego $\lambda\sigma_{\text{gc}}$ es CR. □

Capítulo 7

El cálculo $\lambda\sigma_{\mathbf{gc}}$ tipado

La versión tipada del cálculo $\lambda\sigma_{\mathrm{gc}}$ es muy similiar a la presentada en la sección 1.4.1. En este capítulo se muestran las diferencias con el sistema anterior, y se demuestra que se mantiene la propiedad Subject Reduction.

Los términos, sustituciones, tipos y bases serán los mismos, y por ende las reglas de tipado serán las mismas. La única diferencia, entonces, son las reglas del cálculo, que serán las mismas que las de $\lambda\sigma_{\mathrm{gc}}$, con excepción de *Beta*, *VarCons*, *Abs*, *ShiftCons*, *Map*, y *MapGC* que se reemplazan por sus versiones con anotaciones de tipo:

(Beta)	$(\lambda A.\, a)\ b$	$\longrightarrow$	$a[b : A \cdot \mathbf{id}]$	
(VarCons)	$1[a : A \cdot s]$	$\longrightarrow$	a	
(Abs)	$(\lambda A.\, a)[s]$	$\longrightarrow$	$\lambda A.\, (a[1 : A \cdot (s \circ \uparrow)])$	(si $s \Rrightarrow \lambda a$)
(ShiftCons)	$\uparrow \circ (a : A \cdot s)$	$\longrightarrow$	s	
(Map)	$(a : A \cdot s) \circ t$	$\longrightarrow$	$a[t] : A \cdot (s \circ t)$	(si $t \Rrightarrow a$)
(MapGC)	$(a : A \cdot s) \circ t$	$\longrightarrow$	$a : A \cdot (s \circ t)$	(si $t \nRightarrow a$)

Teorema 116 (Subject Reduction)**.** Sean a y b términos, s y t sustituciones, E y E' bases,

$$a \to_{\lambda\sigma_{\mathrm{gc}}} b \quad \text{entonces} \quad E \vdash a : A \Longrightarrow E \vdash b : A$$

$$s \to_{\lambda\sigma_{\mathrm{gc}}} t \quad \text{entonces} \quad E \vdash s \rhd E' \Longrightarrow E \vdash t \rhd E'$$

Demostración. Por inducción en el contexto de la reducción. Mostramos que se cumple en el caso base, es decir, cuando la reducción es en la raíz del término. El caso inductivo tiene idéntica resolución que para $\lambda\sigma$, y no lo mostraremos.

Beta: Sea $(\lambda A.\, a)\ b \to_{\mathrm{Beta}} a[b : A \cdot \mathbf{id}]$. Supongamos $E \vdash (\lambda A.\, a)\ b : B$. Por *app*, $E \vdash (\lambda A.\, a) : A \to B$ y $E \vdash b : A$. Por *abs* $A, E \vdash a : B$. Luego, $E \vdash b : A \cdot \mathbf{id} \rhd A, E$, con lo que por *clos* podemos concluir que $E \vdash a[b : A \cdot \mathbf{id}] : B$.

GC: Sea $a[s] \to_{\mathrm{GC}} a$, con $s \nRightarrow a$. Supongamos que $E \vdash a[s] : A$. Supongamos que $E \vdash a : B$, con $A \neq B$. Por teorema 33 obtenemos que $E \vdash \sigma(a)[\sigma(s)] : A$. Por corolario 64 y 66, $s \nRightarrow a \Longrightarrow \sigma(s) \nRightarrow \sigma(a)$. Por

lema 47, $\sigma(a)[\sigma(s)] \twoheadrightarrow_\sigma \sigma(a)$, con lo que nuevamente por teorema 33 obtenemos $E \vdash \sigma(a) : A$. Pero por teorema 33 también concluimos que $E \vdash \sigma(a) : B$, lo que es absurdo. El absurdo provino de suponer que $A \neq B$.

VarCons : Sea $1[a : A \cdot s] \to_{\text{VarCons}} a$. Supongamos $E \vdash 1[a : A \cdot s] : B$. Por *clos*, $E \vdash a : A \cdot s : E'$ y $E' \vdash 1 : B$, para algún E'. Por *cons*, $E \vdash a : A \cdot s \rhd A, E'' = E'$, con $E \vdash s \rhd E''$. Por *var*, $A, E'' \vdash 1 : A$, luego $A = B$.

App_1: Sea $(a\ b)[s] \to_{\text{App}_1} a[s]\ b[s]$, con $s \Vdash a$ y $s \Vdash b$. Supongamos $E \vdash (a\ b)[s] : A$. Por *clos*, $E \vdash s \rhd E'$ y $E' \vdash a\ b : A$, luego $E' \vdash a : B \to A$ y $E' \vdash b : B$. Por *clos* obtenemos $E \vdash a[s] : B \to A$ y $E \vdash b[s] : B$. Luego, por *app* obtenemos $E \vdash a[s]\ b[s] : A$.

App_2: Sea $(a\ b)[s] \to_{\text{App}_2} a[s]\ b$, con $s \Vdash a$ y $s \not\Vdash b$. Supongamos $E \vdash (a\ b)[s] : A$. Por *clos*, $E \vdash s \rhd E'$ y $E' \vdash a\ b : A$, luego $E' \vdash a : B \to A$ y $E' \vdash b : B$. Por *clos* obtenemos $E \vdash a[s] : B \to A$, y $E \vdash b[s] : B$. Pero, siguiendo el razonamiento de *GC*, $E \vdash b : B$, con lo que, por *app*, obtenemos $E \vdash a[s]\ b : A$.

App_3: Se sigue el mismo razonamiento que en App_2.

Abs: Sea $(\lambda A.\, a)[s] \to_{\text{Abs}} \lambda A.\, a[1 : A \cdot (s\circ \uparrow)]$, con $s \Vdash \lambda A.a$. Supongamos $E \vdash (\lambda A.\, a)[s] : B$. Por *clos*, $E \vdash s \rhd E'$ y $E' \vdash \lambda A.\, a : B$. Por *lambda*, $B = A \to C$, y $A, E' \vdash a : C$. Por *shift*, $A, E \vdash \uparrow \rhd E$, y por *clos*, $A, E \vdash s\circ \uparrow \rhd E'$. Como $A, E \vdash 1 : A$ por *var*, por *cons* nos queda $A, E \vdash 1 : A \cdot (s\circ \uparrow) \rhd A, E'$. Finalmente, como $A, E' \vdash a : C$, por *clos* obtenemos $A, E \vdash a[1 : A \cdot (s\circ \uparrow)] : C$, y por ende, $E \vdash \lambda A.\, a[1 : A \cdot (s\circ \uparrow)] : A \to C = B$ por *lambda*.

Clos: Sea $a[s][t] \to_{\text{Clos}} a[s \circ t]$, con $s \Vdash a$, $t \Vdash a[s]$, y $s \circ t \Vdash a$. Supongamos $E \vdash a[s][t] : A$. Por *clos*, $E \vdash t \rhd E'$ y $E' \vdash a[s] : A$, luego $E' \vdash s \rhd E''$ y $E'' \vdash a : A$. Luego, por *comp* obtenemos que $E \vdash s \circ t \rhd E''$, y finalmente por *clos*, $E \vdash a[s \circ t] : A$.

ClosGC: Se sigue el mismo razonamiento que para *Clos*, y se concluye por *GC* que $E \vdash a : A$.

IdL: Sea $\mathbf{id}\circ s \to_{\text{IdL}} s$. Supongamos $E \vdash \mathbf{id}\circ s \rhd E''$. Por *comp*, $E \vdash s \rhd E'$ y $E' \vdash \mathbf{id} \rhd E''$. Pero por *id*, $E'' = E'$, luego $E \vdash s \rhd E''$.

ShiftId: Sea $\uparrow \circ \mathbf{id} \to_{\text{ShiftId}} \uparrow$. Sea $E \vdash \uparrow \circ \mathbf{id} \rhd E'$. Por *id*, *shift*, y *comp*, $E = A, E'$. Luego, por *shift* nuevamente, $A, E' \vdash \uparrow \rhd E'$.

