Universidade do Minho
Escola de Ciências

Filipa Simões Mendes

Variações sobre o cálculo-lambda *call-by-value*

outubro de 2022

Filipa Mendes **Variações sobre o cálculo-lambda *call-by-value***

UMinho | 2022

Universidade do Minho
Escola de Ciências

Filipa Simões Mendes

Variações sobre o cálculo-lambda *call-by-value*

Dissertação de Mestrado
Mestrado em Matemática e Computação

Trabalho efetuado sob a orientação do
Professor Doutor José Carlos Soares do Espírito Santo

outubro de 2022

Agradecimentos

Não conseguiria realizar esta dissertação sem o contributo de várias pessoas que fizeram parte deste percurso e, acima de tudo, acreditaram em mim. Desta forma, quero agradecer em especial:

Ao meu Professor e Orientador José Carlos Espiríto Santo, pela orientação atenta, pelo conhecimento partilhado e, principalmente, por me ter motivado e incentivado sempre que me senti mais perdida. Agradeço-lhe ainda por me ter dado a conhecer a Lógica, área pela qual me entusiasmei, na minha licenciatura em Matemática.

À minha amiga e colega de mestrado Bruna, por ter vivido toda esta aventura comigo. Aos meus amigos Catarina, Sofia e Luís, por todo o amor e loucura que partilham comigo, nem que seja de longe a longe. Agradeço ainda à Catarina pela paciência, pelos conselhos e porque soube sempre o que dizer quando mais precisei, durante a realização deste trabalho.

Aos meus queridos Zé e Gabi, pelas noites de frisbee, jogos e gargalhadas bem passadas, que me tornam um pouco mais sã no meio de tantas abstrações, pelo apoio e pela compreensão da minha falta de disponibilidade.

Ao Samuel, por todas as conversas de incentivo e pelo interesse no meu trabalho.

Ao meu namorado, Hugo, pela companhia nas horas de estudo, pela paciência em ouvir-me falar do meu trabalho mesmo não percebendo, por me acolher nos momentos mais difíceis desta jornada, e, principalmente, por partilhar os mesmos sonhos que eu, incentivando-me sempre a lutar pelos meus objetivos.

Por fim, aos meus pais e irmã, por serem os meus maiores apoiantes e por acreditarem sempre em mim e nos meus sonhos.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Resumo

Variações sobre o cálculo-lambda *call-by-value*

Plotkin apresentou em 1975 a primeira proposta para o cálculo-lambda *call-by-value*, um sistema formal que modela uma linguagem de programação *call-by-value*, e vários outros sistemas têm sidos propostos desde então. No entanto, este sistema é incompleto para a semântica CPS (*continuation-passing-style*), ou seja, existem termos com a mesma semântica que não podem ser provados iguais no cálculo de Plotkin.

Nesta dissertação estudamos a extensão de Sabry-Felleisen, o cálculo computacional de Moggi e o cálculo-λ_Q de Dyckhoff-Lengrand, três sistemas formais com origens diferentes, que são uma resposta ao problema da incompletude do cálculo de Plotkin relativamente à semântica CPS.

Provamos que existe uma correspondência equacional entre a extensão de Sabry-Felleisen e o cálculo computacional. Propomos uma simplificação do cálculo-λ_Q e identificamos o núcleo correspondente. Por fim, provamos que existe uma correspondência equacional entre o núcleo do cálculo-λ_Q simplificado e o núcleo do cálculo computacional.

Palavras-chave: cálculo-lambda, *call-by-value*, *call-by-name*, *continuation-passing-style*, linguagens de programação funcionais, cálculo de sequentes, lógica intuicionista.

Abstract

Variations about the call-by-value lambda-calculus

In 1975 Plotkin presented the very first call-by-value lambda-calculus, a formal system that models a call-by-value programming language. Since then, many other systems has been presented. Nevertheless, this system turned out to be incomplete for the CPS (continuation-passing-style) semantics. In other words, there are terms with the same semantic that are not proved equal in Plotkin's lambda-calculus.

In this dissertation, we study Sabry-Felleisen's extension of Plotkin's calculus, Moggi's computational lambda-calculus and Dyckhoff-Lengrand's λ_Q-calculus, three formal systems with different origins, that are an answer to Plotkin's incompleteness problem with respect to CPS semantics.

We prove that there is an equational correspondence between Sabry-Felleisen's extension and the computational calculus. We present a simplified λ_Q-calculus and we identify its kernel. Finally, we prove that there is an equational correspondence between the kernel of the simplified λ_Q-calculus and the kernel of the computational lambda-calculus.

Keywords: lambda-calculus, call-by-value, call-by-name, continuation-passing-style, funcional programming languages, sequent calculus, intuicionistic logic.

Índice

Lista de Figuras **x**

Siglas **xi**

1 Introdução **1**

2 Recapitulação **3**

2.1 Cálculo-λ . . . 3

2.1.1 Sintaxe . . . 3

2.1.2 Redução . . . 5

2.1.3 Confluência . . . 7

2.2 Conceitos sobre relações de redução . . . 14

2.3 Cálculo-λ com tipos simples . . . 15

2.4 Sistemas dedutivos . . . 16

2.4.1 Dedução natural e isomorfismo de Curry-Howard . . . 16

2.4.2 Cálculo de sequentes . . . 18

3 λ-calculi e linguagens de programação **21**

3.1 Cálculo-λ e linguagem CBN . . . 21

3.2 Cálculo-λ_v de Plotkin e linguagem CBV . . . 23

3.3 CBN vs CBV . . . 28

3.3.1 *Call-by-value* CPS . . . 28

3.3.2 *Call-by-name Thunks* . . . 29

3.4 Extensão de Sabry-Felleisen . . . 30

4 Cálculo computacional **35**

4.1 Motivação 35
4.2 O sistema 35
4.3 Relação com a extensão de Sabry-Felleisen 39

5 Cálculo λ_Q **51**
5.1 Motivação 51
5.2 O sistema 53
5.2.1 Versão original 53
5.2.2 Versão simplificada 54
5.3 Relação com o cálculo computacional 56
5.3.1 Correspondência equacional entre núcleos 58
5.3.2 Confluência dos núcleos 78

6 Conclusão **91**

Bibliografia **93**

Lista de Figuras

1 Árvore de sequentes 17

Siglas

CBN	*Call-by-name*
CBV	*Call-by-value*
CPS	*Continuation-passing-style*
LJ	Cálculo de sequentes para a lógica intuicionista
LJQ	Cálculo de sequentes focado para a lógica intuicionista
NJ	Dedução natural com sequentes para a lógica intuicionista

Capítulo 1

Introdução

Na década de 1930, surge o cálculo-lambda, um sistema formal que estuda propriedades de funções independentemente do seu significado, inventado por A. Church [4]. Tanto na Matemática como na Ciência da Computação, foi grande o impacto desta invenção [3]. Através do cálculo-lambda, Church desenvolveu uma representação dos números naturais, conhecida por numerais de Church, e com ela surge, pela primeira vez, a noção de computabilidade. Church apresentou ainda a ideia de que todas as funções intuitivamente computáveis são funções lambda-definíveis, isto é, definíveis através do cálculo-lambda. Esta hipótese, que não pode ser matematicamente provada, é conhecida como a Tese de Church e, até hoje, ainda não foi refutada.

Em 1964, P. Landin utiliza, pela primeira vez, o cálculo-lambda para estudar linguagens funcionais, que podem ser *call-by-value* (CBV) ou *call-by-name* (CBN), diferindo na chamada de funções [12]. Enquanto que numa linguagem de programação *call-by-name* as funções podem ser chamadas por qualquer argumento, numa linguagem de programação *call-by-value* as funções são somente chamadas por valores. Em 1975, Plotkin apresenta a primeira proposta para o cálculo-lambda *call-by-value*, conhecido como cálculo de Plotkin, mostrando ainda como este modela uma linguagem de programação *call-by-value*. Analogamente, Plotkin mostra como o cálculo-lambda usual modela uma linguagem de programação *call-by-name* [14]. Porém, o cálculo de Plotkin é incompleto para a semântica CPS (*Continuation-Passing-Style*). Esta é uma semântica utilizada nas linguagens de programação que, tal como o nome indica, recorre a continuações.

O objetivo desta dissertação é estudar o cálculo de Plotkin, bem como vários outros sistemas que têm sido propostos em alternativa desde então e que visam ser uma resposta para o problema da incompletude do cálculo de Plotkin. Vamos estudar três desses sistemas: a extensão de Sabry-Felleisen, que resulta de

identificar e adicionar diretamente ao cálculo de Plotkin as reduções em falta, o cálculo computacional, que tem origem na semântica das linguagens de programação, e o cálculo-λ_Q, que resulta de aplicar o isomorfismo de Curry-Howard a um sistema lógico para a lógica intuicionista, o cálculo de sequentes "focado" LJQ, pelo que tem origem na teoria da demonstração. Este estudo incide na relação que estes sistemas apresentam com a semântica CPS, isto é, se são corretos e completos para a semântica CPS, e na relação que apresentam entre si.

Este trabalho é constituído por 6 capítulos. No segundo e próximo capítulo, recordamos o cálculo-lambda, sem tipos e com tipos simples, e apresentamos a dedução natural com sequentes e o cálculo de sequentes, ambos para a lógica intuicionista, relacionando estes dois sistemas dedutivos com o cálculo-lambda com tipos simples, através do isomorfismo de Curry-Howard. Além disso, ainda neste capítulo, introduzimos alguns conceitos importantes, que permitem estabelecer relações entre diferentes cálculos. No terceiro capítulo, estudamos como o cálculo-lambda e o cálculo de Plotkin determinam linguagens de programação *call-by-name* e *call-by-value*, respetivamente, e como estas se relacionam entre si. No fim do terceiro capítulo, apresentamos a extensão ao cálculo de Plotkin desenvolvida por Sabry-Felleisen. No quarto capítulo, estudamos o cálculo computacional de Moggi e comparamos este cálculo com a extensão de Sabry-Felleisen, provando que formam uma equivalência equacional. No quinto capítulo, definimos uma versão simplificada do cálculo-λ_Q de Dyckhoff-Lengrand e estudamos a sua relação com o cálculo computacional, mostrando que os seus núcleos formam uma correspondência equacional. Finalmente, no último capítulo, refletimos sobre o trabalho que foi desenvolvido, salientando eventuais contributos originais bem como temas de interesse para estudar no futuro.

Capítulo 2

Recapitulação

2.1 Cálculo-λ

2.1.1 Sintaxe

Definição 1 (**termos-λ**). *Seja $\mathcal{A} = \mathcal{V} \cup \{(,), \lambda\}$, onde $\mathcal{V}$ é um conjunto numerável de variáveis, denotadas por x, y, z, etc. O conjunto de todas as palavras em $\mathcal{A}$ é representado por $\mathcal{A}^*$. O conjunto dos termos-λ, representado por Λ, é definido indutivamente sobre $\mathcal{A}^*$ por:*

$$M, N, P, Q \in \Lambda ::= x \mid (\lambda x M) \mid (MN)$$

Ao termo-λ da forma $(\lambda x M)$ chama-se abstração e diz-se que x é a variável e M o corpo da abstração. Ao termo-λ da forma (MN) chama-se aplicação e diz-se que M e N estão na posição de função e de argumento, respetivamente.

Alguns exemplos de termos-λ são: x, (xy), $(\lambda x(yz))$, $((\lambda xx)(\lambda zz))$ e λxy.

Por convenção, com o objetivo de simplificar a notação: (i) omite-se os parêntesis mais externos; (ii) $\lambda xy \ldots z.M$ abrevia $\lambda x(\lambda y(\cdots(\lambda zM)\cdots))$; (iii) a associação é feita à esquerda.

Definição 2 (**Ocorrências ligadas e livres**). *Uma ocorrência de uma variável x num termo-λ diz-se ligada se estiver no corpo de uma abstração $\lambda x.M$. Caso contrário, diz-se livre.*

No termo-λ $(\lambda x.xy)(xz)$, por exemplo, a primeira ocorrência de x é ligada e a segunda é livre, enquanto que as ocorrências de y e z são livres.

Definição 3 (**Variáveis livres**). *O conjunto das variáveis com ocorrências livres em M, notado por $LIV(M)$, é definido recursivamente por:*

$$LIV(x) = x$$
$$LIV(\lambda x.M) = LIV(M) - \{x\}$$
$$LIV(MN) = LIV(M) \cup LIV(N)$$

Diz-se que M é um termo-λ fechado ou um combinador se $LIV(M) = \{\}$, ou seja, se M não tiver ocorrências livres de variáveis. Diz-se também que um termo-λ da forma $\lambda x.M$ é uma abstração vacuosa se $x \notin LIV(M)$.

Tomemos, como exemplos, os termos-λ $M = \lambda x.x$ e $N = \lambda x.yz$. M é um combinador e N é uma abstração vacuosa. Em particular, M é o combinador $\mathbf{I}$.

Definição 4 (**Subtermos**). *O conjunto dos subtermos de M, representado por $SUB(M)$, é definido recursivamente por:*

$$SUB(x) = x$$
$$SUB(\lambda x.M) = \{\lambda x.M\} \cup SUB(M)$$
$$SUB(MN) = \{MN\} \cup SUB(M) \cup SUB(N)$$

O termo-λ $M = (x(\lambda z.xy))y$, por exemplo, tem como subtermos x, y, xy, $\lambda z.xy$, $x(\lambda z.xy)$ e M. É de notar que $z \notin SUB(M)$.

Adotando a convenção de variáveis Barendregt, quando se considerar termos-λ, assume-se que os nomes das variáveis ligadas são distintos dos nomes das variáveis livres [2]. Desta forma, em particular, o problema da captura de variáveis desaparece.

Definição 5 (**Substituição**). *A substituição das ocorrências livres de x por N em M, representada por $[N/x]M$, é definida recursivamente por:*

$$[N/x]x = N$$
$$[N/x]y = y, \text{ se } y \neq x$$
$$[N/x](\lambda y.P) = \lambda y.[N/x]P$$
$$[N/x](PQ) = [N/x]P[N/x]Q$$

2.1.2 Redução

O conceito de redução em Λ é, por definição, uma relação binária $R \subseteq \Lambda^2$. Além disso, se R_1 e R_2 são dois conceitos de reduções, então $R_1 R_2 = R_1 \cup R_2$.

Definição 6 (**Regra de Inferência**). *Uma regra de inferência tem a forma:*

$$\frac{(N, N') \in R \quad \ldots \quad (P, P') \in R}{(M, M') \in R}\ RI$$

RI representa o nome da regra aplicada, $(N, N') \in R, \ldots, (P, P') \in R$ representam as hipóteses e $(M, M') \in R$ representa a conclusão.

As regras que vamos apresentar de seguida têm a nomenclatura utilizada em [11].

Definição 7 (**Relação compatível**). *Uma relação binária $R \subseteq \Lambda^2$ diz-se compatível se:*

$$\frac{(M, N) \in R}{(\lambda x.M, \lambda x.N) \in R}\ \xi \qquad \frac{(M, N) \in R}{(MP, NP) \in R}\ \nu \qquad \frac{(M, N) \in R}{(PM, PN) \in R}\ \mu$$

Definição 8 (**Fecho compatível**). *Dada uma relação binária $R \subseteq \Lambda^2$, o seu fecho compatível, denotado por $\to_R$, é definido indutivamente por:*

$$\frac{(M, N) \in R}{M \to_R N}\ \to_R \qquad \frac{M \to_R N}{\lambda x.M \to_R \lambda x.N}\ \xi \qquad \frac{M \to_R N}{MP \to_R NP}\ \nu \qquad \frac{M \to_R N}{PM \to_R PN}\ \mu$$

Esta relação designa-se relação de redução-R em um passo e é a menor relação compatível que contém R.

Definição 9 (**Fecho reflexivo e transitivo**). *Dada $\to_R \subseteq \Lambda^2$, o seu fecho reflexivo e transitivo, denotado por $\twoheadrightarrow_R$, é definido indutivamente por:*

$$\frac{M \to_R N}{M \twoheadrightarrow_R N}\ \twoheadrightarrow_R \qquad \frac{M \in \Lambda}{M \twoheadrightarrow_R M}\ \rho \qquad \frac{M \twoheadrightarrow_R N \quad N \twoheadrightarrow_R P}{M \twoheadrightarrow_R P}\ \tau$$

Designa-se esta relação por redução R em zero, um ou mais passos ou apenas redução-R. Para cada $n \in \mathbb{N}$, pode definir-se recursivamente a relação de redução-R em n passos e representa-se por $\to_R^n$.

Definição 10 (**Fecho de equivalência**). *Dada $\to_R \subseteq \Lambda^2$, o seu fecho congruente, denotado por $=_R$, é definido indutivamente por:*

$$\frac{M \to_R N}{M =_R N}\ =_R \qquad \frac{M \in \Lambda}{M =_R M}\ \rho \qquad \frac{M =_R N \quad N =_R P}{M =_R P}\ \tau \qquad \frac{M =_R N}{N =_R M}\ \sigma$$

A esta relação damos o nome de igualdade-R.

Definição 11 (**Relação** α)**.** *Seja $\alpha \subseteq \Lambda^2$ a relação apresentada sobre a forma da seguinte regra:*

$$(\alpha) \qquad \lambda x.M \to \lambda y.[y/x]M, \text{ se } y \neq x \text{ e } y \notin LIV(M)$$

Dois termos-λ M e N dizem-se iguais-α se diferem apenas no nome das variáveis das abstrações.

Tome-se, como exemplo, $M = \lambda x.xy$ e $N = \lambda z.zy$. Com efeito, $\lambda x.xy =_\alpha \lambda z.[z/x](xy) = \lambda z.zy$. Assumiremos que dois termos iguais-α são iguais.