ShiftCons: Sea $\uparrow \circ (a : A \cdot s) \to_{\text{ShiftCons}} s$. Supongamos $E \vdash \uparrow \circ (a : A \cdot s) \rhd E''$. Por *comp*, $E \vdash a : A \cdot s \rhd E'$, y $E' \vdash \uparrow \rhd E''$. Por *cons*, $E \vdash a : A$ y $E \vdash s \rhd E_1$, con $E' = A, E_1$. Luego, $E'' = E_1$ y $E \vdash s \rhd E''$.

Map: Sea $(a : A \cdot s) \circ t \to_{\text{Map}} a[t] : A \cdot (s \circ t)$, con $s \Vdash a$. Supongamos $E \vdash (a : A \cdot s) \circ t \rhd E'$. Por *comp*, $E \vdash t : E_2$ y $E_2 \vdash a : A \cdot s : E'$. Por *cons*, $E_2 \vdash a : A$ y $E_2 \vdash s \rhd E_3$, con $E' = A, E_3$. Por *comp*, $E \vdash s \circ t \rhd E_3$. Finalmente, por *clos* $E \vdash a[t] : A$, y por *cons*, $E \vdash a[t] : A \cdot (s \circ t) \rhd A, E_3 = E'$.

MapGC: Se sigue el mismo razonamiento que para *Map*, y se concluye por GC que $E \vdash a : A \cdot (s \circ t) \rhd E'$.

Ass: Sea $(s \circ t) \circ u \to_{\text{Ass}} s \circ (t \circ u)$. Sea $E \vdash (s \circ t) \circ u \rhd E'$. Por *comp*, $E \vdash u \rhd E_1$, y $E_1 \vdash s \circ t \rhd E'$. Por *comp* de nuevo, $E_1 \vdash t \rhd E_2$ y $E_2 \vdash s \rhd E'$. Luego, por *comp*, $E \vdash t \circ u \rhd E_2$, y por *comp* de nuevo, se concluye que $E \vdash s \circ (t \circ u) \rhd E'$

□

Capítulo 8

$\lambda\sigma_{\mathbf{gc}}$ y PSN

En este capítulo se discute la relación entre los términos fuertemente normalizantes de λ, $\lambda\sigma$, y $\lambda\sigma_{\mathrm{gc}}$. Primero mostraremos en 8.1 que el cálculo $\lambda\sigma_{\mathrm{gc}}$ resuelve el contraejemplo que demuestra que $\lambda\sigma$ no cumple PSN. Luego, en 8.2, hablaremos sobre dos posibles escenarios acerca de la relación entre estos cálculos, en términos de la normalización fuerte.

8.1 $\lambda\sigma_{\mathbf{gc}}$ resuelve el contraejemplo de Melliès

El término que presenta Melliès para mostrar no PSN de $\lambda\sigma$ es

$$T = \lambda((\lambda I\ (I\ 1))\ (I\ 1))$$

donde $I = \lambda 1$.

Este término resulta de la traducción del término en λ-cálculo

$$(\lambda v.(\lambda x.(\lambda y.y)\ ((\lambda z.z)\ x))\ ((\lambda w.w)\ v))$$

que es tipable (en λ y en $\lambda\sigma$).

Sea s_i definido en los naturales como

$$\begin{aligned} s_1 &= ((\lambda a)\ b) \cdot \mathbf{id} \\ s_{i+1} &= \uparrow \circ (b[s_i] \cdot \mathbf{id}) \end{aligned}$$

para a y b dos términos cualesquiera.

Desde el término T existe una derivación a un término que contiene a $s_1 \circ s_2$ como subtérmino, y desde allí puede encontrarse una derivación infinita de la

siguiente manera:

$$\begin{array}{rcl}
s_1 \circ s_2 & = & (((\lambda a)\ b) \cdot \mathbf{id}) \circ s_2 \\
& \to_{\text{Map}} & ((\lambda a)\ b)[s_2] \cdot (\mathbf{id} \circ s_2) \\
& \to_{\text{App + IdL}} & ((\lambda a)[s_2]\ b[s_2]) \cdot s_2 \\
& \to_{\text{Abs}} & ((\lambda a[1 \cdot (s_2 \circ \uparrow)])\ b[s_2]) \cdot s_2 \\
& \to_{\text{Beta}} & a[1 \cdot (s_2 \circ \uparrow)][b[s_2] \cdot \mathbf{id}] \cdot s_2 \\
& \to_{\text{Clos}} & a[(1 \cdot (s_2 \circ \uparrow)) \circ (b[s_2] \cdot \mathbf{id})] \cdot s_2 \\
& \to_{\text{Map}} & a[1[b[s_2] \cdot \mathbf{id}] \cdot ((s_2 \circ \uparrow) \circ (b[s_2] \cdot \mathbf{id}))] \cdot s_2 \\
& \to_{\text{VarCons}} & a[b[s_2] \cdot ((s_2 \circ \uparrow) \circ (b[s_2] \cdot \mathbf{id}))] \cdot s_2 \\
& \to_{\text{Ass}} & a[b[s_2] \cdot (\boxed{s_2 \circ (\uparrow \circ (b[s_2] \cdot \mathbf{id}))})] \cdot s_2
\end{array}$$

Se puede notar que el subtérmino remarcado es equivalente a $s_2 \circ s_3$. Partiendo de $s_2 \circ s_3$ se puede ver que existe un contexto C tal que $s_2 \circ s_3 \twoheadrightarrow C[s_3 \circ s_4]$, y en general se puede demostrar que

$$(\forall i \in \mathbb{N}_{>0})\ (\exists C)\ s_i \circ s_{i+1} \twoheadrightarrow C[s_{i+1} \circ s_{i+2}]$$

obteniendo así la derivación infinita.

En $\lambda\sigma_{\text{gc}}$, en vez de aplicar *Map* como primer paso, se aplica *MapGC*, puesto que para $i > 1$, $|s_i| = (0, 0)$. Luego, la derivación será

$$\begin{array}{rcl}
s_1 \circ s_2 & = & (((\lambda a)\ b) \cdot \mathbf{id}) \circ s_2 \\
& \to_{\text{MapGC}} & ((\lambda a)\ b) \cdot (\mathbf{id} \circ s_2) \\
& \to_{\text{Beta + IdL}} & a[b \cdot \mathbf{id}] \cdot s_2
\end{array}$$

En el apéndice C se muestra un programa que obtiene la prueba para $T \in \mathcal{SN}_{\lambda\sigma_{\text{gc}}}$ mediante el análisis exhaustivo de todas las derivaciones posibles.

8.2 Problema abierto: $\mathcal{SN}_{\lambda\sigma} \overset{?}{\subset} \mathcal{SN}_{\lambda\sigma_{\mathbf{gc}}} \overset{?}{\supset} \mathcal{SN}_{\lambda}$

Si se estableciera que todo término fuertemente normalizante de $\lambda\sigma$ es SN en $\lambda\sigma_{\text{gc}}$, entonces se tendría que, efectivamente, $\lambda\sigma_{\text{gc}}$ se encuentra "más cerca" de preservar la normalización fuerte que $\lambda\sigma$. La figura 8.1 muestra gráficamente esto.

Sea $\lambda\sigma + GC$ el cálculo resultante de agregar la regla GC a $\lambda\sigma$. Se puede ver que todo término SN de este cálculo es SN en $\lambda\sigma_{\text{gc}}$, puesto que la función identidad es una función estricta de $\lambda\sigma + GC$ a $\lambda\sigma_{\text{gc}}$ (ver capítulo 3). Luego, quedaría por ver que todo término SN de $\lambda\sigma$ es SN en $\lambda\sigma + GC$. Cabe mencionar que esto no es cierto si se quita la condición de la regla GC, como lo muestra el siguiente ejemplo:

$$(1\ 1)[\uparrow][(\lambda 1\ 1) \cdot \mathbf{id}]$$

Si se eliminara la sustitución $[\uparrow]$, queda la versión à la $\lambda\sigma$ de Ω. Notar que este término efectivamente es SN.

Por último, quedaría por establecer si con la definición de "basura" presentada en este trabajo se logra efectivamente la preservación de la normalización fuerte. Si así fuera, y suponiendo lo expuesto anteriormente, la relación entre

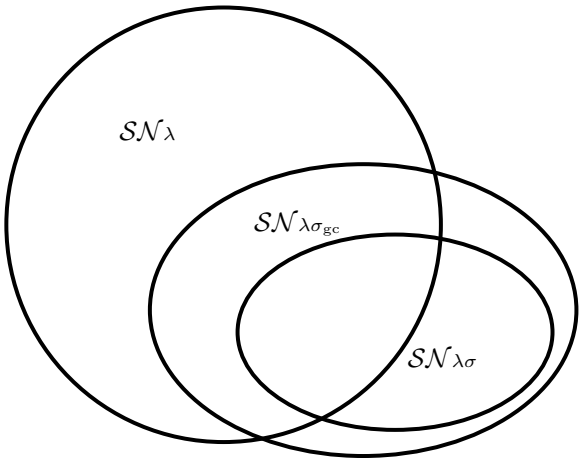


Figura 8.1: Relación hipotética entre los términos fuertemente normalizantes de λ, $\lambda\sigma$, y $\lambda\sigma_{\mathrm{gc}}$

los conjuntos de términos fuertemente normalizantes quedaría establecida como en la figura 8.2. Cabe mencionar aquí que antes del cierre de este trabajo se creía que $\lambda\sigma_{\mathrm{gc}}$ no cumplía PSN. Para ello se presentaba un término tipable con una supuesta derivación infinita. Para esta demostración se introdujo el sistema tipado (capítulo 7). En la construcción de dicha derivación infinita se encontró un error, que no fue posible subsanar. En el estudio en profundidad de las derivaciones de este término se observó que, aunque restrictiva, la definición de "basura" aquí presentada reduce considerablemente la cantidad de pasos de reducción. Las complicaciones que pueden surgir para establecer PSN son:

1. Si bien $\lambda\sigma_{\mathrm{gc}}$ toma ideas de λes, la traducción entre uno y otro no es inmediata (ver la diferencia entre la basura en uno y otro cálculo que se brinda en la sección 2.1). Esto no permite establecer PSN por medio de una traducción entre estos cálculos. Pruebas basadas en la traducción entre dos cálculos se muestran en, por ejemplo, [Kes07] y [KR95].

2. No tenemos conocimiento de un cálculo con sustituciones múltiples que cumpla esta propiedad, y habría que estudiar con cuidado si las pruebas realizadas para cálculos con sustituciones simples pueden ser extendidas para contemplar este caso. Por ejemplo, Bloo y Geuvers idean una técnica en [BG97] que permite establecer PSN para λx, λs, y λv, mediante la traducción a un cálculo con *labels*, pero sustituciones simples.

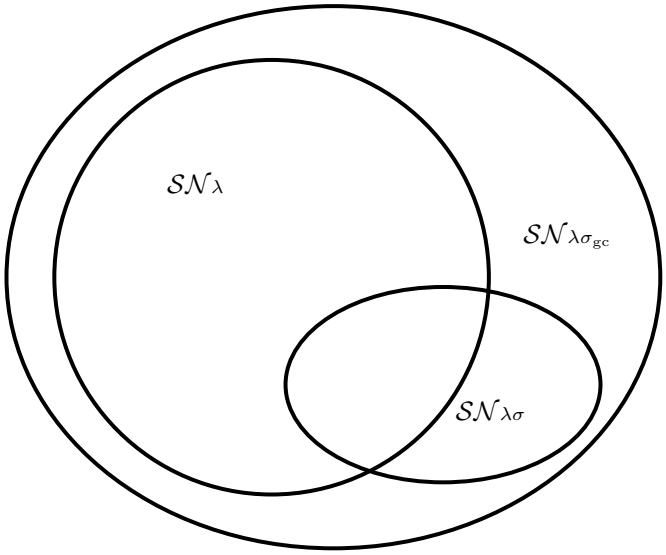


Figura 8.2: Relación hipotética entre los términos fuertemente normalizantes de λ, $\lambda\sigma$, y $\lambda\sigma_{\mathrm{gc}}$, si $\lambda\sigma_{\mathrm{gc}}$ preservara la normalización fuerte.

Capítulo 9

Conclusiones

En este trabajo hemos presentado un conjunto de herramientas teóricas para comprender y refinar el concepto de *basura* en un cálculo como $\lambda\sigma$.

Primero, en la subsección 1.4.3, se mostró una forma precisa de obtener las variables libres de un término. Luego, en la sección 2.1 se dió un caracterización a la basura, aunque se mostró que tal caracterización era limitada, en tanto que no toda substitución que es basura en λx es basura en $\lambda\sigma_{\mathrm{gc}}$. Aún así, se mostró en 8.1 que existen términos que no pertenecen a $\mathcal{SN}_{\lambda\sigma}$ pero sí a $\mathcal{SN}_{\lambda\sigma_{\mathrm{gc}}}$, demostrando la eficacia de colectar las substituciones basura.

Además, en el capítulo 4 se estableció una forma original de mostrar la confluencia, mostrando que si para toda variable libre i del término a vale $i[s] \twoheadrightarrow b_i \twoheadleftarrow i[t]$ para dos sustituciones s y t, entonces vale $a[s] \twoheadrightarrow c \twoheadleftarrow a[t]$.

A continuación presentamos distintos problemas que podrían constituir una extensión del presente trabajo.

9.1 Generalización de lemas a $\lambda\sigma$

En el capítulo 4 presentamos distintos lemas aplicables a $\lambda\sigma_{\mathrm{gc}}$. El estudio de las condiciones bajo las cuales estos lemas podrían generalizarse a $\lambda\sigma$ brindaría una valiosa base teórica para futuros trabajos relacionados con este cálculo.

Como ejemplo, el lema 75 podría ser útil en el estudio de la confluencia de extensiones a $\lambda\sigma$, tal como lo ha sido en $\lambda\sigma_{\mathrm{gc}}$ (capítulo 5).

9.2 Confluencia en términos puros

$\lambda\sigma_{\mathrm{gc}}$ no cumple la propiedad de confluencia en términos abiertos. Esto se demuestra con el mismo contraejemplo del teorema 34:

$$a[b[s] \cdot s] \twoheadleftarrow ((\lambda a)\ b)[s] \twoheadrightarrow a[b[s] \cdot (s \circ \mathbf{id})]$$

En $\lambda\sigma$, para cerrar este par crítico hay que agregar la regla

$$\text{(IdR)} \quad s \circ \mathbf{id} \quad \longrightarrow \quad s$$

que trae consigo nuevos pares críticos. En [Rio93] se presenta un cálculo denominado $\lambda\sigma_{\mathrm{SP}}$, que agrega la regla *IdR* y otras reglas derivadas de los pares

críticos que se van formando. Si bien no vale MC para términos y sustituciones con metavariables, sí vale MC para términos puros, es decir, con metavariables en los términos y no en las sustituciones.

En $\lambda\sigma_{\text{gc}}$, agregar esta regla no genera nuevos pares críticos que no se puedan cerrar, con lo que quedaría ver si el sistema $\lambda\sigma_{\text{gc}}+$ *IdR* es WCR en términos puros. Para ello es necesario determinar cuando una sustitución afecta o no un término, con lo que es necesario conocer de antemano las variables libres de los términos. Para ello, puede utilizarse la técnica utilizada en [Kes07], donde las metavariables son anotadas con un conjunto de variables, que representan las variables libres de la metavariable. Luego, restaría ver que los lemas utilizados en la prueba de WCR de σ_{gc} siguen valiendo para términos puros.

Cabe destacar que si se agregan los lemas 78 y 80 como reglas al cálculo, se generan infinitos nuevos pares críticos, con lo que no es una posibilidad.

9.3 Acercamiento a λes

Como se mencionó en la sección 2.1, otras variantes de la definición de "basura" pueden ser estudiadas, con el objetivo de obtener una versión similar a la definida por λes.