Definição 12 (**Relação** β)**.** *Seja $\beta \subseteq \Lambda^2$ a relação apresentada sobre a forma da seguinte regra:*

$$(\beta) \qquad (\lambda x.M)N \to [N/x]M$$

Vejamos que $(\lambda xy.(xy)x)MN \twoheadrightarrow_\beta MNM$ em 2 passos, como exemplo. Ora,

$$\begin{aligned}(\lambda xy.(xy)x)MN &\to_\beta [M/x](\lambda y.(xy)x)N \\ &= (\lambda y.([M/x]x[M/x]y)[M/x]x)N \\ &= (\lambda y.(My)M)N \\ &\to_\beta [N/y]((My)M) \\ &= ([N/y]M[N/y]y)[N/y]M \\ &= MNM.\end{aligned}$$

É de notar que $[N/y]M = M$, uma vez que convencionamos que os nomes das variáveis ligadas e livres são distintos, pelo que y não ocorre livre em M.

Consideremos agora os combinadores $\mathbf{I}$, $\mathbf{K} = \lambda xy.x$, $\mathbf{S} = \lambda xyz.xz(yz)$ e $\mathbf{\Omega} = (\lambda x.xx)(\lambda x.xx)$. Temos que $\mathbf{I}x \to_\beta x$ e $\mathbf{SKK}x \twoheadrightarrow_\beta \mathbf{K}x(\mathbf{K}x) \to_\beta x$. Para qualquer $n \in \mathbb{N}$, $\mathbf{\Omega} \to_\beta^n \mathbf{\Omega}$. Com efeito, $\mathbf{\Omega} = (\lambda x.xx)(\lambda x.xx) \to_\beta [(\lambda x.xx)/x](xx) = \mathbf{\Omega}$, permanecendo assim num *loop* de reduções β.

Definição 13 (**redex-R e reduto**)**.** *A um termo-λ M tal que $(M, N) \in R$, para algum $N \in \Lambda$, dá-se o nome de redex-R. Neste caso, N diz-se reduto ou contractum-R.*

Definição 14 (**Forma normal-R**)**.** *Um termo-λ M diz-se uma forma normal-R, abreviadamente fn-R, se nenhum dos subtermos de M é um redex-R. Um termo-λ N diz-se uma forma normal-R de um termo-λ M se N é uma forma normal-R e $M \twoheadrightarrow_R N$.*

Consideremos o termo-λ $M = (\lambda x.(\lambda y.yx)z)v$. Podemos reduzi-lo de duas formas possíveis. Se reduzirmos primeiramente o redex-β exterior, $M = (\lambda x.(\lambda y.yx)z)v \to_\beta (\lambda y.yv)z \to_\beta zv$. Ora, zv é fn-β e $M \twoheadrightarrow_\beta zv$. Logo, zv é fn-β de M. Se reduzirmos em primeiro lugar o redex-β interior, $M = (\lambda x.(\lambda y.yx)z)v \to_\beta (\lambda x.zx)v \to_\beta zv$. Mais uma vez, zv é fn-β de M.

De seguida, verificamos que uma forma normal-β de um termo-λ é única.

2.1.3 Confluência

Nesta secção vamos provar que a relação β é confluente. Esta prova foi desenvolvido por Tait and Martin-Löf e pode ser facilmente aplicada a outros cálculos.

Definição 15 (**Propriedade do diamante**). *Uma relação binária $R \subseteq \Lambda^2$ satisfaz a propriedade do diamante, representando por $R \models \diamond$, se, para quaisquer $M, N, N' \in \Lambda$,*

$$(M,N) \in R \wedge (M,N') \in R \implies \exists P \in \Lambda : (N,P) \in R \wedge (N',P) \in R.$$

Definição 16 (**Confluência**). *Uma relação binária $R \subseteq \Lambda^2$ diz-se confluente se o seu fecho reflexivo e transitivo $\twoheadrightarrow_R$ satisfaz a propriedade do diamante.*

Lema 1. *Seja $R \subseteq \Lambda^2$ uma relação binária e R^* o seu fecho reflexivo e transitivo. Então,*

$$R \models \diamond \implies R^* \models \diamond.$$

Demonstração. A prova segue por indução em $(M,N) \in R^*$. É necessário ver que, para todo $(M,N') \in R^*$, $\exists P \in \Lambda$ tal que $(N,P) \in R^*$ e $(N',P) \in R^*$. □

Pretendemos mostrar que β é confluente, ou seja, que $\twoheadrightarrow_\beta$ satisfaz a propriedade do diamante. Com esse objetivo, definimos a relação binária $\twoheadrightarrow_1 \in \Lambda^2$ tal que: (i) $\twoheadrightarrow_1 \models \diamond$; (ii) o fecho transitivo de $\twoheadrightarrow_1$ é $\twoheadrightarrow_\beta$.

Definição 17 (**Relação de redução β paralela**). *Seja $\twoheadrightarrow_1 \subseteq \Lambda^2$ a relação binária definida indutivamente por:*

$$\frac{M \in \Lambda}{M \twoheadrightarrow_1 M}\ \rho \qquad \frac{M \twoheadrightarrow_1 M'}{\lambda x.M \twoheadrightarrow_1 \lambda x.M'}\ \xi \qquad \frac{M \twoheadrightarrow_1 M' \quad N \twoheadrightarrow_1 N'}{MN \twoheadrightarrow_1 M'N'}\ \epsilon$$

$$\frac{M \twoheadrightarrow_1 M' \quad N \twoheadrightarrow_1 N'}{(\lambda x.M)N \twoheadrightarrow_1 [N'/x]M'}\ \beta$$

Consideremos o termo-λ $M = (\lambda x.(\lambda y.y)x)((\lambda z.v)u)$. Ora,

$$\dfrac{\dfrac{\dfrac{y \in \Lambda}{y \twoheadrightarrow_1 y}\,\rho \quad \dfrac{x \in \Lambda}{x \twoheadrightarrow_1 x}\,\rho}{(\lambda y.y)x \twoheadrightarrow_1 [x/y]y = x}\,\beta \quad \dfrac{\dfrac{v \in \Lambda}{v \twoheadrightarrow_1 v}\,\rho \quad \dfrac{u \in \Lambda}{u \twoheadrightarrow_1 u}\,\rho}{(\lambda z.v)u \twoheadrightarrow_1 [u/z]v = v}\,\beta}{M = (\lambda x.(\lambda y.y)x)((\lambda z.v)u) \twoheadrightarrow_1 [v/x]x = v}\,\beta$$

Assim, $M \twoheadrightarrow_1 v$. É de notar que $M \twoheadrightarrow_\beta v$ em 3 passos, mas não é verdade que $M \rightarrow_\beta v$. Com efeito, $M \rightarrow_\beta (\lambda y.y)((\lambda z.v)u) \rightarrow_\beta (\lambda z.v)u \rightarrow_\beta v$.

Lema 2. *Sejam $x, y, P, N, M \in \Lambda$ tais que $x \neq y$ e $x \notin LIV(P)$. Então,*

$$[P/y]([N/x]M) = [([P/y]N)/x]([P/y]M).$$

Demonstração. A prova segue por indução em M.

- Caso $M = x$:

  Ora, $[P/y]([N/x]x) = [P/y]N = [[P/y]N/x]x = [[P/y]N/x]([P/y]x)$.

- Caso $M = y$:

  Ora, $[P/y]([N/x]y) = [P/y]y = P = [[P/y]N/x]P = [[P/y]N/x]([P/y]y)$. Note-se que $x \notin LIV(P)$, na terceira igualdade.

- Caso $M = z$, com $x \neq z \neq y$:

  Ora, $[P/y]([N/x]z) = [P/y]z = [[P/y]N/x]z = [[P/y]N/x]([P/y]z)$.

- Caso $M = \lambda z.Q$:

  Ora, $[P/y]([N/x](\lambda z.Q)) = \lambda z.[P/y]([N/x]Q) = \lambda z.[[P/y]N/x]([P/y]Q)$
  $= [[P/y]N/x]([P/y](\lambda z.Q))$, onde a segunda igualdade segue por hipótese de indução em Q.

- Caso $M = QQ'$:

  Ora, $[P/y]([N/x](QQ')) = [P/y]([N/x]Q)[P/y]([N/x]Q')$
  $= [[P/y]N/x]([P/y]Q)[[P/y]N/x]([P/y]Q') = [[P/y]N/x]([P/y](QQ'))$, onde a segunda igualdade segue por hipótese de indução em Q e Q'.

□

Lema 3. *Sejam $N, N', M \in \Lambda$ tais que $N \twoheadrightarrow_1 N'$. Então,*

$$[N/x]M \twoheadrightarrow_\beta [N'/x]M.$$

Demonstração. A prova segue por indução em M.

- Caso $M = x$:

  Ora, $[N/x]x = N \twoheadrightarrow_\beta N' = [N'/x]x$, onde $N \twoheadrightarrow_\beta N'$ por hipótese.

- Caso $M = y$:

  Ora, $[N/x]y = y \twoheadrightarrow_\beta y = [N'/x]y$, onde $y \twoheadrightarrow_\beta y$ pela regra ρ.

- Caso $M = \lambda y.P$:

  Por hipótese de indução, $[N/x]P \twoheadrightarrow_\beta [N'/x]P$. Daqui segue que $\lambda y.[N/x]P \twoheadrightarrow_\beta \lambda y.[N'/x]P$, pela regra ξ. Logo, $[N/x](\lambda y.P) = \lambda y.[N/x]P \twoheadrightarrow_\beta \lambda y.[N'/x]P = [N'/x](\lambda y.P)$.

- Caso $M = PQ$:

  Por hipótese de indução, $[N/x]P \twoheadrightarrow_\beta [N'/x]P$ e $[N/x]Q \twoheadrightarrow_\beta [N'/x]Q$.

  De $[N/x]P \twoheadrightarrow_\beta [N'/x]P$ segue que $[N/x]P[N/x]Q \twoheadrightarrow_\beta [N'/x]P[N/x]Q$, pela regra μ, e de $[N/x]Q \twoheadrightarrow_\beta [N'/x]Q$ segue que $[N'/x]P[N/x]Q \twoheadrightarrow_\beta [N'/x]P[N'/x]Q$, pela regra ν. Aplicando a regra τ obtemos $[N'/x]P[N'/x]Q \twoheadrightarrow_\beta [N'/x]P[N'/x]Q$. Logo, $[N/x](PQ) = [N/x]P[N/x]Q \twoheadrightarrow_\beta [N'/x]P[N'/x]Q = [N'/x](PQ)$.

□

Lema 4. *Sejam $N, N', M, M' \in \Lambda$ tais que $N \twoheadrightarrow_1 N'$ e $M \twoheadrightarrow_1 M'$. Então,*

$$[N/x]M \twoheadrightarrow_1 [N'/x]M'.$$

Demonstração. A prova segue por indução na relação $M \twoheadrightarrow_1 M'$.

- Caso ρ:

  Suponhamos que $P \twoheadrightarrow_1 P$. É direto que $[N/x]P \twoheadrightarrow_1 [N'/x]P$, pelo lema 3.

- Caso ξ:

  Suponhamos que $\lambda y.P \twoheadrightarrow_1 \lambda y.P'$. Por hipótese de indução, $[N/x]P \twoheadrightarrow_1 [N'/x]P'$. Daqui segue que $\lambda y.[N/x]P \twoheadrightarrow_1 \lambda y.[N'/x]P'$, pela regra ξ. Logo, $[N/x](\lambda y.P) = \lambda y.[N/x]P \twoheadrightarrow_1 \lambda y.[N'/x]P' = [N'/x](\lambda y.P')$.

- Caso ϵ:

  Suponhamos que $PQ \twoheadrightarrow_1 P'Q'$. Por hipótese de indução, $[N/x]P \twoheadrightarrow_1 [N'/x]P'$ e $[N/x]Q \twoheadrightarrow_1 [N'/x]Q'$. Logo, $[N/x](PQ) = [N/x]P[N/x]Q \twoheadrightarrow_1 [N'/x]P'[N'/x]Q' = [N'/x](P'Q')$, aplicando a regra ϵ.

- Caso β:

  Suponhamos que $(\lambda y.P)Q \twoheadrightarrow_1 [Q'/y]P'$. Por hipótese de indução, $[N/x]P \twoheadrightarrow_1 [N'/x]P'$ e $[N/x]Q \twoheadrightarrow_1 [N'/x]Q'$. Logo, $[N/x]((\lambda y.P)Q) = [N/x](\lambda y.P)[N/x]Q = (\lambda y.[N/x]P)[N/x]Q \twoheadrightarrow_1 [([N'/x]Q')/y]([N'/x]P') = [N'/x]([Q'/y]P')$, aplicado a regra β e pelo lema 2.

□

Lema 5. *Sejam $M, N, P \in \Lambda$. Então,*

$$(i) \qquad \lambda x.M \twoheadrightarrow_1 N \implies N = \lambda x.M', \text{ com } M \twoheadrightarrow_1 M';$$

$$(ii) \qquad MN \twoheadrightarrow_1 P \implies \begin{cases} P = M'N', \text{ com } M \twoheadrightarrow_1 M' \text{ e } N \twoheadrightarrow_1 N'; \\ M = \lambda x.Q, \text{ com } P = [N'/x]Q', \ Q \twoheadrightarrow_1 Q' \text{ e } N \twoheadrightarrow_1 N'. \end{cases}$$

Demonstração. Pelo princípio genérico, sabemos que um par pertencente a uma relação definida por indução apenas pode ser obtido através de uma das regras de indução. A prova segue segundo este princípio.

- De $\lambda x.M \twoheadrightarrow_1 N$ segue, pela regra ξ, que existe M' tal que $N = \lambda x.M$ e $M \twoheadrightarrow_1 M'$.

- De $\lambda x.M \twoheadrightarrow_1 N$ segue, pela regra ρ, que $N = \lambda x.M$. Uma vez que $M \twoheadrightarrow_1 M'$, então esta regra pode ser vista como um caso particular da regra ξ onde $M' = M$.

- De $MN \twoheadrightarrow_1 P$ segue, pela regra ϵ, que existem M' e N' tais que $P = M'N'$, $M \twoheadrightarrow_1 M'$ e $N \twoheadrightarrow_1 N'$.

- De $MN \twoheadrightarrow_1 P$ segue, pela regra β, que existem Q, N, Q' e N' tais que $M = \lambda x.Q$, $P = [N'/x]Q'$, $N \twoheadrightarrow_1 N'$ e $Q \twoheadrightarrow_1 Q'$.

□

Lema 6. *A relação $\twoheadrightarrow_1$ satisfaz a propriedade do diamante.*

Demonstração. A prova segue por indução em $M \twoheadrightarrow_1 N$. Vamos mostrar que, para todo $M \twoheadrightarrow_1 N'$, existe um $P \in \Lambda$ tal que $N \twoheadrightarrow_1 P$ e $N' \twoheadrightarrow_1 P$.

- Caso ρ:

  Suponhamos que $N \twoheadrightarrow_1 N$, ou seja, $M = N$. Então, basta tomar $P = N'$. De facto, para todo $M = N \twoheadrightarrow_1 N'$, temos que $N \twoheadrightarrow_1 N' = P$ e $N' \twoheadrightarrow_1 N' = P$, pela regra ρ.

- Caso ξ:

  Suponhamos que $\lambda x.Q \twoheadrightarrow_1 \lambda x.Q'$, ou seja, $M = \lambda x.Q$ e $N = \lambda x.Q'$. De $M = \lambda x.Q \twoheadrightarrow_1 N'$, segue que $N' = \lambda x.Q''$, com $Q \twoheadrightarrow_1 Q''$, pelo lema 5. Então, basta tomar $P = \lambda x.Q'''$, com $Q' \twoheadrightarrow_1 Q'''$ e $Q'' \twoheadrightarrow_1 Q'''$. De facto, de $Q' \twoheadrightarrow_1 Q'''$ segue que $N = \lambda x.Q' \twoheadrightarrow_1 \lambda x.Q''' = P$ e, analogamente, de $Q'' \twoheadrightarrow_1 Q'''$ segue que $N' = \lambda x.Q'' \twoheadrightarrow_1 \lambda x.Q''' = P$, pela regra ξ.

- Caso ϵ:

  Suponhamos que $QF \twoheadrightarrow_1 Q'F'$, ou seja, $M = QF$ e $N = Q'F'$, por consequência de $Q \twoheadrightarrow_1 Q'$ e $F \twoheadrightarrow_1 F'$. Pelo lema 5 temos dois subcasos:

  - De $M = QF \twoheadrightarrow_1 N'$, segue que $N' = Q''F''$, com $Q \twoheadrightarrow_1 Q''$ e $F \twoheadrightarrow_1 F''$. Então, basta tomar $P = Q'''F'''$, com $Q' \twoheadrightarrow_1 Q'''$, $Q'' \twoheadrightarrow_1 Q'''$, $F' \twoheadrightarrow_1 F'''$ e $F'' \twoheadrightarrow_1 F'''$. De facto, de $Q' \twoheadrightarrow_1 Q'''$ e $F' \twoheadrightarrow_1 F'''$ segue que $N = Q'F' \twoheadrightarrow_1 Q'''F''' = P$ e, de forma análoga, de $Q'' \twoheadrightarrow_1 Q'''$ e $F'' \twoheadrightarrow_1 F'''$ segue que $N' = Q''F'' \twoheadrightarrow_1 Q'''F''' = P$, pela regra ϵ.
  - De $M = QF \twoheadrightarrow_1 N'$, segue que $Q = \lambda x.L$, com $N' = [F''/x]L''$, $L \twoheadrightarrow_1 L''$ e $F \twoheadrightarrow_1 F''$. Novamente pelo lema 5, de $\lambda x.L = Q \twoheadrightarrow_1 Q'$ segue que $Q' = \lambda x.L'$, com $L \twoheadrightarrow_1 L'$. Assim, $M = QF = (\lambda x.L)F$, $N = Q'F' = (\lambda x.L')F'$ e $N' = [F''/x]L''$.
    Então, basta tomar $P = [F'''/x]L'''$, com $L' \twoheadrightarrow_1 L'''$, $L'' \twoheadrightarrow_1 L'''$, $F' \twoheadrightarrow_1 F'''$ e $F'' \twoheadrightarrow_1 F'''$. De facto, de $L' \twoheadrightarrow_1 L'''$ e $F' \twoheadrightarrow_1 F'''$ segue que $N = (\lambda x.L')F' \twoheadrightarrow_1 [F'''/x]L''' = P$, pela regra β, e de $L'' \twoheadrightarrow_1 L'''$ e $F'' \twoheadrightarrow_1 F'''$ segue que $N' = [F''/x]L'' \twoheadrightarrow_1 [F'''/x]L''' = P$, pelo lema 4.

- Caso β:

  Suponhamos que $(\lambda x.L)F \twoheadrightarrow_1 [F'/x]L'$, ou seja, $M = (\lambda x.L)F$ e $N = [F'/x]L'$, por consequência de $L \twoheadrightarrow_1 L'$ e $F \twoheadrightarrow_1 F'$.
  Pelo lema 5 temos dois subcasos:

- De $M = (\lambda x.L)F \twoheadrightarrow_1 N'$, segue que $N' = (\lambda x.L'')F''$, com $L \twoheadrightarrow_1 L''$ e $F \twoheadrightarrow_1 F''$. Então, basta tomar $P = [F'''/x]L'''$, com $L' \twoheadrightarrow_1 L'''$, $L'' \twoheadrightarrow_1 L'''$, $F' \twoheadrightarrow_1 F'''$ e $F'' \twoheadrightarrow_1 F'''$. De facto, de $L' \twoheadrightarrow_1 L'''$ e $F' \twoheadrightarrow_1 F'''$ segue que $N = [F'/x]L' \twoheadrightarrow_1 [F'''/x]L''' = P$, pelo lema 4, e de $L'' \twoheadrightarrow_1 L'''$ e $F'' \twoheadrightarrow_1 F'''$ segue que $N' = (\lambda x.L'')F'' \twoheadrightarrow_1 [F'''/x]L''' = P$, pela regra β.
- De $M = (\lambda x.L)F \twoheadrightarrow_1 N'$, segue que $N' = [F''/x]L''$, com $L \twoheadrightarrow_1 L''$ e $F \twoheadrightarrow_1 F''$. Então, basta tomar $P = [F'''/x]L'''$, com $L' \twoheadrightarrow_1 L'''$, $L'' \twoheadrightarrow_1 L'''$, $F' \twoheadrightarrow_1 F'''$ e $F'' \twoheadrightarrow_1 F'''$. De facto, de $L' \twoheadrightarrow_1 L'''$ e $F' \twoheadrightarrow_1 F'''$ segue que $N = [F'/x]L' \twoheadrightarrow_1 [F'''/x]L''' = P$ e, analogamente, de $L'' \twoheadrightarrow_1 L'''$ e $F'' \twoheadrightarrow_1 F'''$ segue que $N' = [F''/x]L'' \twoheadrightarrow_1 [F'''/x]L''' = P$, pelo lema 4.

□

Lema 7. *O fecho transitivo de $\twoheadrightarrow_1$ é dado pela relação $\twoheadrightarrow_\beta$.*

Demonstração. Sabemos que $\rightarrow_\beta \subseteq \twoheadrightarrow_1 \subseteq \twoheadrightarrow_\beta$. Uma vez que $\twoheadrightarrow_\beta$ corresponde ao fecho transitivo (e reflexivo) de $\rightarrow_\beta$, então é direto que também corresponde ao de $\twoheadrightarrow_1$.

□

Teorema 1 (**Confluência de** β)**.** *A relação β é confluente.*

Demonstração. Pelo lema 6, temos que $\twoheadrightarrow_1 \models \diamond$. Assim, pelos lemas 7 e 1, segue que $\twoheadrightarrow_\beta \models \diamond$ e, consequentemente pela definição de confluência, β é confluente. □

Teorema 2 (**Church-Rosser**)**.** *Sejam $M, N \in \Lambda$. Então,*

$$M =_\beta N \iff \exists P \in \Lambda : M \twoheadrightarrow_\beta P \wedge N \twoheadrightarrow_\beta P.$$

Demonstração. Primeiramente, suponhamos que $M =_\beta N$. Queremos provar que $\exists P \in \Lambda$ tal que $M \twoheadrightarrow_\beta P$ e $N \twoheadrightarrow_\beta P$. A prova segue por indução em $M =_\beta N$.

- Caso $=_\beta$:

  Suponhamos que $M =_\beta N$, por consequência de $M \rightarrow_\beta N$. Então, basta tomar $P = N$. De facto, de $M \rightarrow_\beta N$ segue que $M \twoheadrightarrow_\beta N = P$, pela regra $\twoheadrightarrow_\beta$, e $N \twoheadrightarrow_\beta N = P$, pela regra ρ.

- Caso ρ:

  Suponhamos que $M = \beta M$, isto é, $N = M$. Então, basta tomar $P = M$. De facto, $M \twoheadrightarrow_\beta M = P$ e $N = M \twoheadrightarrow_\beta M = P$ pela regra ρ.

- Caso τ:

  Suponhamos que $M =_\beta N$, por consequência de $M =_\beta Q$ e $Q =_\beta N$. Por hipótese de indução, $\exists P' \in \Lambda$ tal que $M \twoheadrightarrow_\beta P'$ e $Q \twoheadrightarrow_\beta P'$ e $\exists P'' \in \Lambda$ tal que $Q \twoheadrightarrow_\beta P''$ e $N \twoheadrightarrow_\beta P''$. Uma vez que β é confluente, então de $Q \twoheadrightarrow_\beta P'$ e $Q \twoheadrightarrow_\beta P''$ segue que $\exists P''' \in \Lambda$ tal que $P' \twoheadrightarrow_\beta P'''$ e $P'' \twoheadrightarrow_\beta P'''$. Então, basta tomar $P = P'''$. De facto, $M \twoheadrightarrow_\beta P' \twoheadrightarrow_\beta P''' = P$ e $N \twoheadrightarrow_\beta P'' \twoheadrightarrow_\beta P''' = P$.

- Caso σ:

  Suponhamos que $M =_\beta N$, por consequência de $N =_\beta M$. De $N =_\beta M$ segue que $\exists Q \in \Lambda$ tal que $N \twoheadrightarrow_\beta Q$ e $M \twoheadrightarrow_\beta Q$, por hipótese de indução. Então, basta tomar $P = Q$. De facto, $M \twoheadrightarrow_\beta Q = P$ e $N \twoheadrightarrow_\beta Q = P$.

Reciprocamente, suponhamos agora que $\exists P \in \Lambda$ tal que $M \twoheadrightarrow_\beta P$ e $N \twoheadrightarrow_\beta P$. Queremos provar que $M =_\beta N$. De $N \twoheadrightarrow_\beta P$ segue que $N =_\beta P$ e, por isso, $P =_\beta N$, respetivamente pelas regras $=_\beta$ e σ. De $M \twoheadrightarrow_\beta P$ segue que $M =_\beta P$, pela regra $=_\beta$. Logo, de $M =_\beta P$ e $P =_\beta N$ obtemos finalmente que $M =_\beta N$, pela regra τ.

□

Corolário 1. *Cada termo-λ tem no máximo uma forma normal-β.*

Demonstração. Suponhamos que N e P são fn's-β de um termo-λ M. Então, $M \twoheadrightarrow_\beta N$ e $M \twoheadrightarrow_\beta P$. Uma vez que $\twoheadrightarrow_\beta$ satisfaz a propriedade do diamante, $\exists Q \in \Lambda$ tal que $N \twoheadrightarrow_\beta Q$ e $P \twoheadrightarrow_\beta Q$. Assim, como N e P são fn's-β, isto é, nenhum dos seus termos é um redex-β, então $N = Q = P$. □

Teorema 3 (Consistência). *Sejam M e N duas formas normais-β distintas. Então, $M \neq_\beta N$.*

Demonstração. A prova segue por redução ao absurdo. Suponhamos que $M =_\beta N$. Então, pelo teorema de Church-Rosser, $\exists P \in \Lambda$ tal que $M \twoheadrightarrow_\beta P$ e $N \twoheadrightarrow_\beta P$. Mas, desta forma, $P = M$ e $P = N$ e, portanto, $M = N$. Isto é absurdo visto que, por hipótese, $M \neq N$. □

2.2 Conceitos sobre relações de redução

Os seguintes conceitos podem ser encontrados em [16]. Estes serão fundamentais, em particular, nos capítulos 4 e 5, com o objetivo estabelecer relações entre diferentes cálculos.

Definição 18 (**Correspondência Equacional**). *Sejam $\mathcal{A}$ e $\mathcal{B}$ dois conjuntos com a relação de redução em um passo $\rightarrow_{\mathcal{A}}$ e $\rightarrow_{\mathcal{B}}$, respetivamente. Considere-se os mapeamentos $f(\cdot) : \mathcal{A} \rightarrow \mathcal{B}$ e $g(\cdot) : \mathcal{B} \rightarrow \mathcal{A}$. Então, f e g formam uma correspondência equacional entre $\mathcal{A}$ e $\mathcal{B}$ se:*

- $M =_{\mathcal{A}} N \implies f(M) =_{\mathcal{B}} f(N)$;
- $M =_{\mathcal{B}} N \implies g(M) =_{\mathcal{A}} g(N)$;
- $M =_{\mathcal{A}} g(f(M))$;
- $f(g(M)) =_{\mathcal{B}} M$.

Definição 19 (**Conexão de Galois**). *Sejam $\mathcal{A}$ e $\mathcal{B}$ dois conjuntos com a relação de redução em um passo $\rightarrow_{\mathcal{A}}$ e $\rightarrow_{\mathcal{B}}$, respetivamente. Considere-se os mapeamentos $f(\cdot) : \mathcal{A} \rightarrow \mathcal{B}$ e $g(\cdot) : \mathcal{B} \rightarrow \mathcal{A}$. Então, f e g formam uma conexão de Galois de $\mathcal{A}$ para $\mathcal{B}$ se:*

- $M \twoheadrightarrow_{\mathcal{A}} N \implies f(M) \twoheadrightarrow_{\mathcal{B}} f(N)$;
- $M \twoheadrightarrow_{\mathcal{B}} N \implies g(M) \twoheadrightarrow_{\mathcal{A}} g(N)$;
- $M \twoheadrightarrow_{\mathcal{A}} g(f(M))$;
- $f(g(M)) \twoheadrightarrow_{\mathcal{B}} M$.

Definição 20 (**Pré-conexão de Galois**). *Sejam $\mathcal{A}$ e $\mathcal{B}$ dois conjuntos com a relação de redução em um passo $\rightarrow_{\mathcal{A}}$ e $\rightarrow_{\mathcal{B}}$, respetivamente. Considere-se os mapeamentos $f : \mathcal{A} \rightarrow \mathcal{B}$ e $g : \mathcal{B} \rightarrow \mathcal{A}$. Então, f e g formam uma pré-conexão de Galois de $\mathcal{A}$ para $\mathcal{B}$ se, à definição anterior, removermos a última condição.*

Definição 21 (**Reflexão**). *Sejam $\mathcal{A}$ e $\mathcal{B}$ dois conjuntos com a relação de redução em um passo $\rightarrow_{\mathcal{A}}$ e $\rightarrow_{\mathcal{B}}$, respetivamente. Considere-se os mapeamentos $f(\cdot) : \mathcal{A} \rightarrow \mathcal{B}$ e $g(\cdot) : \mathcal{B} \rightarrow \mathcal{A}$. Então, f e g formam uma reflexão em $\mathcal{A}$ de $\mathcal{B}$ se:*

- $M \twoheadrightarrow_{\mathcal{A}} N \implies f(M) \twoheadrightarrow_{\mathcal{B}} f(N)$;

- $M \twoheadrightarrow_{\mathcal{B}} N \implies g(M) \twoheadrightarrow_{\mathcal{A}} g(N)$;
- $M \twoheadrightarrow_{\mathcal{A}} g(f(M))$;
- $M = f(g(M))$.

Teorema 4 (**Confluência por simulação**). *Se f e g formam uma pré-conexão de Galois de $\mathcal{A}$ para $\mathcal{B}$ e $\rightarrow_{\mathcal{B}}$ é confluente, então $\rightarrow_{\mathcal{A}}$ é confluente.*

2.3 Cálculo-λ com tipos simples

Nesta secção vamos apresentar o cálculo-λ com tipificação à la Curry [5], representado por $\lambda^{\rightarrow}$. Este sistema pode ser encontrado em [17].

Definição 22 (**Tipos simples**). *Seja $\mathbb{B} = \mathbb{A} \cup \{(,), \rightarrow\}$, onde $\mathbb{A}$ é um conjunto numerável de variáveis de tipo, denotadas por a, b, c, etc. O conjunto de todas as palavras em $\mathbb{B}$ é representado por $\mathbb{B}^*$. O conjunto dos tipo simples, representado por $\mathbb{T}$, é definido indutivamente sobre $\mathbb{B}^*$ por:*

$$\rho, \sigma, \tau \in \mathbb{T} ::= a \mid (\sigma \rightarrow \tau)$$

As variáveis de tipo dizem-se tipos atómicos e os tipos da forma $(\sigma \rightarrow \tau)$ dizem-se o tipo das funções de σ em τ.

Por convenção: (i) omite-se os parêntesis mais externos; (ii) a associação de $\rightarrow$ é feita à direita, isto é, $\rho \rightarrow \sigma \rightarrow \tau$ abrevia $\rho \rightarrow (\sigma \rightarrow \tau)$.

Definição 23 (**Contextos de tipificação**). *Um contexto de tipificação é um conjunto finito (eventualmente vazio) de elementos da forma $y : \tau$, onde y é uma variável, τ um tipo simples e a cada variável é atribuído no máximo um tipo, e representa-se por Γ, Δ, Γ', etc.*

Por convenção: (i) omite-se os parêntesis mais externos do conjunto; (ii) utiliza-se $\Gamma, x : \sigma$ para denotar $\Gamma \cup \{x : \sigma\}$, assumindo que estes dois conjuntos são disjuntos.

Definição 24 (**Relação de tipificação à la Curry**). *A relação de tipificação à la Curry entre contextos Γ, termos-λ M e tipos simples τ, representada por $\Gamma \vdash M : \tau$, é definida indutivamente por:*

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}\ Var \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x.M) : \sigma \rightarrow \tau}\ Abs \qquad \frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\ App$$

A relação $\Gamma \vdash M : \tau$ lê-se como M tem tipo τ no contexto Γ. A M chama-se sujeito e a τ predicado.

Definição 25 (**Termo-λ tipificável**). *Um termo-λ M diz-se tipificável se $\Gamma \vdash M : \sigma$ para algum Γ e algum σ.*

Vejamos o caso do combinador $\mathbf{S} = \lambda xyz.xz(yz)$, que é tipificável. Basta tomar $\Gamma = \{\}$ e $\sigma = (\sigma \to \tau \to \rho) \to (\sigma \to \tau) \to \sigma \to \rho$. Ora:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Delta \vdash x : \sigma \to \tau \to \rho}\,Var \quad \dfrac{}{\Delta \vdash z : \sigma}\,Var}{\Delta \vdash xz : \tau \to \rho}\,App \quad \dfrac{\dfrac{}{\Delta \vdash y : \sigma \to \tau}\,Var \quad \dfrac{}{\Delta \vdash z : \sigma}\,Var}{\Delta \vdash yz : \tau}\,App}{\Delta = x : \sigma \to \tau \to \rho, y : \sigma \to \tau, z : \sigma \vdash xz(yz) : \rho}\,App}{x : \sigma \to \tau \to \rho, y : \sigma \to \tau \vdash \lambda z.xz(yz) : \sigma \to \rho}\,Abs}{x : \sigma \to \tau \to \rho \vdash \lambda yz.xz(yz) : (\sigma \to \tau) \to \sigma \to \rho}\,Abs}{\vdash \lambda xyz.xz(yz) : (\sigma \to \tau \to \rho) \to (\sigma \to \tau) \to \sigma \to \rho}\,Abs$$

2.4 Sistemas dedutivos

Na década de 1930, G. Gentzen introduziu o cálculo de sequentes e a dedução natural [8]. Estes são sistemas dedutivos, onde são identificados axiomas e regras que permitem construir deduções.

2.4.1 Dedução natural e isomorfismo de Curry-Howard

Nesta secção vamos estudar a dedução natural com sequentes NJ, que se relaciona com o cálculo-λ com tipos simples através do isomorfismo de Curry-Howard. De uma forma informal, o isomorfismo de de Curry-Howard estabelece uma equivalência entre estes dois sistemas, criando uma correspondência entre fórmulas e tipos e entre provas e termos, respetivamente. O sistema NJ encontra-se apresentado em [17].