9.4 $\lambda\mathbf{s}_{\mathbf{gc}}$: Garbage collection para λs

En [KR95] se presenta un cálculo con sustituciones explícitas e índices de de Bruijn, que resulta de incorporar la metasustitución de λ_{DB} como reglas del cálculo, de igual forma que lo hace λx para λ.

Podría estudiarse si una versión con garbage collection permite una extensión que cumpla PSN y MC.

Para ello, podría extenderse el operador φ_j^i para que permita a i ser 0, para conseguir que $\varphi_0^0(a)$ sea reducir en 1 todos los índices de a. Luego, la regla GC quedaría como

$$(\text{GC}) \quad a\sigma^i b \quad \longrightarrow \quad \varphi_0^0(b) \quad (\text{si } i \notin \text{FV}(a))$$

Apéndice A

Demostraciones para $\lambda\sigma$

En este capítulo se demuestra el lema 122, necesario para demostrar el lema 47. Para ello, son necesarios algunos lemas adicionales.

Tomamos la definición de [Rio93]:

Definición 117. Sean $i \geq 0, n \geq 1$, se define $s_{i\,n}$ como

$$s_{i\,n} = 1 \cdot 2 \cdot \ldots \cdot i \cdot \uparrow^{i+n-1} \qquad s_{0\,n} = \uparrow^{n-1} \qquad s_{0\,1} = \mathbf{id}$$

Lema 118.

$$1 \cdot (s_{i\,n} \circ \uparrow) \twoheadrightarrow_\sigma s_{i+1\,n}$$

Demostración. La demostración es muy simple y se encuentra en [Rio93]. □

Lema 119. Para todo término a en σ-f.n., $a[s_{i\,1}] \twoheadrightarrow_\sigma a$. En particular, $a[\mathbf{id}] \twoheadrightarrow a$

Demostración. Por inducción en el término:

- $a = 1$:
  - $i = 0$: $1[\mathbf{id}] \ \to_{\text{VarId}} \ 1$
  - $i > 0$: $1[1 \cdot \ldots \cdot i \cdot \uparrow^i] \ \to_{\text{VarCons}} \ 1$
- $a = 1[\uparrow^k]$: $1[\uparrow^k][s_{i\,1}] \ \to_{\text{Clos}} \ 1[\uparrow^k \circ s_{i\,1}]$
  Separamos según k e i. Los detalles de estos pasos se pueden ver en [Rio93].
  - Si $k < i$,

    $1[\uparrow^k \circ s_{i\,1}] = 1[\uparrow^k \circ (1 \cdot \ldots \cdot i \cdot \uparrow^i)] \ \twoheadrightarrow_\sigma \ 1[k{+}1 \cdot \ldots \cdot i \cdot \uparrow^i] \ \to_{\text{VarCons}} \ k{+}1 = 1[\uparrow^k]$
  - Si $k \geq i$,

    $$1[\uparrow^k \circ s_{i\,1}] = 1[\uparrow^k \circ (1 \cdot \ldots \cdot i \cdot \uparrow^i)] \ \twoheadrightarrow_\sigma \ 1[\uparrow^{i+k-i}] = 1[\uparrow^k]$$
- $a = b\ c$: $(b\ c)[s_{i\,1}] \ \to_{\text{App}} \ b[s_{i\,1}]\ c[s_{i\,1}] \ \twoheadrightarrow_{\text{h.i.}} \ b\ c$
- $a = \lambda b$:

$$\begin{array}{rll} (\lambda b)[s_{i\,1}] & \to_{\text{Abs}} & \lambda b[1 \cdot (s_{i\,1} \circ \uparrow)] \\ & \twoheadrightarrow_{\text{L 118}} & \lambda b[s_{i+1\,1}] \\ & \twoheadrightarrow_{\text{h.i.}} & \lambda b \end{array}$$

□

Lema 120. Sea s una sustitución tal que $|s| = (m, n)$. Por lema 68, existen términos $a_1, \ldots, a_m$ tales que

$$\sigma(s) = a_1 \cdot \ldots \cdot a_m \cdot \uparrow^n \quad \vee \quad \sigma(s) = \mathbf{id}$$

Sea i una variable.

1. Si $i > m$, entonces $\sigma(i[s]) = i + n - m$.
2. Si $i \leq m$, entonces $\sigma(i[s]) = a_i$.

Demostración. Parte de esta prueba se encuentra en [Rio93], pero la repetimos aquí por claridad.

Por confluencia de σ se tiene que

$$\sigma(i[s]) = \sigma(\sigma(i)[\sigma(s)]) = \sigma(i[\sigma(s)])$$

Tomamos la σ-f. n. de s y separamos en casos:

- Si $\sigma(s) = \mathbf{id}$, por lema 119, $i[\mathbf{id}] \twoheadrightarrow_\sigma i$.
- Si $i = 1$,
  1. Si $1 > m$, entonces $m = 0$ y $\sigma(s) = \uparrow^n$. $1[\uparrow^n]$ está en forma normal, y corresponde a $1 + n - m$
  2. Si $1 \leq m$, entonces $i[a_1 \cdot \ldots \cdot a_m \cdot \uparrow^n] \ \to_{\text{VarCons}} \ a_1$ y está en σ-f.n.
- Si $i = 1[\uparrow^{i-1}]$

$$\begin{array}{lcl}
1[\uparrow^{i-1}][\sigma(s)] & \to_{\text{Clos}} & 1[\uparrow^{i-1} \circ \sigma(s)] \\
 & = & 1[(\uparrow \circ (\uparrow \circ (\ldots (\uparrow \circ \uparrow) \ldots))) \circ \sigma(s)] \\
 & \to_{\text{Ass}} & 1[\uparrow \circ ((\uparrow \circ (\ldots (\uparrow \circ \uparrow) \ldots)) \circ \sigma(s))] \\
 & \to_{\text{Ass}^{i-2}} & 1[\uparrow \circ (\uparrow \circ (\ldots (\uparrow \circ (\uparrow \circ \sigma(s))) \ldots))] \\
 & = & 1[\uparrow \circ (\uparrow \circ (\ldots (\uparrow \circ (\uparrow \circ (a_1 \cdot \ldots \cdot a_m \cdot \uparrow^n))) \ldots))] \\
1.(i > m) & \to_{\text{ShiftCons}^m} & 1[\uparrow^{i-1-m} \circ \uparrow^n] \\
 & \to_{\text{Ass}^{i-1-m}} & 1[\uparrow^{i+n-m-1}] = i + n - m \\
2.(i \leq m) & \to_{\text{ShiftCons}^{i-1}} & 1[a_i \cdot \ldots \cdot a_m \uparrow^n] \\
 & \to_{\text{VarShift}} & a_i
\end{array}$$

□

Lema 121. Sea a un término en σ-f. n., y sea s una sustitución. Si $(\forall i \in \text{FV}(a))\ \sigma(i[s]) = i$ entonces $a[s] \twoheadrightarrow a$.

Demostración. Por estar a en σ-f.n. basta con mostrar que $\sigma(a[s]) = a$.

Se demuestra por inducción en la σ-f. n. de a:

- Caso $a = i$: Por hipótesis vale.
- Caso $a = a_1\ a_2$: Como $\mathrm{FV}(a_i) \subseteq \mathrm{FV}(a)$ para $i \in \{1, 2\}$, vale por h.i. que $a_i[s] \twoheadrightarrow a_i$. Luego,

$$(a_1\ a_2)[s]\ \to_{\mathrm{App}}\ a_1[s]\ a_2[s]\ \twoheadrightarrow_{\mathrm{h.i.}}\ a_1\ a_2$$

- Caso $a = \lambda b$: Sea $s' = \sigma(s)$
  - Si $s' = \mathbf{id}$, entonces $(\lambda b)[s]\ \twoheadrightarrow\ (\lambda b)[\mathbf{id}]\ \twoheadrightarrow_{\mathrm{L\ 119}}\ \lambda b$
  - Si $s' = a_1 \cdot ... \cdot a_m \cdot \uparrow^n$, con $|s| = (m, n)$,

$$(\lambda b)[s]\ \twoheadrightarrow\ (\lambda b)[s']\ \to_{\mathrm{Abs}}\ \lambda b[1 \cdot (s' \circ \uparrow)]$$

    Luego,

$$\lambda b[1 \cdot ((a_1 \cdot ... \cdot a_m \cdot \uparrow^n) \circ \uparrow)]\quad \twoheadrightarrow_{\mathrm{Map}}\quad \lambda b[1 \cdot a_1[\uparrow] \cdot ... \cdot a_m[\uparrow] \cdot \uparrow^{n+1}]$$

    Sea $s'' = \sigma(1 \cdot a_1[\uparrow] \cdot ... \cdot a_m[\uparrow] \cdot \uparrow^{n+1}) = 1 \cdot a'_1 \cdot ... \cdot a'_m \cdot \uparrow^{n+1}$ para $a'_i = \sigma(a_i[\uparrow])$.
    Por hipótesis, sabemos que si $i \in \mathrm{FV}(a)$, $\sigma(i[s]) = i$. Por definición de FV, para $i > 1$, $i \in \mathrm{FV}(\lambda b) \Leftrightarrow i + 1 \in \mathrm{FV}(b)$. Luego puede ocurrir
    - Si $i + 1 \leq m + 1$, por lema 120

$$\sigma((i + 1)[1 \cdot (s' \circ \uparrow)]) = \sigma((i + 1)[s'']) = a'_i$$

      Por lema 120 nuevamente,

$$\sigma(i[s]) = i[a_1 \cdot ... \cdot a_m \cdot \uparrow^n] = a_i$$

      Por hipótesis, $\sigma(i[s]) = i$, con lo que $a_i = i$ (pues a_i está en σ-f.n. y σ es CR). Luego,

$$a'_i = \sigma(a_i[\uparrow]) = \sigma(i[\uparrow]) = i + 1$$

    - Si $i + 1 > m + 1$, por lema 120

$$\sigma((i + 1)[1 \cdot (s' \circ \uparrow)]) = i + 1 + (n + 1) - (m + 1)$$

      Como $i > m$, por lema 120 nuevamente,

$$\sigma(i[s]) = i + n - m$$

      Por hipótesis, $\sigma(i[s]) = i$, luego, por ser σ CR, $n = m$. Con lo que queda

$$\sigma((i + 1)[s'']) = i + 1 + (n + 1) - (m + 1) = i + 1$$

    En ambos casos, queda $\sigma((i + 1)[1 \cdot (s \circ \uparrow)]) = i + 1$.
    Si $1 \in \mathrm{FV}(b)$, $\sigma(1[1 \cdot (s \circ \uparrow)]) = 1$.
    Luego, $(\forall i \in \mathrm{FV}(b))\ \sigma(i[1 \cdot (s \circ \uparrow)]) = i$, con lo que se puede aplicar h. i. Luego,

$$\lambda b[1 \cdot (s \circ \uparrow)] \twoheadrightarrow \lambda b$$

□

Lema 122. Sean s una sustitución y a un término en σ-f.n.,

$$s \not\Rightarrow \lambda a \implies \lambda a[1 \cdot (s\circ \uparrow)] \twoheadrightarrow \lambda a$$

Demostración. Sean:

$$|s| = (m, n) \qquad t' = \sigma(1 \cdot (s\circ \uparrow))$$

Vamos a ver que

$$(\forall i \in \mathrm{FV}(a))\ \sigma(i[t']) = i$$

Para concluir por lema 121 que

$$\lambda a[1 \cdot (s\circ \uparrow)] \twoheadrightarrow \lambda a$$

Notar que $t' = 1 \cdot \sigma(s\circ \uparrow)$
Por definición de $\not\Rightarrow$, tenemos dos casos:

- $\mathrm{FV}(\lambda a) = \emptyset$: Como $\mathrm{FV}(\lambda a) = \mathrm{FV}(a) \setminus 1 = \emptyset$, se pueden dar dos casos:
  - $\mathrm{FV}(a) = \emptyset$: Se puede aplicar el lema 121 de forma trivial.
  - $\mathrm{FV}(a) = \{1\}$: Mostramos que se aplican las hipótesis del lema 121. Como
    $$1[t'] \ \to_{\mathrm{VarCons}}\ 1$$
    se tiene que
    $$(\forall i \in \mathrm{FV}(a))\ \sigma(i[t']) = i$$
- $\mathrm{FV}(\lambda a) \neq \emptyset$: Entonces $m = n$ y $(\forall i \in \mathrm{FV}(\lambda a))\ i > m$. Luego, como $|1 \cdot (s\circ \uparrow)| = (m+1, m+1)$, para toda variable $i \in \mathrm{FV}(a)$ tenemos

  $$\begin{array}{lll} \text{si } i > 1 & \implies & i > m+1 \ \wedge\ \sigma(i[t']) =_{\mathrm{L}\ 120} i \\ \text{si } i = 1 & \implies & 1[1 \cdot (s\circ \uparrow)] \ \to_{\mathrm{VarCons}}\ 1 \end{array}$$

  Luego,
  $$(\forall i \in \mathrm{FV}(a))\ \sigma(i[t']) = i$$

En todos los casos se puede aplicar el lema 121, luego

$$\lambda a[1 \cdot (s\circ \uparrow)] \twoheadrightarrow \lambda a$$

□

Apéndice B

Programa intérprete de $\lambda\sigma$ en Prolog

En este apéndice mostraremos un simple intérprete de $\lambda\sigma$ realizado en Prolog. El predicado `contraejemplo` busca ocurrencias del término $a = 1[\uparrow \circ \uparrow][1[\uparrow] \cdot \mathbf{id}]$ en todas las posibles derivaciones del término $a[s]$, con $s = 1 \cdot \uparrow$. Este predicado no se cumple, como muestra la siguiente salida de `Prolog`

```
?- [sigma].
% sigma compiled 0.01 sec, 736 bytes
Yes

?- contraejemplo.
No
```

De esta forma se concluye que no siempre para un término a y una sustitución s, $a[s] \twoheadrightarrow_\sigma a$.

```
a(sust(sust(one, clos(shift, shift)), cons(sust(one, shift), id))).

s(cons(one, shift)).

contraejemplo :- a(A), s(S), steps(sust(A, S), A).

step(app(A, B), app(C, B)) :- step(A, C).
step(app(A, B), app(A, C)) :- step(B, C).

step(abs(A), abs(B)) :- step(A, B).

step(sust(A,S), sust(B,S)) :- step(A,B).
step(sust(A,S), sust(A,T)) :- step(S,T).

step(cons(A, S), cons(B, S)) :- step(A,B).
step(cons(A, S), cons(A, T)) :- step(S,T).

step(clos(S, T), clos(U, T)) :- step(S, U).
step(clos(S, T), clos(S, U)) :- step(T, U).

% beta
step(app(abs(X), Y), sust(X, cons(Y, id))).
```

```
% varid
step(sust(one, id), one).

% varcons
step(sust(one, cons(A,_)), A).

% app
step(sust(app(A, B), S), app(sust(A,S), sust(B,S))).

% abs
step(sust(abs(A), S), abs(sust(A, cons(one, clos(S, shift))))).

% clos
step(sust(sust(A, S), T), sust(A, clos(S,T))).

% idl
step(clos(id, S), S).

% shiftid
step(clos(shift, id), shift).

% shiftcons
step(clos(shift, cons(_,S)), S).

% map
step(clos(cons(A,S), T), cons(sust(A,T), clos(S,T))).

% ass
step(clos(clos(S, T), U), clos(S, clos(T, U))).

steps(A, B) :- step(A, C), steps(C, B).
steps(A, B) :- step(A, B).
```

Listing B.1: Archivo sigma.pl

Apéndice C

Programa intérprete de $\lambda\sigma_{\mathbf{gc}}$ en Java

En este apéndice mostraremos un simple intérprete de $\lambda\sigma_{\text{gc}}$ realizado en Java.