Definição 26 (**Fórmulas**). *Seja $\mathcal{A}_p = \mathcal{V}_p \cup \{(,), \supset\}$, onde $\mathcal{V}_p$ é um conjunto numerável de variáveis proposicionais, denotadas por p, q, p', etc. O conjunto de todas as palavras em $\mathcal{A}_P$ é representado por $\mathcal{A}_p^*$. O conjunto das fórmulas, representado por $\mathcal{F}_p$, é definido indutivamente sobre $\mathcal{A}_p^*$ por:*

$$A, B, C ::= p \mid (A \supset B)$$

Definição 27 (**Sequente de NJ**). *Um sequente é um par (Γ, A), que será representado por $\Gamma \Rightarrow A$, onde Γ é um conjunto finito de fórmulas e A é uma fórmula. A Γ dá-se o nome de contexto, antecedente ou lado esquerdo do sequente. A A dá-se o nome de conclusão, consequente ou lado direito do sequente.*

Por convenção: (i) omite-se os parêntesis mais externos; (ii) a associação de $\supset$ é feita à direita; (iii) utiliza-se Γ, A para denotar $\Gamma \cup \{A\}$, assumindo que estes dois conjuntos são disjuntos.

De forma intuitiva, um sequente $\Gamma \Rightarrow A$, com $\Gamma = B_1, \cdots, B_n$, transmite a ideia de que $B_1 \wedge \cdots \wedge B_n$ implica A.

Definição 28 (**Árvore de sequentes**). *Uma árvore de sequentes tem a forma:*

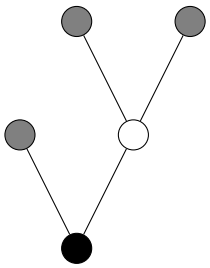

Figura 1: Árvore de sequentes

Cada nodo representa um sequente. Ao nodo preto chamamos raiz ou conclusão e aos nodos cinzentos, que representam unicamente axiomas, chamamos folhas. Uma árvore com um único nodo é simultaneamente folha e raiz.

Definição 29 (**Derivações de** NJ). *O conjunto de derivações de NJ, apresentado em [17], é o menor conjunto de árvores de sequentes fechado para as seguintes regras de inferência:*

$$\frac{}{\Gamma, A \Rightarrow A}\ Ax \qquad \frac{\Gamma \Rightarrow A \supset B \quad \Gamma \Rightarrow A}{\Gamma \Rightarrow B}\ E\supset \qquad \frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \supset B}\ I\supset$$

Uma derivação $\mathcal{D}$ diz-se derivação de $\Gamma \Rightarrow A$, e representa-se por $\Gamma \overset{\mathcal{D}}{\Rightarrow} A$, se $\mathcal{D}$ é uma árvore de sequentes com conclusão $\Gamma \Rightarrow A$. Um sequente $\Gamma \Rightarrow A$ diz-se derivável se existir $\mathcal{D}$ derivação de $\Gamma \Rightarrow A$.

Seja $\Delta = A \supset B \supset C, A \supset B, A$. Vejamos a seguinte derivação do sequente $\Rightarrow (A \supset B \supset C) \supset (A \supset B) \supset A \supset C$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Delta \Rightarrow A \supset B \supset C}\ Ax \quad \dfrac{}{\Delta \Rightarrow A}\ Ax}{\Delta \Rightarrow B \supset C}\ E\supset \quad \dfrac{\dfrac{}{\Delta \Rightarrow A \supset B}\ Ax \quad \dfrac{}{\Delta \Rightarrow A}\ Ax}{\Delta \Rightarrow B}\ E\supset}{\Delta = A \supset B \supset C, A \supset B, A \Rightarrow C}\ I\supset}{A \supset B \supset C, A \supset B \Rightarrow A \supset C}\ I\supset}{A \supset B \supset C \Rightarrow (A \supset B) \supset A \supset C}\ I\supset}{\Rightarrow (A \supset B \supset C) \supset (A \supset B) \supset A \supset C}\ I\supset$$

Comparando esta derivação com a tipificação feita na secção 2.3, podemos identificar a ideia inerente ao isomorfismo de Curry-Howard, que apresentaremos de seguida.

Definição 30 (**Traduções entre $\mathcal{F}_p$ e $\mathbb{T}$**). *Sejam $f(\cdot) : \mathcal{F}_p \to \mathbb{T}$ e $g(\cdot) : \mathbb{T} \to \mathcal{F}_p$ duas bijeções definidas entre os conjuntos numeráveis $\mathcal{F}_p$ e $\mathbb{T}$.*

Consideremos, por exemplo, que $f(A) = \sigma$. Se $g(\sigma) = B$, então $B = A$.

Definição 31 (**Contexto de um sequente associado**). *Dado um contexto de tipificação $\Gamma = \{y_1 : \tau_1, \ldots, y_n : \tau_n\}$, o contexto de um sequente associado, representado por $g(\Gamma)$, é dado por $g(\Gamma) = \{g(\tau_1), \ldots, g(\tau_n)\}$.*

Proposição 1 (**Isomorfismo de Curry-Howard**).

(i) *Se $\Delta \Rightarrow A$ é derivável em NJ e $\Delta = g(\Gamma)$, para algum Γ em $\lambda^{\to}$, então existe $M \in \Lambda$ tal que $\Gamma \vdash M : f(A)$ em $\lambda^{\to}$.*

(ii) *Se $\Gamma \vdash M : \tau$ em $\lambda^{\to}$, então $g(\Gamma) \Rightarrow g(\tau)$ é derivável em NJ.*

Demonstração. A prova de (i) segue por indução nas derivações de NJ e a prova de (ii) segue por indução nas tipificações de $\lambda^{\to}$. □

2.4.2 Cálculo de sequentes

Nesta secção vamos estudar o cálculo de sequentes LJ, que vamos relacionar, mais à frente, com o fragmento LJQ. O sistema LJ pode ser encontrado em [17].

Definição 32 (**Multiconjunto**). *Um multiconjunto sobre A é uma função $f : A \to \mathbb{N}$, onde $f(a) = n$ significa que a ocorre n vezes em A.*

Definição 33 (**Sequente de LJ**). *Um sequente é um par (Γ, Δ), que será representado por $\Gamma \Rightarrow \Delta$, onde Γ é um multiconjunto finito sobre $\mathcal{F}_p$ e Δ tem no máximo uma fórmula. A Γ dá-se o nome de contexto, antecedente ou lado esquerdo do sequente. A Δ dá-se o nome de conclusão, consequente ou lado direito do sequente.*

Definição 34 (**Derivações de LJ**). *O conjunto de derivações de LJ é o menor conjunto de árvores de sequentes fechado para as seguintes regras de inferência:*

- *Axioma:*

$$\frac{}{A \Rightarrow A}\ Ax$$

- *Regras Estruturais:*

$$\frac{\Gamma \Rightarrow C}{\Gamma, A \Rightarrow C}\ LW \qquad \frac{\Gamma \Rightarrow}{\Gamma \Rightarrow A}\ RW$$

$$\frac{\Gamma, A, A \Rightarrow C}{\Gamma, A \Rightarrow C}\ LC$$

- *Regras Lógicas:*

$$\frac{\Gamma \Rightarrow A \quad \Gamma, B \Rightarrow C}{\Gamma, A \supset B \Rightarrow C}\ L\supset \qquad \frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \supset B}\ R\supset$$

- *Regra do Corte:*

$$\frac{\Gamma \Rightarrow A \quad \Gamma, A \Rightarrow B}{\Gamma \Rightarrow B}\ Cut$$

Uma derivação $\mathcal{D}$ diz-se derivação de $\Gamma \Rightarrow A$, e representa-se por $\Gamma \overset{\mathcal{D}}{\Rightarrow} A$, se $\mathcal{D}$ é uma árvore de sequentes com conclusão $\Gamma \Rightarrow A$. Um sequente $\Gamma \Rightarrow A$ diz-se derivável se existir $\mathcal{D}$ derivação de $\Gamma \Rightarrow A$.

Definição 35 (**Regra eliminável**). *Uma regra de inferência RI pertencente a um cálculo sequentes diz-se eliminável se, para todo o sequente $\Gamma \Rightarrow A$ derivável nesse cálculo, existe uma derivação de $\Gamma \Rightarrow A$ nesse cálculo sem recurso à regra RI.*

Teorema 5 (**Gentzen**). *A regra do corte, Cut, é eliminável.*

Demonstração. A prova pode ser encontrada em [9]. □

Vejamos duas derivações distintas do sequente $\Rightarrow (A \supset B \supset C) \supset (A \supset B) \supset A \supset C$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \Rightarrow A}\,Ax}{A \supset B, A \Rightarrow A}\,LW \quad \dfrac{\dfrac{\dfrac{\dfrac{}{A \Rightarrow A}\,Ax}{A, B \supset C \Rightarrow A}\,LW \quad \dfrac{\dfrac{\dfrac{}{B \Rightarrow B}\,Ax}{A, B \Rightarrow B}\,LW \quad \dfrac{\dfrac{\dfrac{}{C \Rightarrow C}\,Ax}{B, C \Rightarrow C}\,LW}{A, B, C \Rightarrow C}\,LW}{A, B \supset C, B \Rightarrow C}\,L\supset}{\boxed{A \supset B}, A, B \supset C \Rightarrow C}\,\boxed{L\supset}}{A \supset B \supset C, A \supset B, A \Rightarrow C}\,L\supset}{A \supset B \supset C, A \supset B \Rightarrow A \supset C}\,R\supset}{A \supset B \supset C \Rightarrow (A \supset B) \supset A \supset C}\,R\supset}{\Rightarrow (A \supset B \supset C) \supset (A \supset B) \supset A \supset C}\,R\supset$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \Rightarrow A}\,Ax}{A \supset B, A \Rightarrow A}\,LW \quad \dfrac{\dfrac{\dfrac{}{A \Rightarrow A}\,Ax \quad \dfrac{\dfrac{}{B \Rightarrow B}\,Ax}{A, B \Rightarrow B}\,LW}{A \supset B, A \Rightarrow B}\,L\supset \quad \dfrac{\dfrac{\dfrac{}{C \Rightarrow C}\,Ax}{A, C \Rightarrow C}\,LW}{A \supset B, A, C \Rightarrow C}\,LW}{A \supset B, A, \boxed{B \supset C} \Rightarrow C}\,\boxed{L\supset}}{A \supset B \supset C, A \supset B, A \Rightarrow C}\,L\supset}{A \supset B \supset C, A \supset B \Rightarrow A \supset C}\,R\supset}{A \supset B \supset C \Rightarrow (A \supset B) \supset A \supset C}\,R\supset}{\Rightarrow (A \supset B \supset C) \supset (A \supset B) \supset A \supset C}\,R\supset$$

As regras destacadas indicam o ponto a partir do qual as derivações diferem, quando são percorridas de baixo para cima, e as fórmulas destacadas indicam a fórmula utilizada nessa derivação.

Enquanto que na dedução natural com sequentes temos regras de introdução e de eliminação dos conectivos, no cálculo de sequente temos regras de introdução dos conectivos à esquerda e à direita do sequente, pelo que temos mais alternativas de derivações.

Em suma, o cálculo-λ é um sistema consistente que, quando considerando como um sistema de tipos, é equivalente á dedução natural com sequentes para a lógica intuicionista.

No próximo capítulo vamos compreender a importância do cálculo-λ nas linguagens de programação funcionais.

Capítulo 3

λ-calculi e linguagens de programação

Uma linguagem de programação funcional pode ser vista como programas, resultados e uma estratégia de avaliação, equivalente a uma máquina abstrata, onde os programas são avaliados através da estratégia de avaliação originando, assim, resultados.

Plotkin mostra que é possível modelar uma linguagem de programação *call-by-name* e *call-by-value* através dos cálculos λ e λ_v, respetivamente [14]. Para isso, os programas, os resultados e a estratégia de avaliação de uma linguagem devem ser modelados pelos termos fechados, pelos valores (variáveis e abstrações) e pela estratégia de redução do cálculo correspondente, respetivamente.

3.1 Cálculo-λ e linguagem CBN

As demonstrações dos resultados deste sistema *call-by-name* serão omitidas, uma vez que são análogas às do sistema *call-by-value*, que serão realizadas na secção seguinte.

Definição 36 (**Relação de redução à esquerda**). *Seja* $\rightarrow_n \subseteq \Lambda^2$ *a relação binária definida indutivamente por:*

$$\frac{M, N \in \Lambda}{(\lambda x.M)N \rightarrow_n [N/x]M}\ \beta \qquad \frac{M \rightarrow_n M'}{MN \rightarrow_n M'N}\ \nu \qquad \frac{M \rightarrow_n M'}{xM \rightarrow_n xM'}\ \mu$$

Observemos que, se $M \rightarrow_n N$, então M é necessariamente uma aplicação. Além disso, se $M \rightarrow_n N$ então $M \rightarrow_\beta N$.

Definição 37 (**Relação de redução standard**). *Seja* $\rightarrow_s \subseteq \Lambda^2$ *a relação binária definida indutivamente por:*

$$\frac{x \in \Lambda}{x \to_s x}\ \rho \qquad \frac{M \to_s M'}{\lambda x.M \to_s \lambda x.M'}\ \xi \qquad \frac{M \to_n N \quad N \to_s P}{M \to_s P}\ \tau \qquad \frac{M \to_s M' \quad N \to_s N'}{MN \to_s M'N'}\ \epsilon$$

Teorema 6. *A relação $\to_s$ é reflexiva e transitiva.*

Teorema 7 (**Standarização**)**.** *Sejam $M, N \in \Lambda$. Então,*

$$M \twoheadrightarrow_\beta N \iff M \to_s N.$$

Definição 38 (**Valor**)**.** *Um termo-λ diz-se um valor se não for uma aplicação, isto é, se for uma variável ou uma abstração, e representa-se por V ou W.*

A relação de redução standard não é determinista na medida em que não está determinado o momento de paragem de reduções à esquerda. Se M é uma aplicação e $M \to_s N$, para algum N, tanto podemos aplicar a regra τ, e consequentemente temos uma redução à esquerda, como podemos aplicar a regra ϵ. Desta forma, necessitamos do resultado seguinte, que nos dá uma forma determinista de obter um valor.

Corolário 2. *Sejam $M, V \in \Lambda$. Então,*

$$M \twoheadrightarrow_\beta V \implies M \twoheadrightarrow_n W, \text{ para algum } W.$$

Com efeito, dada uma abstração, é possível fazer reduções-β no corpo dessa abstração, sendo que o mesmo não acontece com a relação de redução à esquerda.

Observemos alguns exemplos:

1. Ora, $M = (\lambda x.x)((\lambda y.y)z) \to_\beta (\lambda y.y)z \to_\beta z = V$. Vejamos:

$$\dfrac{\dfrac{}{(\lambda x.x)((\lambda y.y)z) \to_n (\lambda y.y)z}\ \beta \quad \dfrac{\dfrac{}{(\lambda y.y)z \to_n z}\ \beta \quad \dfrac{}{z \to_s z}\ \rho}{(\lambda y.y)z \to_s z}\ \tau}{(\lambda x.x)((\lambda y.y)z) \to_s z}\ \tau$$

   Com efeito, $M \to_s V$ e $M \twoheadrightarrow_n V$.

2. Ora, $M = \lambda u.((\lambda x.x)u)((\lambda y.y)z) \to_\beta \lambda u.((\lambda x.x)u)z \to_\beta \lambda u.uz = V$. Vejamos:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(\lambda x.x)u \to_n u}\ \beta \quad \dfrac{}{u \to_s u}\ \rho}{(\lambda x.x)u \to_s u}\ \tau \quad \dfrac{\dfrac{}{(\lambda y.y)z \to_n z}\ \beta \quad \dfrac{}{z \to_s z}\ \rho}{(\lambda y.y)z \to_s z}\ \tau}{((\lambda x.x)u)((\lambda y.y)z) \twoheadrightarrow_s uz}\ \epsilon}{\lambda u.((\lambda x.x)u)((\lambda y.y)z) \to_s \lambda u.uz}\ \xi$$

Com efeito, $M \to_s V$ e, tomando $W = M$, $M \twoheadrightarrow_n W$.

Definição 39 (**Programas**). *O conjunto de programas de uma linguagem de programação, representado por $\mathcal{P}$, é o conjunto de termos-λ fechados que modela essa linguagem.*

Definição 40 (**Função de avaliação**). *Seja $eval_n : \mathcal{P} \rightsquigarrow \mathcal{P}$*[1] *a função parcial de avaliação de uma linguagem de programação call-by-name definida por:*

$$eval_n(\lambda x.M) = \lambda x.M$$
$$eval_n(MN) = eval_n([N/x]P), \text{ se } eval_n(M) = \lambda x.P$$

Utilizamos a notação $eval_n(M)\downarrow$ para indicar que $eval_n(M)$ está definido.

Teorema 8. *Sejam $M, V \in \Lambda$ tal que M é um termo fechado. Então,*

$$(eval_n(M)\downarrow \ \wedge \ eval_n(M) = V) \iff M \twoheadrightarrow_n V.$$

Este teorema expressa como a estratégia de avaliação $eval_n$ da linguagem *call-by-name* é modelada pela estratégia de redução $\twoheadrightarrow_n$ no cálculo-λ.

3.2 Cálculo-λ_v de Plotkin e linguagem CBV

Definição 41 (**Relação β_v**). *Seja $\beta_v \subseteq \Lambda^2$ a relação apresentada sobre a forma da seguinte regra:*

$$(\beta_v) \qquad (\lambda x.M)V \to [V/x]M$$

É de notar que esta relação difere da relação β no termo N, que tem obrigatoriamente de ser um valor neste sistema.

Os resultados relativos à confluência, ao Teorema de Church-Rosser e à consistência da relação β_v, podem ser provados analogamente ao que foi realizado para a relação β, no capítulo 2.1, uma vez que existem menos pares divergentes.

Definição 42 (**Relação de redução à esquerda**). *Seja $\to_v \subseteq \Lambda^2$ a relação binária definida indutivamente por:*

$$\frac{M, V \in \Lambda}{(\lambda x.M)V \to_v [V/x]M}\ \beta_v \qquad \frac{M \to_v M'}{MN \to_v M'N}\ \nu \qquad \frac{M \to_v M'}{VM \to_v VM'}\ \mu$$

[1] Utilizamos $\rightsquigarrow$ para representar uma função parcial.

Mais uma vez, podemos observar que se $M \to_v N$, então M não é um valor, mas sim uma aplicação. Mais, se $M \to_v N$ então $M \to_{\beta_v} N$.

Definição 43 (Relação de redução standard). *Seja $\to_s \subseteq \Lambda^2$ a relação binária definida indutivamente por:*

$$\frac{x \in \Lambda}{x \to_s x}\ \rho \qquad \frac{M \to_s M'}{\lambda x.M \to_s \lambda x.M'}\ \xi \qquad \frac{M \to_v N \quad N \to_s P}{M \to_s P}\ \tau \qquad \frac{M \to_s M' \quad N \to_s N'}{MN \to_s M'N'}\ \epsilon$$

Teorema 9. *A relação $\to_s$ é reflexiva e transitiva.*

Demonstração. Para provar que a relação é reflexiva, é necessário mostrar que, para todo $M \in \Lambda$, $M \to_s M$. Neste caso, a prova segue por uma simples indução em M. Por fim, para provar que a relação é transitiva, é necessário mostrar que, para todos $M, N, P \in \Lambda$, se $M \to_s N$ e $N \to_s P$, então $M \to_s P$. Neste caso, a prova segue por indução em $M \to_s N$. □

Teorema 10 (Standarização). *Sejam $M, N \in \Lambda$. Então,*

$$M \twoheadrightarrow_{\beta_v} N \iff M \to_s N.$$

Demonstração. A prova pode ser encontrada em [14]. □

Corolário 3. *Sejam $M, V \in \Lambda$. Então,*

$$M \twoheadrightarrow_{\beta_v} V \implies M \twoheadrightarrow_v W, \text{ para algum } W.$$

Demonstração. Suponhamos que $M \twoheadrightarrow_{\beta_v} V$. Então, pelo teorema da standarização, $M \to_s V$. Falta ver que $M \twoheadrightarrow_v W$, para algum W. A prova segue segue por indução em $M \to_s V$.

- Caso ρ:

  Suponhamos que $x \to_s x$, ou seja, $M = V = x$. Basta tomar $W = M$. É direto que $M = x \twoheadrightarrow_v x = W$, pela reflexividade da relação $\twoheadrightarrow_v$.

- Caso ξ:

  Suponhamos que $\lambda x.N \to_s \lambda x.N'$, ou seja, $M = \lambda x.N$ e $V = \lambda x.N'$. Basta tomar $W = M$. Com efeito, $M = \lambda x.N \twoheadrightarrow_v \lambda x.N = W$, novamente pela reflexividade da relação $\twoheadrightarrow_v$.

- Caso τ:

  Suponhamos que $M \to_s V$. Por hipótese, $M \to_v N$ e $N \to_s V$. Logo, por hipótese de indução, existe V' tal que $N \twoheadrightarrow_v V'$. Basta tomar $W = V'$. Com efeito, de $M \to_v N$ e de $N \twoheadrightarrow_v V'$ obtemos $M \twoheadrightarrow_v V' = W$, pela transitividade da relação $\twoheadrightarrow_v$.

□

Observemos os mesmos exemplos utilizados na secção anterior:

1. Ora, $M = (\lambda x.x)((\lambda y.y)z) \to_{\beta_v} (\lambda x.x)z \to_{\beta_v} z = V$. Vejamos:

$$\dfrac{\dfrac{\dfrac{}{(\lambda y.y)z \to_v z}\,\beta_v}{(\lambda x.x)((\lambda y.y)z) \to_v (\lambda x.x)z}\,\mu \quad \dfrac{\dfrac{}{(\lambda x.x)z \to_v z}\,\beta_v \quad \dfrac{}{z \to_s z}\,\rho}{(\lambda x.x)z \to_s z}\,\tau}{(\lambda x.x)((\lambda y.y)z) \to_s z}\,\tau$$

   Com efeito, $M \to_s V$ e $M \twoheadrightarrow_v V$.

2. Ora, $M = \lambda u.((\lambda x.x)u)((\lambda y.y)z) \to_{\beta_v} \lambda u.((\lambda x.x)u)z \to_{\beta_v} \lambda u.uz = V$. Vejamos:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(\lambda x.x)u \to_v u}\,\beta_v \quad \dfrac{}{u \to_s u}\,\rho}{(\lambda x.x)u \to_s u}\,\tau \quad \dfrac{\dfrac{}{(\lambda y.y)z \to_v z}\,\beta_v \quad \dfrac{}{z \to_s z}\,\rho}{(\lambda y.y)z \to_s z}\,\tau}{((\lambda x.x)u)((\lambda y.y)z) \to_s uz}\,\epsilon}{\lambda u.((\lambda x.x)u)((\lambda y.y)z) \to_s \lambda u.uz}\,\xi$$

   ou

$$\dfrac{\dfrac{\dfrac{}{(\lambda x.x)u \to_v u}\,\beta_v}{((\lambda x.x)u)((\lambda y.y)z) \to_v u((\lambda y.y)z)}\,\nu \quad \dfrac{\dfrac{\dfrac{}{(\lambda y.y)z \to_v z}\,\beta_v}{u((\lambda y.y)z) \to_v uz}\,\mu \quad \dfrac{\dfrac{}{u \to_s u}\,\rho \quad \dfrac{}{z \to_s z}\,\rho}{uz \to_s uz}\,\epsilon}{u((\lambda y.y)z) \to_s uz}\,\tau}{((\lambda x.x)u)((\lambda y.y)z) \to_s uz}\,\tau}{\lambda u.((\lambda x.x)u)((\lambda y.y)z) \to_s \lambda u.uz}\,\xi$$

   Com efeito, $M \to_s V$ e, tomando $W = M$, $M \twoheadrightarrow_v W$.

Definição 44 (Função de avaliação). *Seja $eval_v : \mathcal{P} \rightharpoonup \mathcal{P}$ a função parcial de avaliação de uma linguagem call-by-value definida por:*

$$eval_v(\lambda x.M) = \lambda x.M$$

$$eval_v(MN) = eval_v([V/x]P), \text{ se } eval_v(M) = \lambda x.P \text{ e } eval_v(N) = V$$

Utilizamos a notação $eval_v(M)\downarrow$ para indicar que $eval_v(M)$ está definido.

Observe-se que, se V é um valor fechado, então $eval_v(V) = V$.

Definição 45. *Notamos $eval_v(M) \simeq eval_v(N)$ se as seguintes condições forem satisfeitas:*

- $eval_v(M)\downarrow \iff eval_v(N)\downarrow$;
- $eval_v(M)\downarrow \ \wedge eval_v(N)\downarrow \implies eval_v(M) = eval_v(N)$.

Lema 8. *Sejam $M, N \in \Lambda$ tal que $M \twoheadrightarrow_v N$ e M é um termo-λ fechado. Então,*

$$eval_v(M) \simeq eval_v(N).$$

Demonstração. A prova segue por indução na relação $M \twoheadrightarrow_v N$. Suponhamos que

- Caso β:

  Suponhamos que $(\lambda x.M)V \twoheadrightarrow_v [V/x]M$. Como o redex-$\beta_v$ é fechado, então V é um valor fechado. Pela observação feita anteriormente, daqui segue que $eval_v(V) = V$. Assim, pela definição 44, $eval_v((\lambda x.M)V) = eval_v([V/x]M)$ e, portanto, $eval_v((\lambda x.M)V)\downarrow \iff eval_v([V/x]M)\downarrow$. Suponhamos agora que $eval_v((\lambda x.M)V)\downarrow$ e $eval_v([V/x]M)\downarrow$. Como $eval_v(V) = V$, é direto que $eval_v((\lambda x.M)V) = eval_v([V/x]M)$. Logo, $eval_v((\lambda x.M)V) \simeq eval_v([V/x]M)$.

- Caso ν:

  Suponhamos que $MN \twoheadrightarrow_v M'N$, com $M \twoheadrightarrow_v M'$. Por hipótese de indução, $eval_v(M) \simeq eval_v(M')$, donde segue que $eval_v(M)\downarrow \iff eval_v(M')\downarrow$. Daqui segue que $eval_v(MN)\downarrow \iff eval_v(M'N)\downarrow$. Suponhamos agora que $eval_v(MN)\downarrow$ e $eval_v(M'N)\downarrow$. Em particular, $eval_v(M)\downarrow$ e $eval_v(M')\downarrow$. Falta ver que $eval_v(MN) = eval_v(M'N)$. Seja $eval_v(M) = eval_v(\lambda x.P)$ e $eval_v(N) = V$. Então, $eval_v(MN) = eval_v([V/x]P)$. Como $eval_v(M)$ e $eval_v(M')$ estão definidos, então, por hipótese de indução, $eval_v(M) = eval_v(M')$. Logo, $eval_v(M') = eval_v(\lambda x.P)$ e, por isso, $eval_v(M'N) = eval_v([V/x]P)$, donde segue que $eval_v(MN) = eval_v(M'N)$. Desta forma, $eval_v(MN) \simeq eval_v(M'N)$.

- Caso μ:

  Suponhamos que $VM \twoheadrightarrow_v VM'$, com $M \twoheadrightarrow_v M'$. Por hipótese de indução, $eval_v(M) \simeq eval_v(M')$, donde segue que $eval_v(M)\downarrow \iff eval_v(M')\downarrow$. Daqui segue que $eval_v(VM)\downarrow$

$\iff$ $eval_v(VM') \downarrow$. Suponhamos agora que $eval_v(VM) \downarrow$ e $eval_v(VM') \downarrow$. Em particular, $eval_v(M) \downarrow$ e $eval_v(M') \downarrow$. Falta ver que $eval_v(VM) = eval_v(VM')$. Seja $eval_v(V) = eval_v(\lambda x.P)$ e $eval_v(M) = W$. Assim sendo, $eval_v(VM) = eval_v([W/x]P)$. Como $eval_v(M)$ e $eval_v(M')$ estão definidos, então, por hipótese de indução, $eval_v(M) = eval_v(M')$. Logo, $eval_v(M') = W$ e, consequentemente, $eval_v(VM') = eval_v([W/x]P)$, donde segue que $eval_v(VM) = eval_v(VM')$. Desta forma, $eval_v(VM) \simeq eval_v(VM')$.

- Caso ρ:

  Suponhamos que $M \twoheadrightarrow_v M$. É direto que $eval_v(M)\downarrow \iff eval_v(M)\downarrow$ e que, se $eval_v(M)\downarrow$, então $eval_v(M) = eval_v(M)$. Logo, $eval_v(M) \simeq eval_v(M)$.

- Caso τ:

  Suponhamos que $M \twoheadrightarrow_v P$, por consequência de $M \twoheadrightarrow_v N$ e $N \twoheadrightarrow_v P$. Por hipótese de indução, $eval_v(M) \simeq eval_v(N)$ e $eval_v(N) \simeq eval_v(P)$. Daqui segue, respetivamente, que $eval_v(M) \downarrow \iff eval_v(N) \downarrow$ e que $eval_v(N) \downarrow \iff eval_v(P) \downarrow$. Logo, $eval_v(M) \downarrow \iff eval_v(P)\downarrow$. Suponhamos que $eval_v(M)\downarrow$ e $eval_v(P)\downarrow$. Então, por hipótese de indução, $eval_v(N) \downarrow$, $eval_v(M) = eval_v(N)$ e $eval_v(N) = eval_v(P)$. Logo, $eval_v(M) = eval_v(P)$. Desta forma, $eval_v(M) \simeq eval_v(P)$.

□

Teorema 11. *Sejam $M, V \in \Lambda$ tal que M é um termo fechado. Então,*

$$(eval_v(M)\downarrow \wedge eval_v(M) = V) \iff M \twoheadrightarrow_v V.$$

Demonstração. Suponhamos, primeiramente, que $eval_v(M) \downarrow$ e $eval_v(M) = V$. Queremos mostrar que $M \twoheadrightarrow_v V$. De $eval_v(M) = V$ segue que existe uma prova de $eval_v(M) = V$ usando as regras que definem $eval_v(M)$ com tamanho n. A prova segue por indução nesse tamanho.

- Caso $eval_v(\lambda x.P) = \lambda x.P$:

  É direto que $\lambda x.P \twoheadrightarrow_v \lambda x.P$, pela reflexividade de $\twoheadrightarrow_v$.

- Caso $eval_v(PQ) = eval([W/x]N)$:

  Suponhamos que existe uma prova de $eval_v(P) = \lambda x.N$ com tamanho n_1, uma prova de $eval_v(Q) = W$ com tamanho n_2 e uma prova de $eval_v([W/x]N) = V$ com tamanho n_3.

Daqui segue que $eval_v(PQ) = V$ tem tamanho $n = n_1 + n_2 + n_3 + 1$. Logo, por hipótese de indução, $P \twoheadrightarrow_v \lambda x.N$, $Q \twoheadrightarrow_v W$ e $[W/x]N \twoheadrightarrow_v V$. Por definição de $\rightarrow_v$, $PQ \rightarrow_v (\lambda x.N)Q \rightarrow_v (\lambda x.N)W \rightarrow_v [W/x]N$, donde segue que $PQ \twoheadrightarrow_v [W/x]N \twoheadrightarrow_v V$ e, portanto, $PQ \twoheadrightarrow_v V$.

Reciprocamente, suponhamos que $M \twoheadrightarrow_v V$. Queremos mostrar que $eval_v(M)\downarrow$ e $eval_v(M) = V$. De $M \twoheadrightarrow_v V$ segue que $eval_v(M) \simeq eval_v(V)$, pelo lema 8. Como $eval_v(V) = V$, então $eval_v(V)\downarrow$ e, consequentemente, $eval_v(M)\downarrow$. Logo, $eval_v(M) = eval_v(V) = V$.

□

Este teorema expressa como a estratégia de avaliação $eval_v$ da linguagem *call-by-value* é modelada pela estratégia de redução $\twoheadrightarrow_v$ no cálculo-λ_v.

3.3 CBN vs CBV

3.3.1 *Call-by-value* CPS

Plotkin simula a linguagem *call-by-value* através da *call-by-name*, recorrendo às "continuações".

As demonstrações dos resultados que vamos ver podem ser encontradas em [14].

Definição 46 (Tradução de Plotkin). *O mapeamento* $\overline{\cdot} : \Lambda \rightarrow \Lambda$*, que envia termos-*$\lambda$ *da linguagem call-by-value para termos da linguagem call-by-name, é definido recursivamente por:*

$$\begin{aligned} \overline{V} &= \lambda k.k\Psi(V) \\ \overline{MN} &= \lambda k.\overline{M}(\lambda y.\overline{N}(\lambda z.yzk)) \\ \Psi(x) &= x \\ \Psi(\lambda x.M) &= \lambda x.\overline{M} \end{aligned}$$

Teorema 12 (Indiferença). *Seja* $M \in \mathcal{P}$*. Então,*

$$eval_v(\overline{M}\mathbf{I}) = eval_n(\overline{M}\mathbf{I}).$$

Informalmente, este teorema expressa que, dado o combinador $\mathbf{I}$ como continuação inicial, o resultado das avaliações *call-by-name* e *call-by-value* do termo traduzido é o mesmo.

Teorema 13 (Simulação). *Seja* $M \in \mathcal{P}$*. Então,*

$$\Psi(eval_v(M)) = eval_n(\overline{M}\mathbf{I}).$$

Informalmente, este teorema expressa que, dado o combinador $\mathbf{I}$ como continuação inicial, a avaliação *call-by-name* do termo traduzido corresponde à tradução do valor obtido pela avaliação *call-by-value* do termo original, pelo que a estratégia de avaliação *call-by-value* é corretamente simulada.

Teorema 14 (Tradução). *Sejam $M, N \in \Lambda$. Então,*

$$(i) \qquad M =_{\beta_v} N \implies \overline{M} =_{\beta_v} \overline{N};$$
$$(ii) \qquad \overline{M} =_{\beta_v} \overline{N} \iff \overline{M} =_{\beta} \overline{N}.$$

Informalmente, este teorema expressa como as relações β_v e β são preservadas pela tradução. Além disso, estabelece a incompletude do cálculo Plotkin para a semântica CPS, uma vez que:

$$\overline{M} =_{\beta} \overline{N} \not\Longrightarrow M =_{\beta_v} N.$$

3.3.2 *Call-by-name Thunks*

A simulação da linguagem *call-by-name* é bem mais simples, dado que $\beta_v \subseteq \beta$.

Hatcliff e Danvy fazem esta simulação utilizando *thunks*, em alternativa às continuações, preservando os resultados da Indiferença, Simulação e Tradução de Plotkin [10].

Definição 47 (Tradução de Hatcliff-Danvy). *O mapeamento $\mathcal{T}(\cdot) : \Lambda \to \Lambda$, que envia termos da linguagem call-by-name para termos da linguagem call-by-value, é definido recursivamente por:*

$$\mathcal{T}(x) = x\mathbf{I}$$
$$\mathcal{T}(\lambda x.M) = \lambda x.\mathcal{T}(M)$$
$$\mathcal{T}(MN) = \mathcal{T}(M)(\lambda z.\mathcal{T}(N)), \text{ onde } z \notin LIV(N)$$

Observemos que a tradução de uma aplicação de dois termos M e N resulta numa aplicação de dois termos onde, agora, temos necessariamente um valor na posição de argumento, $\mathcal{T}(M)(\lambda z.\mathcal{T}(N))$. Desta forma, podemos aplicar a regra β_v. Em particular e crucialmente, este valor é uma abstração vacuosa. Assim, aplicando a abstração a um termo, como é o caso do combinador $\mathbf{I}$, obtemos apenas o corpo da abstração $\mathcal{T}(N)$.