El siguiente archivo muestra la ejecución para el contraejemplo de Mèllies:

```
package sigmagc;

import java.util.LinkedList;
import java.util.Queue;

public class RecursiveExecutor {

  private Sigmagc sigmagc;

  public RecursiveExecutor() {
    sigmagc = new Sigmagc();
  }

  public long execute(Term t) {
    long i = 0;
    Queue<Term> list = new LinkedList<Term>();
    sigmagc.addAllT(list , t);
    for (Term term : list) {
      i += execute(term) + 1;
    }
    return i;
  }

  public static void main(String[] args) {
    Term appId1 = Terms.app(Terms.idFunc(), Terms.one);
    Term appIdAppId1 = Terms.app(Terms.idFunc(), appId1);
    Term mellies = Terms.abs(Terms.app(Terms.abs(appIdAppId1),
        appId1.clone()));

    RecursiveExecutor lpe = new RecursiveExecutor();
    long i = lpe.execute(mellies);

    System.out.println("Mellies tiene " + i + " terminos");
  }
}
```

Listing C.1: Archivo RecursiveExecutor.java

El siguiente archivo contiene la lógica de ejecución de las reglas de $\lambda\sigma_{gc}$:

```
package sigmagc;

import java.util.HashSet;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Set;

public class Sigmagc {

  public boolean afecta(Term t, Sust s) {
    int[] peso = s.getPeso();

    Set<Integer> fv = fv(t);
    return fv.size() > 0 &&
     (peso[0] != peso[1] || existsLE(fv, peso[0]));
  }


  private boolean existsLE(Set<Integer> vars, int i) {
    for (Integer var : vars) {
      if (var <= i)
        return true;
    }
    return false;
  }

  public Set<Integer> fv(Term t) {
    if (t instanceof One) {
      HashSet<Integer> list = new HashSet<Integer>();
      list.add(1);
      return list;
    }
    else if (t instanceof Abs) {
      Abs a = (Abs) t;
      Set<Integer> list = fv(a.t);
      Set<Integer> newSet = new HashSet<Integer>(list.size());
      for (Integer var : list) {
        if (var > 1) {
          newSet.add(var-1);
        }
      }
      return newSet;
    }
    else if (t instanceof App) {
      App ap = (App) t;
      Set<Integer> fv1 = fv(ap.t1);
      Set<Integer> fv2 = fv(ap.t2);

      fv1.addAll(fv2);
      return fv1;
    }
    else if (t instanceof TSust) {
      TSust ts = (TSust) t;
      Set<Integer> fvt = fv(ts.t);
      Set<Integer> fvs = fv(ts.s);

      int[] p = ts.s.getPeso();
```

```
    for (Integer var : fvt) {
      if (var > p[0]) {
        fvs.add(var + p[1] - p[0]);
      }
    }
    return fvs;
  }
  else {
    throw new Error("no deberia suceder");
  }
}

public Set<Integer> fv(Sust s) {
  if (s instanceof Id) {
    return new HashSet<Integer>();
  }
  else if (s instanceof Shift) {
    return new HashSet<Integer>();
  }
  else if (s instanceof Cons) {
    Cons c = (Cons) s;
    Set<Integer> list = fv(c.t);
    list.addAll(fv(c.s));
    return list;
  }
  else if (s instanceof Clos) {
    Clos c = (Clos) s;
    Set<Integer> fvt = fv(c.s1);
    Set<Integer> fvs = fv(c.s2);

    int[] p = c.s2.getPeso();

    for (Integer var : fvt) {
      if (var > p[0]) {
        fvs.add(var + p[1] - p[0]);
      }
    }
    return fvs;
  }
  else {
    throw new Error("no deberia suceder");
  }
}


public void addAllT(Queue<Term> list, Term curTerm) {
  if (curTerm instanceof One) {
    return;
  }

  Term t1 = applyBeta(curTerm);
  if (t1 != null) {
    list.add(t1.clone());
  }

  Term t2 = applyGc(curTerm);
  if (t2 != null) {
    list.add(t2.clone());
  }

  Term t3 = applyVarCons(curTerm);
```

```
if (t3 != null) {
  list.add(t3.clone());
}

Term t4 = applyApp1(curTerm);
if (t4 != null) {
  list.add(t4.clone());
}

Term t5 = applyApp2(curTerm);
if (t5 != null) {
  list.add(t5.clone());
}

Term t6 = applyApp3(curTerm);
if (t6 != null) {
  list.add(t6.clone());
}

Term t7 = applyAbs(curTerm);
if (t7 != null) {
  list.add(t7.clone());
}

Term t8 = applyClos(curTerm);
if (t8 != null) {
  list.add(t8.clone());
}

Term t9 = applyClosGC(curTerm);
if (t9 != null) {
  list.add(t9.clone());
}

if (curTerm instanceof App) {
  App ap = (App) curTerm;
  LinkedList<Term> l1 = new LinkedList<Term>();
  LinkedList<Term> l2 = new LinkedList<Term>();
  addAllT(l1, ap.t1);
  addAllT(l2, ap.t2);
  for (Term st1 : l1) {
    list.add(Terms.app(st1, ap.t2));
  }
  for (Term st2 : l2) {
    list.add(Terms.app(ap.t1, st2));
  }
}
else if (curTerm instanceof Abs) {
  Abs ab = (Abs) curTerm;
  LinkedList<Term> l1 = new LinkedList<Term>();
  addAllT(l1, ab.t);
  for (Object st1 : l1) {
    list.add(Terms.abs((Term)st1));
  }
}
else if (curTerm instanceof TSust) {
  TSust ts = (TSust) curTerm;
  LinkedList<Term> l1 = new LinkedList<Term>();
  LinkedList<Sust> l2 = new LinkedList<Sust>();
  addAllT(l1, ts.t);
  addAllS(l2, ts.s);
  for (Term st1 : l1) {
```

```
        list.add(Terms.sust(st1, ts.s));
      }
      for (Sust st2 : l2) {
        list.add(Terms.sust(ts.t, st2));
      }
    }
  }

  public void addAllS(Queue<Sust> list, Sust curSust) {
    if (curSust instanceof Id || curSust instanceof Shift) {
      return;
    }

    Sust s1 = applyIdL(curSust);
    if (s1 != null) {
      list.add(s1.clone());
    }

    Sust s2 = applyShiftId(curSust);
    if (s2 != null) {
      list.add(s2.clone());
    }

    Sust s3 = applyShiftCons(curSust);
    if (s3 != null) {
      list.add(s3.clone());
    }

    Sust s4 = applyMap(curSust);
    if (s4 != null) {
      list.add(s4.clone());
    }

    Sust s5 = applyMapGc(curSust);
    if (s5 != null) {
      list.add(s5.clone());
    }

    Sust s6 = applyAss(curSust);
    if (s6 != null) {
      list.add(s6.clone());
    }

    if (curSust instanceof Cons) {
      Cons co = (Cons) curSust;
      LinkedList<Term> l1 = new LinkedList<Term>();
      LinkedList<Sust> l2 = new LinkedList<Sust>();
      addAllT(l1, co.t);
      addAllS(l2, co.s);
      for (Term st1 : l1) {
        list.add(Terms.cons(st1, co.s));
      }
      for (Sust st2 : l2) {
        list.add(Terms.cons(co.t, st2));
      }
    }
    else if (curSust instanceof Clos) {
      Clos cl = (Clos)curSust;
      LinkedList<Sust> l1 = new LinkedList<Sust>();
      LinkedList<Sust> l2 = new LinkedList<Sust>();
      addAllS(l1, cl.s1);
      addAllS(l2, cl.s2);
```

```
      for (Sust st1 : l1) {
        list.add(Terms.clos(st1, cl.s2));
      }
      for (Sust st2 : l2) {
        list.add(Terms.clos(cl.s1, st2));
      }
    }

  }

  public Term applyBeta(Term t) {
    if (!(t instanceof App)) {
      return null;
    }
    App ap = (App)t;
    if (!(ap.t1 instanceof Abs)) {
      return null;
    }
    Abs t1 = (Abs) ap.t1;
    Term t2 = ap.t2;
    Term tp = t1.t;
    Sust s = Terms.cons(t2, Terms.id);
    return Terms.sust(tp,s);
  }

  public Term applyGc(Term t) {
    if (!(t instanceof TSust)) {
      return null;
    }
    TSust ts = (TSust)t;
    if (afecta(ts.t, ts.s)) {
      return null;
    }
    return ts.t;
  }

  public Term applyVarCons(Term t) {
    if (!(t instanceof TSust)) {
      return null;
    }
    TSust ts = (TSust)t;
    if (!(ts.t instanceof One) || !(ts.s instanceof Cons)) {
      return null;
    }
    Cons co = (Cons) ts.s;
    return co.t;
  }

  public Term applyApp1(Term t) {
    if (!(t instanceof TSust)) {
      return null;
    }
    TSust ts = (TSust)t;
    if (!(ts.t instanceof App)) {
      return null;
    }
    App ap = (App)ts.t;
    if (!afecta(ap.t1, ts.s) || !afecta(ap.t2, ts.s)) {
      return null;
    }

    return Terms.app(Terms.sust(ap.t1, ts.s), Terms.sust(ap.t2, ts.
```

```
      s));
}

public Term applyApp2(Term t) {
  if (!(t instanceof TSust)) {
    return null;
  }
  TSust ts = (TSust)t;
  if (!(ts.t instanceof App)) {
    return null;
  }
  App ap = (App)ts.t;
  if (!afecta(ap.t1, ts.s) || afecta(ap.t2, ts.s)) {
    return null;
  }

  return Terms.app(Terms.sust(ap.t1, ts.s), ap.t2);
}

public Term applyApp3(Term t) {
  if (!(t instanceof TSust)) {
    return null;
  }
  TSust ts = (TSust)t;
  if (!(ts.t instanceof App)) {
    return null;
  }
  App ap = (App)ts.t;
  if (afecta(ap.t1, ts.s) || !afecta(ap.t2, ts.s)) {
    return null;
  }

  return Terms.app(ap.t1, Terms.sust(ap.t2, ts.s));
}

public Term applyAbs(Term t) {
  if (!(t instanceof TSust)) {
    return null;
  }
  TSust ts = (TSust)t;
  if (!(ts.t instanceof Abs)) {
    return null;
  }
  Abs ab = (Abs)ts.t;

  if (!afecta(ab, ts.s)) {
    return null;
  }
  return Terms.abs(Terms.sust(ab.t, Terms.cons(Terms.one, Terms.
      clos(ts.s, Terms.shift))));
}

public Term applyClos(Term t) {
  if (!(t instanceof TSust)) {
    return null;
  }
  TSust ts = (TSust)t;
  if (!(ts.t instanceof TSust)) {
    return null;
  }
  TSust innerTs = (TSust) ts.t;
  Term term = innerTs.t;
```

```
    if (!afecta(term, innerTs.s) || !afecta(ts.t, ts.s)) {
      return null;
    }

    Sust sust = Terms.clos(innerTs.s, ts.s);
    if (!afecta(term, sust)) {
      return null;
    }
    return Terms.sust(term, sust);
  }

  public Term applyClosGC(Term t) {
    if (!(t instanceof TSust)) {
      return null;
    }
    TSust ts = (TSust)t;
    if (!(ts.t instanceof TSust)) {
      return null;
    }
    TSust innerTs = (TSust) ts.t;
    Term term = innerTs.t;
    if (!afecta(term, innerTs.s) || !afecta(ts.t, ts.s)) {
      return null;
    }

    Sust sust = Terms.clos(innerTs.s, ts.s);
    if (afecta(term, sust)) {
      return null;
    }
    return term;
  }

  public Sust applyIdL(Sust s) {
    if (!(s instanceof Clos)) {
      return null;
    }
    Clos cl = (Clos) s;
    if (!(cl.s1 instanceof Id)) {
      return null;
    }
    return cl.s2;
  }

  public Sust applyShiftId(Sust s) {
    if (!(s instanceof Clos)) {
      return null;
    }
    Clos cl = (Clos) s;
    if (!(cl.s1 instanceof Shift)) {
      return null;
    }
    if (!(cl.s2 instanceof Id)) {
      return null;
    }
    return Terms.shift;
  }

  public Sust applyShiftCons(Sust s) {
    if (!(s instanceof Clos)) {
      return null;
    }
    Clos cl = (Clos) s;
```

```
    if (!(cl.s1 instanceof Shift)) {
      return null;
    }
    if (!(cl.s2 instanceof Cons)) {
      return null;
    }
    Cons co = (Cons) cl.s2;
    return co.s;
  }

  public Sust applyMap(Sust s) {
    if (!(s instanceof Clos)) {
      return null;
    }
    Clos cl = (Clos) s;
    if (!(cl.s1 instanceof Cons)) {
      return null;
    }
    Cons co = (Cons) cl.s1;
    if (!afecta(co.t, cl.s2)) {
      return null;
    }
    return Terms.cons(Terms.sust(co.t, cl.s2), Terms.clos(co.s, cl.
        s2));
  }

  public Sust applyMapGc(Sust s) {
    if (!(s instanceof Clos)) {
      return null;
    }
    Clos cl = (Clos) s;
    if (!(cl.s1 instanceof Cons)) {
      return null;
    }
    Cons co = (Cons) cl.s1;
    if (afecta(co.t, cl.s2)) {
      return null;
    }
    return Terms.cons(co.t, Terms.clos(co.s, cl.s2));
  }

  public Sust applyAss(Sust s) {
    if (!(s instanceof Clos)) {
      return null;
    }
    Clos cl = (Clos) s;
    if (!(cl.s1 instanceof Clos)) {
      return null;
    }
    Clos cl2 = (Clos) cl.s1;
    return Terms.clos(cl2.s1, Terms.clos(cl2.s2, cl.s2));
  }

}
```