Vejamos melhor com um exemplo. Seja $M = \lambda x.x$ e $N = yu$, por exemplo. Com efeito, $MN = (\lambda x.x)(yu) \to_n yu$ em λ e $\mathcal{T}(MN) = (\lambda x.xI)(\lambda z.yu) \to_v (\lambda z.yu)I \to_v yu$ em λ_v.

Teorema 15 (**Indiferença**). *Seja $M \in \mathcal{P}$. Então,*

$$eval_v(\mathcal{T}(M)) = eval_n(\mathcal{T}(M)).$$

Teorema 16 (**Simulação**). *Seja $M \in \mathcal{P}$. Então,*

$$\mathcal{T}(eval_n(M)) = eval_v(\mathcal{T}(M)).$$

Teorema 17 (**Tradução**). *Sejam $M, N \in \Lambda$. Então,*

$$(i) \qquad M =_\beta N \iff \mathcal{T}(M) =_\beta \mathcal{T}(N);$$
$$(ii) \qquad \mathcal{T}(M) =_\beta \mathcal{T}(N) \iff \mathcal{T}(M) =_{\beta_v} \mathcal{T}(N).$$

3.4 Extensão de Sabry-Felleisen

Consideremos o contradomínio da tradução CPS de Fischer, que pode ser encontrada em [15].

Definição 48 (**termos-λ em CPS**). *Seja k uma variável de continuação. Seja $\mathcal{B} = \mathcal{A} \cup \{k\}$ e $\mathcal{B}^*$ o conjunto de todas as palavras em $\mathcal{B}$. O conjunto dos termos-λ em CPS, representado por $cps(\Lambda)$, é definido indutivamente sobre $\mathcal{B}^*$ por:*

$$M, N \in cps(\Lambda) ::= KW$$
$$V, W ::= x \mid \lambda k.K$$
$$K ::= k \mid WK \mid \lambda x.M$$

Sabry e Felleisen estendem o sistema λ_v de Plotkin [15], adicionando um conjunto de relações inspiradas na tradução CPS, representado por X, tal que:

$$M =_{\beta_v X} N \iff cps(M) =_{\beta\eta} cps(N),$$

onde as relações (β) e (η) podem ser decompostas nas relações dadas sobre a forma das seguintes regras:

$$(\beta_w) \qquad (\lambda x.P)W \to [W/x]P$$
$$(\beta_k) \qquad (\lambda k.K_1)K_2 \to [K_2/k]K_1$$
$$(\eta_w) \qquad \lambda k.Wk \to W$$

$$(\eta_k) \qquad \lambda x.Kx \to K, \text{ se } x \notin LIV(K)$$

Desta forma, estabelecem a correção e a completude desta extensão, que é representada por λ_v++.

Para isso, com base na tradução CPS de Fischer, desenvolvem e debruçam-se sobre as traduções $C_k : \Lambda \to cps(\Lambda)$ e $C^{-1} : cps(\Lambda) \to \Lambda$, obtendo então X.

Definição 49 (**Contextos de avaliação**). *Seja $\mathcal{C} = \mathcal{A} \cup \{[\,]\}$ e $\mathcal{C}^*$ o conjunto de todas as palavras em $\mathcal{C}$. O conjunto de contextos de avaliação, representado por $\mathbf{ContAv}$, é definido indutivamente sobre $\mathcal{C}^*$ por:*

$$E ::= [\,] \mid VE \mid EN$$

Definição 50 (**termo-λ** $E[\![M]\!]$). *Seja $M \in \Lambda$. Definimos $E[\![M]\!] \in \Lambda$ recursivamente em E por:*

$$[\,][\![M]\!] = M$$
$$(VE)[\![M]\!] = V(E[\![M]\!])$$
$$(EN)[\![M]\!] = (E[\![M]\!])N$$

Informalmente, um programa, isto é, um termo fechado, é decomposto num contexto de avaliação E e num redex-β_v externo mais à esquerda. Após reduzir o redex-β_v, preenchemos os buracos do contexto de avaliação com o respetivo reduto. Assim, a avaliação pode ser definida da seguinte forma:

$$E[\![(\lambda x.M)V]\!] \to_v E[\![[V/x]M]\!].$$

De seguida, vamos apresentar as regras de redução que constituem o conjunto X. É de notar que no artigo original, onde foram apresentadas pela primeira vez, os autores utilizaram uma outra sintaxe, cuja gramática contém a classe dos contextos de avaliação.

Uma vez que, mais à frente, vamos comparar este cálculo com o cálculo computacional, que estudaremos no próximo capítulo, é vantajoso apresentar as regras com a sintaxe que vamos utilizar.

Definição 51 (**Relações de X**). *Seja X o conjunto de relações binárias em Λ apresentadas sobre a forma das seguintes regras:*

$$(\beta_{lift}) \qquad E[\![(\lambda x.M)N]\!] \to (\lambda x.E[\![M]\!])N, \text{ se } E \neq [\,]$$
$$(\beta_{flat}) \qquad (zN)L \to (\lambda x.xL)(zN), \text{ se } x \notin LIV(L)$$
$$(\beta_{id}) \qquad (\lambda x.x)M \to M$$

$$(\beta_\Omega) \qquad (\lambda x.E[\![yx]\!])M \to E[\![yM]\!], \text{ se } x \notin LIV(E[\![y]\!])$$

$$(\eta_v) \qquad (\lambda x.Vx) \to V, \text{ se } x \notin LIV(V)$$

Agora, atentemos à seguinte sintaxe.

Definição 52 (**termos-λ com contextos de avaliação**). *O conjunto de termos-λ com contextos de avaliação, Λ_E, é definido indutivamente sobre C^* por:*

$$M, N \in \Lambda_E ::= V \mid E[VW]$$

$$V, W ::= x \mid \lambda x.M$$

$$E \in ContAv ::= [\,] \mid VE \mid EN$$

É de notar que os conjuntos de termos Λ e Λ_E são equivalentes, sendo que deixaremos esta prova para trabalho futuro. Posto isto, no decurso deste trabalho, representaremos o conjunto de termos-λ com contextos de avaliação por Λ.

Com o objetivo de definir as traduções C_k e C^{-1}, consideremos o conjunto dos contextos de avaliação, $ContAv$, definidos da seguinte forma:

$$E ::= [\,] \mid E[V[\,]] \mid E[[\,]N].$$

Esta apresentação do conjunto de contextos de avaliação e a apresentação considerada na definição 49 são equivalentes. Devido à escassez do tempo, deixaremos esta prova para trabalho futuro.

Definição 53 (**Tradução C_k**). *O mapeamento $C_k(\cdot) : \Lambda \to cps(\Lambda)$ é definido recursivamente da seguinte forma:*

- *Para cada $M \in \Lambda$, $C_k(M)$ um termo em CPS:*

$$C_k(V) = k(\Phi(V))$$

$$C_k(E[xV]) = (xK_k(E))\Phi(V)$$

$$C_k(E[(\lambda x.M)V]) = (\lambda x.C_k(E[M]))\Phi(V)$$

- *Para cada $V \in \Lambda$, $\Phi(V)$ um valor em CPS:*

$$\Phi(x) = x$$

$$\Phi(\lambda x.M) = \lambda k.\lambda x.C_k(M)$$

- *Para cada $E \in ContAv$, $K_k(E)$ um contexto em CPS:*

$$K_k([\,]) = k$$
$$K_k(E[x[\,]]) = xK_k(E)$$
$$K_k(E[(\lambda x.M)[\,]]) = \lambda x.C_k(E[M])$$
$$K_k(E[[\,]N]) = \lambda z.C_k(E[zN])$$

Definição 54 (**Tradução** C^{-1})**.** *O mapeamento $C^{-1}(\cdot) : cps(\Lambda) \longrightarrow \Lambda$ é definido recursivamente do seguinte modo:*

- *Para cada $M \in cps(\Lambda)$, $C^{-1}(M)$ um termo de λ_v++:*

$$C^{-1}(KW) = K^{-1}(K)\Phi^{-1}(W)$$

- *Para cada $V \in cps(\Lambda)$, $\Phi(V)$ um valor de λ_v++:*

$$\Phi^{-1}(x) = x$$
$$\Phi^{-1}(\lambda k.k) = \lambda x.x$$
$$\Phi^{-1}(\lambda k.WK) = \lambda x.C^{-1}((KW)x)$$
$$\Phi^{-1}(\lambda kx.M) = \lambda x.C^{-1}(M)$$

- *Para cada $K \in cps(\Lambda)$, $K^{-1}(M)$ um contexto de avaliação de λ_v++:*

$$K^{-1}(k) = [\,]$$
$$K^{-1}(xK) = K^{-1}(K)[x[\,]]$$
$$K^{-1}((\lambda k.K)K') = K^{-1}([K'/x]K)$$
$$K^{-1}(\lambda x.M) = (\lambda x.C^{-1}(M))[\,]$$

Por fim, os autores concluem que as traduções formam uma equivalência equacional. Seja $\lambda_{cps'}$ o cálculo contituído pelo conjunto de termos $cps(\Lambda)$ e pelas relações (β) e (η). A demonstração do resultado seguinte pode ser encontrada em [15].

Teorema 18 (**Correspondência Equacional**)**.** *As traduções C_k e C^{-1} formam uma equivalência equacional entre λ_v++ e $\lambda_{cps'}$, isto é,*

- $M =_{\beta_v X} N \implies C_k(M) =_{\beta\eta} C_k(N)$;
- $M =_{\beta\eta} N \implies C^{-1}(M) \twoheadrightarrow_{\beta_v X} C^{-1}(N)$;
- $M =_{\beta_v X} C^{-1}(C_k(M))$;
- $C_k(C^{-1}(M) =_{\beta\eta} M$.

Estas traduções não formam uma conexão de Galois porque não satisfazem, unicamente, a condição $C_k(C^{-1}(M)) \twoheadrightarrow_{\beta\eta} M$, para qualquer $M \in cps(\Lambda)$.

De uma forma geral, compreendemos como as linguagens de programação CBN e CBV são modeladas pelo cálculo-lambda usual e pelo cálculo de Plotkin, respetivamente. Além disso, vimos ainda a extensão de Sabry-Felleisen, uma extensão ao cálculo de Plotkin, que resulta de adicionar regras de redução ao cálculo, que visa colmatar o problema da incompletude do cálculo de Plotkin para a semântica CPS.

Capítulo 4

Cálculo computacional

4.1 Motivação

Primeiramente, Moggi desenvolveu o cálculo-λ_{ml}, que distingue valores de computações e, portanto, modela propriedades denotacionais das linguagens de programação.

Inspirado neste cálculo-λ_{ml}, Moggi desenvolve, posteriormente, o cálculo computacional, representado por λ_c, que tem como base a semântica denotacional das linguagens da programação, utilizando *monads* [13].

4.2 O sistema

Sabry e Wadler estendem o sistema λ_v ao sistema λ_c e definem o cálculo-λ_{cps} [16].

Definição 55 (termos-λ_c). *Seja $\mathcal{D} = \mathcal{A} \cup \{let,\ in,\ =\}$ e $\mathcal{D}^*$ o conjunto de todas as palavras em $\mathcal{D}$. O conjunto de termos-λ_c, representado por Λ_c, é definido indutivamente sobre $\mathcal{D}^*$ por:*

$$M, N \in \Lambda_c ::= V \mid (MN) \mid (let\ x = N\ in\ M)$$
$$V, W ::= x \mid (\lambda x.M)$$

Um termo da forma $(let\ x = N\ in\ M)$ diz-se uma expressão-let, sendo que o termo M diz-se o corpo dessa expressão e as ocorrências livres de x em M estão ligadas na expressão.

Tal como no cálculo-λ, convencionamos que os nomes das variáveis livres e ligadas de um termo são diferentes. Assumiremos também que dois termos iguais-α, isto é, que diferem apenas no nome das variáveis ligadas, são iguais.

Definição 56 (**Substituição**). *A substituição das ocorrências livres de x por N em M, representada por $[N/x]M$, é definida recursivamente por:*

$$[N/x]x = N$$

$$[N/x]y = y, \text{ se } y \neq x$$

$$[N/x](\lambda y.P) = \lambda y.[N/x]P$$

$$[N/x](PQ) = [N/x]P[N/x]Q$$

$$[N/x](let\ y = P\ in\ Q) = (let\ y = [N/x]P\ in\ [N/x]Q)$$

Definição 57 (**Reduções**). *Uma relação binária $R \subseteq \Lambda_c^2$ diz-se compatível se:*

$$\frac{(M,N) \in R}{(\lambda x.M, \lambda x.N) \in R}\ \xi \qquad \frac{(M,N) \in R}{(MP,NP) \in R}\ \nu \qquad \frac{(M,N) \in R}{(PM,PN) \in R}\ \mu$$

$$\frac{(M,N) \in R}{(let\ x = M\ in\ P,\ let\ x = N\ in\ P) \in R}\ let_\nu \qquad \frac{(M,N) \in R}{(let\ x = P\ in\ M,\ let\ x = P\ in\ N) \in R}\ let_\mu$$

Dada a relação binária $R \subseteq \Lambda_c^2$ compatível, definem-se as seguintes relações binárias em Λ_c, de forma análoga ao foi feito na secção 2.1.2 para relações binárias em Λ:

- *o fecho compatível de R, representado por $\rightarrow_R$;*
- *o fecho reflexivo e transitivo de $\rightarrow_R$, representado por $\twoheadrightarrow_R$;*
- *o fecho de equivalência de $\rightarrow_R$, representado por $=_R$.*

Definição 58 (**Relação β_v**). *Seja $\beta_v \subseteq \Lambda_c^2$ a relação apresentada sobre a forma da seguinte regra:*

$$(\beta_v) \qquad (\lambda x.M)V \rightarrow [V/x]M$$

Definição 59 (**Relação let**). *Seja $let \in \Lambda_c^2$ a relação binária apresentada sobre a forma das seguintes regras:*

$$(id) \qquad (let\ x = M\ in\ x) \rightarrow M$$

$$(comp) \qquad (let\ y = (let\ x = M\ in\ N)\ in\ P) \rightarrow (let\ x = M\ in\ (let\ y = N\ in\ P))$$

$$(let_v) \qquad (let\ x = V\ in\ M) \rightarrow [V/x]M$$

$$(let_1) \qquad NM \rightarrow (let\ x = N\ in\ xM), \text{ onde } N \text{ não é um valor}$$

$$(let_2) \qquad VN \rightarrow (let\ x = N\ in\ Vx), \text{ onde } N \text{ não é um valor}$$

É de notar que a regra let_2 reduz um redex-β da forma $(\lambda x.M)N$, onde N não é um valor, para um redex-β_v no corpo de let.

O significado das expressões-let faz corresponder a $(\lambda x.M)N$ no cálculo-λ usual. De facto, se a regra β estivesse disponível e $(let\, x = N\, in\, M)$ fosse substituído por $(\lambda x.M)N$, então a relação let seria um subconjunto de $=_\beta$.

Vejamos, por exemplo, o caso do redex-β $(\lambda x.x)(yz)$.

$$\begin{aligned} \text{Ora, } (\lambda x.x)(yz) &\rightarrow_{let_2} (let\, u = yz\, in\, (\lambda x.x)u) \\ &\rightarrow_{\beta_v} (let\, u = yz\, in\, u) \\ &\rightarrow_{id} yz. \end{aligned}$$

Definição 60 (**Relação** η_v)**.** *Seja $\eta_v \in \Lambda_c^2$ a relação binária apresentada sobre a forma da seguinte regra:*

$$(\eta_v) \qquad \lambda x.Vx \rightarrow V, \text{ se } x \notin LIV(V)$$

Definição 61 (**Cálculo-**λ_{cps})**.** *O conjunto dos termos de λ_{cps}, representado por Λ_{cps}, é definido indutivamente sobre $\mathcal{B}^*$ por:*

$$\begin{aligned} M, N \in \Lambda_{cps} &::= KW \mid VWK \\ V, W &::= x \mid \lambda xk.M \\ K &::= k \mid \lambda x.M \end{aligned}$$

As relações binárias deste sistema são apresentadas sobre a forma das seguintes regras:

$$\begin{aligned} &(\beta_v) && (\lambda xk.M)VK \rightarrow [K/k][V/x]M \\ &(\eta_v) && \lambda xk.Vxk \rightarrow V, \text{ se } x \notin LIV(V) \\ &(\beta_{let}) && (\lambda x.M)V \rightarrow [V/x]M \\ &(\eta_{let}) && \lambda x.Kx \rightarrow K, \text{ se } x \notin LIV(K) \end{aligned}$$

Definição 62 (**Tradução** f)**.** *O mapeamento $f(\cdot) : \Lambda_c \rightarrow \Lambda_{cps}$ é definido recursivamente da seguinte forma:*

- *Para cada $M \in \Lambda_c$, cada $K \in \lambda_{cps}$, $M : K$ um termo de λ_{cps}:*

$$V : K = K\Psi(V)$$

$$MN : K = M : (\lambda x.((xN) : K)), \text{ onde M não é um valor}$$

$$VN : K = N : (\lambda y.((Vy) : K)), \text{ onde N não é um valor}$$

$$VW : K = \Psi(V)\Psi(W)K$$

$$(let\, x = M\, in\, N) : K = M : (\lambda x.(N : K))$$

- *Para cada $V \in \Lambda_c$, $\Psi(V)$ um valor de λ_{cps}:*

$$\Psi(x) = x$$

$$\Psi(\lambda x.M) = \lambda xk.f(M)$$

Por fim, define-se:

$$f(M) = M : k$$

Definição 63 (**Tradução** *g*)**.** *Seja $C ::= [\,] \mid let\, x = [\,]\, in\, P$ em λ_c. Para cada $M \in \Lambda_c$, $C[M]$ é um termo-λ_c. O mapeamento $g(\cdot) : \Lambda_{cps} \to \Lambda_c$ é definido recursivamente da seguinte forma:*

- *Para cada $M \in \Lambda_{cps}$, $g(M)$ um termo de λ_c:*

$$g(KV) = \Phi(K)[\Psi(V)]$$

$$g(VWK) = \Phi(K)[\Psi(V)\Psi(W)]$$

- *Para cada $V \in \Lambda_{cps}$, $\Psi(V)$ um termo de λ_c:*

$$\Psi(x) = x$$

$$\Psi(\lambda xk.M) = \lambda x.g(M)$$

- *Para cada $K \in \Lambda_{cps}$, $\Phi(K) \in C$:*

$$\Phi(k) = [\,]$$

$$\Phi(\lambda x.M) = let\, x = [\,]\, in\, g(M)$$

Por fim, estabelecem a reflexão em λ_c de λ_{cps}, que permite concluir que o cálculo computacional é completo para a semântica CPS. A demonstração deste resultado está apresentada em [16]. De modo a simplificar a notação, utilizaremos $M \twoheadrightarrow_c N$ para representar $M \twoheadrightarrow_{\beta_v let \eta_v} N$ e, analogamente, $M \twoheadrightarrow_{cps} N$ para representar $M \twoheadrightarrow_{\beta_v \beta_{let} \eta_v \eta_{let}} N$.

Teorema 19 (Reflexão). *As traduções f e g formam uma reflexão em λ_c de λ_{cps}, isto é,*

- $M \twoheadrightarrow_c N \implies f(M) \twoheadrightarrow_{cps} f(N)$;
- $M \twoheadrightarrow_{cps} N \implies g(M) \twoheadrightarrow_c g(N)$;
- $M \twoheadrightarrow_c g(f(M))$;
- $f(g(M)) = M$.

4.3 Relação com a extensão de Sabry-Felleisen

De modo a simplificar a notação, utilizaremos $M =_c N$ para representar $M =_{\beta_v let \eta_v} N$ e, analogamente, $M =_{v++} N$ para representar $M =_{\beta_v X} N$.

Definição 64. *O mapeamento $h(\cdot) : \Lambda_c \to \Lambda$, que envia termos de λ_c para termos de λ_{v++}, é definido indutivamente por:*

$$h(x) = x$$
$$h(\lambda x.M) = \lambda x.h(M)$$
$$h(MN) = h(M)h(N)$$
$$h(let\ x = N\ in\ M) = (\lambda x.h(M))h(N)$$

Lema 9. *Sejam $M, V \in \Lambda_c$. Então,*

$$h([V/x]M) = [h(V)/x]h(M).$$

Demonstração. A prova segue por indução em $M \in \Lambda_c$.

- Caso $M = x$:

$$\begin{aligned} h([V/x]x) &= h(V) \\ &= [h(V)/x]x \\ &= [h(V)/x]h(x). \end{aligned}$$

- Caso $M = y$:

$$h([V/x]y) = h(y)$$

$$\begin{aligned} &= y \\ &= [h(V)/x]y \\ &= [h(V)/x)]h(y). \end{aligned}$$

- Caso $M = \lambda y.N$:

$$\begin{aligned} h([V/x](\lambda y.N)) &= h(\lambda y.[V/x]N) \\ &= \lambda y.h([V/x]N) \\ &= \lambda y.[h(V)/x]h(N), \text{ por H.I. em } N, \\ &= [h(V)/x](\lambda y.h(N)) \\ &= [h(V)/x]h(\lambda y.N). \end{aligned}$$

- Caso $M = PQ$:

$$\begin{aligned} h([V/x](PQ)) &= h([V/x]P[V/x]Q) \\ &= h([V/x]P)h([V/x]Q) \\ &= [h(V)/x]h(P)[h(V)/x]h(Q), \text{ por H.I. em } P \text{ e } Q, \\ &= [h(V)/x](h(P)h(Q)) \\ &= [h(V)/x]h(PQ). \end{aligned}$$

- Caso $M = (let\ y = N\ in\ P)$:

$$\begin{aligned} h([V/x](let\ y = N\ in\ P)) &= h(let\ y = [V/x]N\ in\ [V/x]P) \\ &= (\lambda y.h([V/x]P))h([V/x]N) \\ &= (\lambda y.[h(V)/x]h(P))[h(V)/x]h(N), \text{ por} \\ &\quad \text{H.I. em } P \text{ e } N, \\ &= [h(V)/x](\lambda y.h(P))[h(V)/x]h(N) \\ &= [h(V)/x]((\lambda y.h(P))h(N)) \\ &= [h(V)/x]h(let\ y = N\ in\ P). \end{aligned}$$

□

Proposição 2. *Sejam $M, N \in \Lambda_c$. Então,*

$$M =_c N \implies h(M) =_{v++} h(N).$$

Demonstração. Suponhamos primeiramente que $M =_c N$. Queremos mostrar que $h(M) =_{v++} h(N)$. A prova segue por indução na relação $=_c$.

- Caso β_v:

  Queremos mostrar que $h((\lambda x.M)V) =_{v++} h([V/x]M)$.

$$\begin{aligned} h((\lambda x.M)V) &= (\lambda x.h(M))h(V) \\ &\to_{\beta_v} [h(V)/x]h(M), \text{ pois } h(V) \text{ é um valor,} \\ &= h([V/x]M), \text{ pelo lema 9.} \end{aligned}$$

- Caso η_v:

  Queremos mostrar que $h(\lambda x.Vx) =_{v++} h(V)$.

$$\begin{aligned} h(\lambda x.Vx) &= \lambda x.h(V)h(x) \\ &= \lambda x.h(V)x \\ &\to_{\eta_v} h(V), \text{ pois } h(V) \text{ é um valor.} \end{aligned}$$

- Caso id:

  Queremos mostrar que $h(let\ x = M\ in\ x) =_{v++} h(M)$.

$$\begin{aligned} h(let\ x = M\ in\ x) &= (\lambda x.h(x))h(M) \\ &= (\lambda x.x)h(M) \\ &\to_{\beta_{id}} h(M). \end{aligned}$$

- Caso $comp$:

  Queremos mostrar que $h(let\ y = (let\ x = M\ in\ N)\ in\ P) =_{v++} h(let\ x = M\ in\ (let\ y = N\ in\ P))$.

$$\begin{aligned} h(let\ y = (let\ x = M\ in\ N)\ in\ P) &= (\lambda y.h(P))h(let\ x = M\ in\ N) \\ &= (\lambda y.h(P))((\lambda x.h(N))h(M)) \end{aligned}$$

$$
\begin{array}{rll}
 & = & E[\![(\lambda x.h(N))h(M)]\!], \text{ com} \\
 & & E = (\lambda y.h(P))E' \text{ e } E' = [\,], \\
\rightarrow_{\beta_{lift}} & & (\lambda x.E[\![h(N)]\!])h(M) \\
 & = & (\lambda x.(\lambda y.h(P))h(N))h(M) \\
 & = & (\lambda x.h(let\ y = N\ in\ P))h(M) \\
 & = & h(let\ x = M\ in\ (let\ y = N\ in\ P))
\end{array}
$$

- Caso let_v:

  Queremos mostrar que $h(let\ x = V\ in\ M) =_{v++} h([V/x]M)$.

$$
\begin{array}{rll}
h(let\ x = V\ in\ M) & = & (\lambda x.h(M))h(V) \\
 & \rightarrow_{\beta_v} & [h(V)/x]h(M), \text{ pois } h(V) \text{ é um valor,} \\
 & = & h([V/x]M).
\end{array}
$$

- Caso let_1:

  Queremos mostrar que $h(MN) =_{v++} h(let\ x = M\ in\ xN)$, onde N não é um valor.

$$
\begin{array}{rll}
h(MN) = h(M)h(N) & \leftarrow_{\beta_{id}} & ((\lambda x.x)h(M))h(N) \\
 & = & E[\![(\lambda x.x)h(M)]\!], \text{ com } E = E'h(N) \text{ e } E' = [\,] \\
 & \rightarrow_{\beta_{lift}} & (\lambda x.E[\![x]\!])h(M) \\
 & = & (\lambda x.xh(N))h(M) \\
 & = & (\lambda x.h(xN))h(M) \\
 & = & h(let\ x = M\ in\ xN).
\end{array}
$$

- Caso let_2:

  Queremos mostrar que $h(VN) =_{v++} h(let\ x = N\ in\ Vx)$, onde N não é um valor.

$$
\begin{array}{rll}
h(VN) & = & h(V)h(N) \\
 & \leftarrow_{\eta_v} & (\lambda x.h(V)x)h(N), \text{ pois } x \notin LIV(V), \\
 & = & (\lambda x.h(Vx))h(N) \\
 & = & h(let\ x = N\ in\ Vx).
\end{array}
$$

- Caso ν:

  Queremos mostrar que $h(MP) =_{v}{}_{++} h(NP)$.

  Por hipótese de indução, $h(M) =_{v}{}_{++} h(N)$. Daqui segue, pela regra ν, que $h(M)Q =_{v}{}_{++} h(N)Q$. Tomando $Q = h(P)$, obtemos $h(M)h(P) =_{v}{}_{++} h(N)h(P)$, pelo que $h(MP) =_{v}{}_{++} h(NP)$.

- Caso μ:

  Análogo ao caso anterior, segundo a regra μ.

- Caso ξ:

  Queremos mostrar que $h(\lambda x.M) =_{v}{}_{++} h(\lambda x.M)$.
  Por hipótese de indução, $h(M) =_{v}{}_{++} h(N)$. Daqui segue, pela regra ξ, que $\lambda x.h(M) =_{v}{}_{++} \lambda x.h(N)$, pelo que $h(\lambda x.M) =_{v}{}_{++} h(\lambda x.M)$.

- Caso let_{μ}:

  Queremos mostrar que $h(let\, x = P\, in\, M) =_{v}{}_{++} h(let\, x = P\, in\, N)$, ou seja, temos que provar que $(\lambda x.h(M))h(P) =_{v}{}_{++} (\lambda x.h(N))h(P)$.
  Por hipótese de indução, $h(M) =_{v}{}_{++} h(N)$. Daqui segue que $\lambda x.h(M) =_{v}{}_{++} \lambda x.h(N)$, donde $(\lambda x.h(M))Q =_{v}{}_{++} (\lambda x.h(N))Q$, respetivamente pelas regras ξ e ν. Tomando $Q = h(P)$, obtemos o resultado pretendido.

- Caso let_{ν}:

  Queremos mostrar que $h(let\, x = M\, in\, P) =_{v}{}_{++} h(let\, x = N\, in\, P)$, ou seja, temos que provar que $(\lambda x.h(P))h(M) =_{v}{}_{++} (\lambda x.h(P))h(N)$.
  Por hipótese de indução, $h(M) =_{v}{}_{++} h(N)$. Daqui segue que $Qh(M) =_{v}{}_{++} Qh(N)$, pela regra μ. Tomando $Q = \lambda x.h(P)$, obtemos o resultado pretendido.

- Caso ρ:

  Queremos mostrar que $h(M) =_{v}{}_{++} h(M)$.
  É direto pela regra ρ.

- Caso τ:

Queremos mostrar que $h(M) =_{v++} h(P)$.

Por hipótese de indução, $h(M) =_{v++} h(N)$ e $h(N) =_{v++} h(P)$. É direto que $h(M) =_{v++} h(P)$, pela regra τ.

- Caso σ:

  Queremos mostrar que $h(N) =_{v++} h(M)$.

  Por hipótese de indução, $h(M) =_{v++} h(N)$. É direto que $h(N) =_{v++} h(M)$, pela regra σ.

□

Definição 65 (**Contextos de avaliação de** λ_c)**.** *Sejam $V, N \in \Lambda_c$. Os contextos de avaliação E em λ_c são definidos indutivamente por:*

$$E ::= [\,] \mid VE \mid EN$$

Definição 66 (**termo-**λ_c $E[\![M]\!]$)**.** *Seja $M \in \Lambda_c$. Definimos $E[\![M]\!] \in \Lambda_c$ por recursão em E da seguinte forma:*

$$[\,][\![M]\!] = M$$
$$(VE)[\![M]\!] = V(E[\![M]\!])$$
$$(EN)[\![M]\!] = (E[\![M]\!])N$$

Lema 10. *Sejam $M, N \in \Lambda_c$. Então,*

$$(let\ x = N\ in\ E[\![M]\!]) =_c E[\![let\ x = N\ in\ M]\!], \text{ se } x \notin LIV(E).$$

Demonstração. A prova segue por indução em E.

- Caso $E = [\,]$:

  É direto que $(let\ x = N\ in\ M) =_c (let\ x = N\ in\ M)$.

- Caso $E = VE'$:

  Queremos provar que $(let\ x = N\ in\ (VE')[\![M]\!]) =_c (VE')[\![let\ x = N\ in\ M]\!]$.

  - Caso $E'[\![M]\!]$ é um valor:

$$let\ x = N\ in\ (VE')[\![M]\!] \quad = \quad let\ x = N\ in\ V(E'[\![M]\!])$$

$$
\begin{aligned}
&\leftarrow_{\eta_v} && let\ x = N\ in\ (\lambda y.Vy)(E'[\![M]\!]) \\
&\rightarrow_{\beta_v} && let\ x = N\ in\ [E'[\![M]\!]/y](Vy) \\
&\leftarrow_{let_v} && let\ x = N\ in\ (let\ y = E'\ [\![M]\!]\ in\ Vy) \\
&\leftarrow_{comp} && let\ y = (let\ x = N\ in\ E'\ [\![M]\!])\ in\ Vy \\
&\leftarrow_{let_2} && V(let\ x = N\ in\ E'[\![M]\!]) \\
&=_c && V(E'[\![let\ x = N\ in\ M]\!]), \text{ por H.I. pois } x \notin LIV(E'), \\
&= && (VE')[\![let\ x = N\ in\ M]\!].
\end{aligned}
$$

- Caso $E'\ [\![M]\!]$ não é um valor:

$$
\begin{aligned}
let\ x = N\ in\ (VE')[\![M]\!] \quad &= && let\ x = N\ in\ V(E'[\![M]\!]) \\
&\rightarrow_{let_2} && let\ x = N\ in\ (let\ y = E'\ [\![M]\!]\ in\ Vy) \\
&\leftarrow_{comp} && let\ y = (let\ x = N\ in\ E'\ [\![M]\!])\ in\ Vy \\
&\leftarrow_{let_2} && V(let\ x = N\ in\ E'[\![M]\!]) \\
&=_c && V(E'[\![let\ x = N\ in\ M]\!]), \text{ por H.I. pois } x \notin LIV(E'), \\
&= && (VE')[\![let\ x = N\ in\ M]\!].
\end{aligned}
$$

- Caso $E = E'P$:

  Queremos provar que $(let\ x = N\ in\ (E'P)[\![M]\!]) =_c (E'P)[\![let\ x = N\ in\ M]\!]$.

  - Caso $E'\ [\![M]\!]$ é um valor:

$$
\begin{aligned}
let\ x = N\ in\ (E'P)[\![M]\!] \quad &= && let\ x = N\ in\ (E'[\![M]\!])P \\
&\leftarrow_{let_v} && let\ x = N\ in\ (let\ y = E'\ [\![M]\!]\ in\ yP) \\
&\leftarrow_{comp} && let\ y = (let\ x = N\ in\ E'\ [\![M]\!])\ in\ yP \\
&\leftarrow_{let_1} && (let\ x = N\ in\ E'[\![M]\!])P \\
&=_c && (E'[\![let\ x = N\ in\ M]\!])P, \text{ por H.I.} \\
& && \text{pois } x \notin LIV(E'), \\
&= && (E'P)[\![let\ x = N\ in\ M]\!].
\end{aligned}
$$

  - Caso $E'\ [\![M]\!]$ não é um valor:

$$
let\ x = N\ in\ (E'P)[\![M]\!] \quad = \quad let\ x = N\ in\ (E'[\![M]\!])P
$$

$$
\begin{array}{rl}
\rightarrow_{let_1} & let\ x = N\ in\ (let\ y = E'\ [\![M]\!]\ in\ yP) \\
\leftarrow_{comp} & let\ y = (let\ x = N\ in\ E'\ [\![M]\!])\ in\ yP \\
\leftarrow_{let_1} & (let\ x = N\ in\ E'[\![M]\!])P \\
=_c & (E'[\![let\ x = N\ in\ M]\!])P, \text{ por H.I.} \\
 & \text{pois } x \notin LIV(E'), \\
= & (E'P)[\![let\ x = N\ in\ M]\!].
\end{array}
$$

□

Proposição 3. *Sejam $M, N \in \Lambda$. Então,*

$$M =_{v++} N \implies M =_c N.$$

Demonstração. Suponhamos que $M =_{v++} N$. Queremos provar que $M =_c N$. A prova segue por indução na relação $=_{v++}$.

- Casos β_v, η_v e ρ:

  É direto.