Listing C.2: Archivo Sigmagc.java

Los archivos que se listan a continuación sontienen la estructura de términos y sustituciones:

```
package sigmagc;

import java.io.Serializable;
```

```
public abstract class Term  implements Serializable {
  public abstract Term clone();

  public abstract boolean equals(Object obj);

}
```

Listing C.3: Archivo Term.java

```
package sigmagc;


public class One extends Term {

  public Term clone() {
    return Terms.one;
  }

  public boolean equals(Object obj) {
    return obj instanceof One;
  }
}
```

Listing C.4: Archivo One.java

```
package sigmagc;

public class Abs extends Term {
  public Term t;

  public Abs(Term toAbs) {
    t = toAbs;
  }

  public Term clone() {
    return new Abs(t.clone());
  }

  public boolean equals(Object obj) {
    return obj instanceof Abs && t.equals(((Abs)obj).t);
  }
}
```

Listing C.5: Archivo Abs.java

```
package sigmagc;

public class App extends Term {
  public Term t1;
  public Term t2;

  public App(Term l, Term r) {
    t1 = l;
    t2 = r;
  }

  public Term clone() {
    return new App(t1.clone(), t2.clone());
  }
```

```
  public boolean equals(Object obj) {
    if (obj instanceof App) {
      App app = (App) obj;
      return t1.equals(app.t1) && t2.equals(app.t2);
    }
    return false;
  }
}
```

Listing C.6: Archivo App.java

```
package sigmagc;

public class TSust extends Term {
  public Term t;
  public Sust s;

  public TSust(Term t, Sust s) {
    this.t = t;
    this.s = s;
  }

  public Term clone() {
    return new TSust(t.clone(), s.clone());
  }

  public boolean equals(Object obj) {
    if (obj instanceof TSust) {
      TSust tsust = (TSust) obj;
      return t.equals(tsust.t) && s.equals(tsust.s);
    }
    return false;
  }
}
```

Listing C.7: Archivo TSust.java

```
package sigmagc;

import java.io.Serializable;

public abstract class Sust implements Serializable {

  protected int[] peso;

  public Sust() {
    peso = new int[2];
  }

  public abstract int[] getPeso();

  public abstract Sust clone();

  public abstract boolean equals(Object o);

}
```

Listing C.8: Archivo Sust.java

```
package sigmagc;
```

```
public class Id extends Sust {

  public Id() {
    peso[0] = 0;
    peso[1] = 0;
  }

  public int[] getPeso() {
    return peso;
  }

  public Sust clone() {
    return Terms.id;
  }

  public boolean equals(Object obj) {
    return obj instanceof Id;
  }
}
```

Listing C.9: Archivo Id.java

```
package sigmagc;

public class Shift extends Sust {

  public Shift() {
    peso[0] = 0;
    peso[1] = 1;
  }

  public int[] getPeso() {
    return peso;
  }

  public Sust clone() {
    return Terms.shift;
  }

  public boolean equals(Object obj) {
    return obj instanceof Shift;
  }

}
```

Listing C.10: Archivo Shift.java

```
package sigmagc;

public class Cons extends Sust {
  public Term t;
  public Sust s;

  public Cons(Term term, Sust sust) {
    t = term;
    s = sust;
  }

  public int[] getPeso() {
    int[] pesoS = s.getPeso();

    peso[0] = pesoS[0]+1;
```

```
    peso[1] = pesoS[1];
    return peso;
  }

  public Sust clone() {
    return new Cons(t.clone(), s.clone());
  }

  public boolean equals(Object obj) {
    if (obj instanceof Cons) {
      Cons cons = (Cons) obj;
      return t.equals(cons.t) && s.equals(cons.s);
    }
    return false;
  }
}
```

Listing C.11: Archivo Cons.java

```
package sigmagc;

public class Clos extends Sust {
  public Sust s1;
  public Sust s2;

  public Clos(Sust s1, Sust s2) {
    this.s1 = s1;
    this.s2 = s2;
  }

  public int[] getPeso() {
    int[] p1 = s1.getPeso();
    int[] p2 = s2.getPeso();

    if (p1[1] <= p2[0]) {
      peso[0] = p2[0] + p1[0] - p1[1];
      peso[1] = p2[1];
    }
    else {
      peso[0] = p1[0];
      peso[1] = p1[1] + p2[1] - p2[0];
    }

    return peso;
  }

  public Sust clone() {
    return new Clos(s1.clone(), s2.clone());
  }

  public boolean equals(Object obj) {
    if (obj instanceof Clos) {
      Clos clos = (Clos)obj;
      return s1.equals(clos.s1) && s2.equals(clos.s2);
    }
    return false;
  }
}
```

Listing C.12: Archivo Clos.java

El siguiente archivo contiene métodos de ayuda para construir términos.

```
package sigmagc;

public class Terms {
  public static Term one = new One();
  public static Sust shift = new Shift();
  public static Sust id = new Id();

  public static Term app(Term t1, Term t2) {
    return new App(t1, t2);
  }

  public static Term abs(Term t) {
    return new Abs(t);
  }

  public static Term sust(Term a, Sust s) {
    return new TSust(a, s);
  }

  public static Sust cons(Term a, Sust s) {
    return new Cons(a, s);
  }

  public static Sust clos(Sust s, Sust t) {
    return new Clos(s, t);
  }

  public static Sust oneid() {
    return cons(one, id);
  }

  public static Sust oneshift() {
    return cons(one, shift);
  }

  public static Term two() {
    return sust(one, shift);
  }

  public static Term idFunc() {
    return abs(Terms.one);
  }

  public static Term three() {
    return sust(one, clos(shift, shift));
  }
}
```

Listing C.13: Archivo Terms.java

Bibliografía

[ACCL91] Martín Abadi, Luca Cardelli, Pierre-L. Curien, and Jean-J. Levy. Explicit substitutions. *Journal of Functional Programming*, 1:31–46, 1991.

[Bar84] Hendrik P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North Holland, Amsterdam, 1984.

[Bar92] Hendrik P. Barendregt. Lambda calculi with types. *Handbook of logic in computer science*, 2:117–309, 1992.

[BG97] Roel Bloo and Herman Geuvers. Explicit substitution: on the edge of strong normalization. *Theoretical Computer Science*, 211, 1997.

[BK86] Jan A. Bergstra and Jan W. Klop. Conditional rewrite rules: Confluence and termination. *J. Comput. Syst. Sci.*, 32(3):323–362, 1986.

[BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, New York, NY, USA, 1998.

[BR95] Roel Bloo and Kristoffer H. Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *In CSN-95: Computer Science in the*, pages 62–72, 1995.

[CC87] Guy Cousineau and Pierre-L. Curien. The categorical abstract machine. *Sci. Comput. Program.*, 8(2):173–202, 1987.

[CHL96] Pierre-L. Curien, Thérèse Hardin, and Jean-J. Lévy. Confluence properties of weak and strong calculi of explicit substitutions. *Journal of the ACM*, 43(2):362–397, 1996.

[CHR92] Pierre-L. Curien, Thérèse Hardin, and Alejandro Ríos. Strong normalization of substitutions. In *MFCS '92: Proceedings of the 17th International Symposium on Mathematical Foundations of Computer Science*, pages 209–217, London, UK, 1992. Springer-Verlag.

[Cur94] Pierre-L. Curien. *Categorical combinators, sequential algorithms, and functional programming (2nd ed.)*. Birkhauser Boston Inc., Cambridge, MA, USA, 1994.

[dB72] Nicolaas G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae*, 34:381–392, 1972.

[DG01] René David and Bruno Guillaume. A λ-calculus with explicit weakening and explicit substitution. *Mathematical. Structures in Comp. Sci.*, 11(1):169–206, 2001.

[GTL89] Jean-Y. Girard, Paul Taylor, and Yves Lafont. *Proofs and types.* Cambridge University Press, New York, NY, USA, 1989.

[Har92] Thérèse Hardin. From categorical combinators to lambda-sygma-calculi, a quest for confluence. Research Report RR-1777, INRIA, 1992. Projet CHLOE.

[Kes07] Delia Kesner. *Lecture Notes in Computer Science*, chapter The theory of calculi with explicit substitutions revisited, pages 238–252. Springer Berlin / Heidelberg, 2007.

[KL05] Delia Kesner and Stephane Lengrand. *Lecture Notes in Computer Science*, chapter Extending the Explicit Substitution Paradigm, pages 407–422. Springer Berlin / Heidelberg, 2005.

[KR95] Fairouz Kamareddine and Alejandro Ríos. A lambda-calculus à la de bruijn with explicit substitutions. In *PLILPS '95: Proceedings of the 7th International Symposium on Programming Languages: Implementations, Logics and Programs*, pages 45–62, London, UK, 1995. Springer-Verlag.

[KR97] Fairouz Kamareddine and Alejandro Ríos. Extending a λ-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms. *Journal of Functional Programming*, 7(4):395–420, 1997.

[Les94] Pierre Lescanne. From $\lambda\sigma$ to $\lambda\upsilon$: a journey through calculi of explicit substitutions. In *POPL '94: Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 60–69, New York, NY, USA, 1994. ACM.

[Mel95] Paul-A. Melliès. Typed lambda-calculi with explicit substitutions may not terminate. In *TLCA '95: Proceedings of the Second International Conference on Typed Lambda Calculi and Applications*, pages 328–334, London, UK, 1995. Springer-Verlag.

[Mun96] César A. Munoz. Confluence and preservation of strong normalisation in an explicit substitutions calculus. In *LICS '96: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, page 440, Washington, DC, USA, 1996. IEEE Computer Society.

[Rio93] Alejandro Rios. *Contributions à l'étude des Lambda-calculus avec Substitutions Explicites.* PhD thesis, lÚniversité Paris VII, 1993.

[Ros93] Kristoffer H. Rose. Explicit cyclic substitutions. In *CTRS '92: Proceedings of the Third International Workshop on Conditional Term Rewriting Systems*, pages 36–50, London, UK, 1993. Springer-Verlag.

[Ter03] Terese. *Term Rewriting Systems.* Cambridge University Press, 2003.

[VARK07] Daniel L. Ventura, Mauricio Ayala-Rincón, and Fairouz Kamareddine. Explicit substitutions calculi with explicit eta rules. En preparación, 2007.

[Zan94] Hans Zantema. Termination of term rewriting: Interpretation and type elimination. *Journal of Symbolic Computation*, 17:23–50, 1994.

[Zan00] Hans Zantema. Termination of term rewriting. Technical Report UU-CS-2000-04, Department of Information and Computing Sciences, Utrecht University, 2000.