- Caso β_{flat}:

  Queremos mostrar que $(zM)L =_c (\lambda x.xL)(zM)$.

$$
\begin{array}{rl}
(zM)L \rightarrow_{let_1} & (let\ y = zM\ in\ yL) \\
\leftarrow_{\beta_v} & (let\ y = zM\ in\ (\lambda x.xL)y) \\
\leftarrow_{let_2} & (\lambda x.xL)(zM).
\end{array}
$$

- Caso β_{lift}:

  Queremos mostrar que $E[\![(\lambda x.M)N]\!] =_c (\lambda x.E[\![M]\!])N$.

  - Caso N é um valor:

$$
\begin{array}{rl}
E[\![(\lambda x.M)N]\!] \rightarrow_{\beta_v} & E[\![[N/x]M]\!] \\
= & [N/x](E[\![M]\!]), \text{ pois } x \notin LIV(E), \\
\leftarrow_{\beta_v} & (\lambda x.E[\![M]\!])N.
\end{array}
$$

- Caso N não é um valor:

$$\begin{aligned} E[\![(\lambda x.M)N]\!] &\to_{let_2} E[\![let\ y = N\ in\ (\lambda x.M)y]\!] \\ &\to_{\beta_v} E[\![let\ y = N\ in\ [y/x]M]\!] \\ &=_c (let\ y = N\ in\ E[\![\,[y/x]M]\!]), \text{ pelo lema 10,} \\ &= (let\ y = N\ in\ [y/x](E[\![M]\!])), \text{ pois } x \notin LIV(E), \\ &\leftarrow_{\beta_v} (let\ y = N\ in\ (\lambda x.E[\![M]\!])y) \\ &\leftarrow_{let_2} (\lambda x.E[\![M]\!])N. \end{aligned}$$

- Caso β_{id}:

  Queremos mostrar que $(\lambda x.x)M =_c M$.

  - Caso M é um valor:

$$(\lambda x.x)M \to_{\beta_v} [M/x]x = M.$$

  - Caso M não é um valor:

$$\begin{aligned} (\lambda x.x)M &\to_{let_2} (let\ y = M\ in\ (\lambda x.x)y) \\ &\to_{\beta_v} (let\ y = M\ in\ y) \\ &\to_{id} M. \end{aligned}$$

- Caso β_ω:

  Queremos mostrar que $(\lambda x.E[\![yx]\!])M =_c E[\![yM]\!]$.

  - Caso M é um valor:

$$\begin{aligned} (\lambda x.E[\![yx]\!])M &\to_{\beta_v} [M/x](E[\![yx]\!]) \\ &= E[\![yM]\!], \text{ pois } x \notin LIV(E[\![y]\!]). \end{aligned}$$

  - Caso M não é um valor:

$$\begin{aligned} (\lambda x.E[\![yx]\!])M &\to_{let_2} (let\ z = M\ in\ (\lambda x.E[\![yx]\!])z) \\ &\to_{\beta_v} (let\ y = M\ in\ [z/x](E[\![yx]\!])) \\ &= (let\ y = M\ in\ E[\![yz]\!]), \text{ pois } x \notin LIV(E[\![y]\!]), \end{aligned}$$

$$
\begin{aligned}
&=_c && E[\![\, let\ y = M\ in\ yz \,]\!], \text{ pelo lema 10,} \\
&\leftarrow_{let_2} && E[\![yM]\!].
\end{aligned}
$$

- Casos ν, μ, ξ, τ e σ:

  Trivial, utilizando as hipóteses de indução.

□

Teorema 20 (**Correspondência Equacional**). *Seja $I(\cdot) : \Lambda \to \Lambda_c$ o mapeamento que corresponde à inclusão dos termos de $\lambda_{v}{+}{+}$ nos termos de λ_c. As traduções h e I formam uma correspondência equacional entre λ_c e $\lambda_{v}{+}{+}$, isto é,*

- $M =_c N \implies h(M) =_{v}{+}{+} h(N)$;
- $M =_{v}{+}{+} N \implies I(M) =_c I(N)$;
- $M =_c I(h(M))$;
- $h(I(M)) =_{v}{+}{+} M$.

Demonstração. A prova segue segundo a definição de correspondência equacional.

Os dois primeiros pontos são diretos pelas proposições 2 e 3, respetivamente.

Vejamos que $M =_c I(h(M))$. A prova segue por indução em $M \in \Lambda_c$.

- Caso $M = x$:

$$x = I(x) = I(h(x)).$$

- Caso $M = \lambda x.N$:

$$
\begin{aligned}
\lambda x.N &=_c \lambda x.I(h(N)), \text{ por H.I. em } N, \\
&= \lambda x.h(N) \\
&= I(\lambda x.h(N)) \\
&= I(h(\lambda x.N)).
\end{aligned}
$$

- Caso $M = PQ$:

$$PQ =_c I(h(P))I(h(Q)), \text{ por H.I. em } P \text{ e } Q,$$

$$\begin{aligned} &= h(P)h(Q) \\ &= I(h(P)h(Q)) \\ &= I(h(PQ)). \end{aligned}$$

- Caso $M = (let\, x = P\, in\, Q)$:

$$\begin{aligned} (let\, x = P\, in\, Q) \quad &= \quad (let\, y = P\, in\, [y/x]Q) \\ &\leftarrow_{\beta_v} \quad (let\, y = P\, in\, (\lambda x.Q)y) \\ &\leftarrow_{let_2} \quad (\lambda x.Q)P \\ &=_c \quad (\lambda x.I(h(Q)))I(h(P)), \text{ por H.I. em } P \text{ e } Q, \\ &= \quad I((\lambda x.h(Q))h(P)) \\ &= \quad I(h(let\, x = P\, in\, Q)). \end{aligned}$$

Por fim, vejamos que $h(I(M)) =_{v++} M$. A prova segue por indução em $M \in \Lambda_{v++}$.

- Caso $M = x$:

$$h(I(x)) = h(x) = x.$$

- Caso $M = \lambda x.N$:

$$\begin{aligned} h(I(\lambda x.N)) &= h(\lambda x.N) \\ &= \lambda x.h(N) \\ &= \lambda x.h(I(N)) \\ &= \lambda x.N, \text{ por H.I. em } N. \end{aligned}$$

- Caso $M = PQ$:

$$\begin{aligned} h(I(PQ)) &= h(PQ) \\ &= h(P)h(Q) \\ &= h(I(P))h(I(Q)) \\ &= PQ, \text{ por H.I. em } P \text{ e } Q. \end{aligned}$$

□

Neste capítulo analisámos o cálculo computacional, uma outra resposta ao problema da incompletude do cálculo de Plotkin para a semântica CPS, um sistema com origem na teoria das linguagens de programação. Mais, provámos que este sistema e a extensão de Sabry-Felleisen formam uma correspondência equacional.

Capítulo 5

Cálculo λ_Q

5.1 Motivação

Consideremos o cálculo de sequentes LJ, estudado na secção 2.4.2.

LJQ, um fragmento de LJ, distingue sequentes ordinários, $\Gamma \Rightarrow A$, de sequentes focados, $\Gamma \rightarrow A$. Estes diferem um do outro na medida em que, em derivações sem corte, um sequente focado representa um sequente cuja última derivação, que lhe deu origem, é feita através de um axioma ou de uma regra à direita. Em LJQ, a regra $L\supset$ utiliza um sequente focado na premissa esquerda restringindo, desta forma, a sua aplicação.

Definição 67 (**Derivações de** LJQ)**.** *O conjunto de derivações de LJQ, apresentado em [7], é o menor conjunto de árvores de sequentes fechado para as seguintes regras:*

$$\frac{}{\Gamma, A \rightarrow A}\ Ax \qquad \frac{\Gamma \rightarrow A}{\Gamma \Rightarrow A}\ Der$$

$$\frac{\Gamma, A \supset B \rightarrow A \quad \Gamma, A \supset B, B \Rightarrow C}{\Gamma, A \supset B \Rightarrow C}\ L\supset \qquad \frac{\Gamma, A \Rightarrow B}{\Gamma \rightarrow A \supset B}\ R\supset$$

$$\frac{\Gamma \rightarrow A \quad \Gamma, A \rightarrow B}{\Gamma \rightarrow B}\ Cut_1 \qquad \frac{\Gamma \rightarrow A \quad \Gamma, A \Rightarrow B}{\Gamma \Rightarrow B}\ Cut_2 \qquad \frac{\Gamma \Rightarrow A \quad \Gamma, A \Rightarrow B}{\Gamma \Rightarrow B}\ Cut_3$$

Consideremos as derivações do sequente $\Rightarrow (A \supset B \supset C) \supset (A \supset B) \supset A \supset C$ apresentadas no capítulo 2.4.2, em LJ. Respetivamente, vamos tentar representar ambas as derivações em LJQ. Seja $\Gamma = A \supset B \supset C, A \supset B, A$. Vejamos:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \to A}\,Ax \quad \dfrac{\dfrac{}{\Gamma, B \supset C \to A}\,Ax \quad \dfrac{\dfrac{}{\Gamma, B \supset C, B \to B}\,Ax \quad \dfrac{\dfrac{}{\Gamma, B \supset C, C \to C}\,Ax}{\Gamma, B \supset C, C \Rightarrow C}\,Der}{\Gamma, B \supset C, B \Rightarrow C}\,L\supset}{\Gamma, B \supset C \Rightarrow C}\,L\supset}{A \supset B \supset C, A \supset B, A \Rightarrow C}\,R\supset}{A \supset B \supset C, A \supset B \to A \supset C}\,Der}{A \supset B \supset C, A \supset B \Rightarrow A \supset C}\,R\supset}{A \supset B \supset C \to (A \supset B) \supset A \supset C}\,Der}{A \supset B \supset C \Rightarrow (A \supset B) \supset A \supset C}\,R\supset}{\to (A \supset B \supset C) \supset (A \supset B) \supset A \supset C}\,Der}{\Rightarrow (A \supset B \supset C) \supset (A \supset B) \supset A \supset C}\,Der$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \to A}\,Ax \quad \dfrac{\not{\mathcal{D}} \qquad \mathcal{D}'}{\Gamma, B \supset C \to B \quad \Gamma, B \supset C, C \Rightarrow C}\,L\supset}{\Gamma, B \supset C \Rightarrow C}\,L\supset}{A \supset B \supset C, A \supset B, A \Rightarrow C}\,R\supset}{A \supset B \supset C, A \supset B \to A \supset C}\,Der}{A \supset B \supset C, A \supset B \Rightarrow A \supset C}\,R\supset}{A \supset B \supset C \to (A \supset B) \supset A \supset C}\,Der}{A \supset B \supset C \Rightarrow (A \supset B) \supset A \supset C}\,R\supset}{\to (A \supset B \supset C) \supset (A \supset B) \supset A \supset C}\,Der}{\Rightarrow (A \supset B \supset C) \supset (A \supset B) \supset A \supset C}\,Der$$

Com efeito, em LJQ, conseguimos apenas representar a primeira derivação de LJ. A representação da segunda derivação falha precisamente na premissa esquerda da regra $L\supset$, visto que o sequente $\Gamma, B \supset C \to B$ não é conclusão de um axioma nem de uma regra à direita. De facto, comparando com a respetiva derivação em LJ, observamos que o sequente deriva de uma regra à esquerda.

O cálculo-λ_Q de Dychoff-Lengrand, que vamos estudar de seguida, resulta de aplicar o isomorfismo de Curry-Howard ao cálculo de sequentes focado LJQ.

Dyckhoff e Lengrand apresentam ainda LJQ como um sistema de tipificação para um certo cálculo-λ, o cálculo-λ_Q.

5.2 O sistema

5.2.1 Versão original

Definição 68 (**termos-λ_Q**). *Seja $\mathcal{E} = \mathcal{A} \cup \{\uparrow, C_1, C_2, C_3\}$ e $\mathcal{E}^*$ o conjunto de todas as palavras em $\mathcal{E}$. O conjunto dos termos-λ_Q, representado por Λ_Q, é definido indutivamente sobre $\mathcal{E}^*$ por:*

$$M, N ::= \uparrow V \mid x(V, y.N) \mid C_2(V, x.N) \mid C_3(M, x.N)$$
$$V, W ::= x \mid \lambda x.M \mid C_1(V, x.W)$$

A um termo-λ_Q da forma $C_i(-, -.-)$, $i = 1, 2, 3$, chama-se substituição explícita.

C_1 representa a substituição de um valor num valor, C_2 a substituição de um valor num termo e $C3$ a substituição de um termo num termo. Por exemplo, $C_3(M, x.N)$ representa a substituição de M por x em N.

Definição 69 (**LJQ com termos-λ_Q**). *O conjunto de tipificações de LJQ, apresentado em [6], é definido indutivamente por:*

$$\frac{}{\Gamma, x : A \to x : A}\ Ax \qquad \frac{\Gamma \to V : A}{\Gamma \Rightarrow \uparrow V : A}\ Der$$

$$\frac{\Gamma, x : A \supset B \to V : A \quad \Gamma, x : A \supset B, y : B \Rightarrow N : C}{\Gamma, x : A \supset B \Rightarrow x(V, y.N) : C}\ L\supset \qquad \frac{\Gamma, x : A \Rightarrow M : B}{\Gamma \to \lambda x.M : A \supset B}\ R\supset$$

$$\frac{\Gamma \to V : A \quad \Gamma, x : A \to W : B}{\Gamma \to C_1(V, x.W) : B}\ Cut_1 \qquad \frac{\Gamma \to V : A \quad \Gamma, x : A \Rightarrow N : B}{\Gamma \Rightarrow C_2(V, x.N) : B}\ Cut_2$$

$$\frac{\Gamma \Rightarrow M : A \quad \Gamma, x : A \Rightarrow N : B}{\Gamma \Rightarrow C_3(M, x.N) : B}\ Cut_3$$

Analisando estas tipificações, observamos que a classe dos valores e dos termos constituem os sequentes focados e ordinários, respetivamente, e que as substituições explícitas representam regras do corte.

Definição 70 (**x-covalor**). *Um termo-λ_Q N diz-se um x-covalor se e só se $N = \uparrow x$ ou N é da forma $x(V, y.P)$, onde $x \notin LIV(VP)$.*

Definição 71 (**Regras de redução para λ_Q**). *As relações binárias de λ_Q são apresentadas sobre a forma das seguintes regras:*

$$C_3(\uparrow(\lambda x.M), y.y(V, z.N)) \to C_3(C_3(\uparrow V, x.M), z.N), \text{ se } y \notin LIV(VN)$$

$$
\begin{aligned}
C_3(\uparrow x, y.N) &\to [x/y]N \\
C_3(M, y.\uparrow y) &\to M \\
C_3(z(V, y.P), x.N) &\to z(V, y.C_3(P, x.N)) \\
C_3(C_3(\uparrow W, y.y(V, z.P)), x.N) &\to C_3(W, y.y(V, z.C_3(P, x.N))), \text{ se } y \notin LIV(VP) \\
C_3(C_3(M, y.P), x.N) &\to C_3(M, y.C_3(P, x.N)), \text{ se o redex não for o} \\
&\qquad \text{redex da regra anterior} \\
C_3(\uparrow(\lambda x.M), y.N) &\to C_2(\lambda x.M, y.N), \text{ se } N \text{ não for } y\text{-covalor} \\
C_1(V, x.x) &\to V \\
C_1(V, x.y) &\to y \\
C_1(V, x.(\lambda y.M)) &\to \lambda y.C_2(V, x.M) \\
C_2(V, x.\uparrow W) &\to \uparrow(C_1(V, \lambda x.W)) \\
C_2(V, x.x(W, z.N)) &\to C_3(\uparrow V, x.x(C_1(V, x.W), z.C_2(V, x.N))) \\
C_2(V, x.y(W, z.N)) &\to y(C_1(V, x.W), z.C_2(V, x.N)) \\
C_2(V, x.C_3(M, y.N)) &\to C_3(C_2(V, x.M), y.C_2(V, x.N))
\end{aligned}
$$

É de notar que as regras de redução deste sistema representam a eliminação do corte no cálculo de sequente.

5.2.2 Versão simplificada

Para a comparação com λ_c não precisamos das substituições explícitas, por isso vamos definir uma nova sintaxe sem as substituições explícitas C_1 e C_2 e vamos adicionar as operações de substituição correspondentes. Consequentemente, o número de regras de redução do sistema vai diminuir. Vamos representar este sistema simplificado por λ_{Q^*}.

Definição 72 (**termos-λ_{Q^*}**). *Seja $\mathcal{F} = \mathcal{A} \cup \{\uparrow, C\}$ e $\mathcal{F}^*$ o conjunto de todas as palavras em $\mathcal{F}$. O conjunto dos termos-λ_{Q^*}, representado por Λ_{Q^*}, é definido indutivamente sobre $\mathcal{F}^*$ por:*

$$
\begin{aligned}
M, N &::= \uparrow V \mid x(V, y.N) \mid C(M, x.N) \\
V, W &::= x \mid \lambda x.M
\end{aligned}
$$

As ocorrências livres de y em N estão ligadas em $x(V, y.N)$ e, analogamente, as ocorrência livres de x em N estão ligadas em $C(M, x.N)$.

Tal como nos sistemas anteriores, convencionamos que os nomes das variáveis livres e ligadas de um termo são diferentes. Assumiremos também que dois termos iguais-α, isto é, que diferem apenas no nome das variáveis ligadas, são iguais.

Definição 73 (Substituição). *A substituição das ocorrência livres de x por V em W ou M é definido recursivamente por:*

$$
\begin{aligned}
[V/x]x &= V \\
[V/x]y &= y \\
[V/x](\lambda y.M) &= \lambda y.[V/x]M \\
[V/x](\uparrow W) &= \uparrow([V/x]W) \\
[V/x](x(W, z.N)) &= C(\uparrow V, x.x([V/x]W, z.[V/x]N)) \\
[V/x](y(W, z.N)) &= y([V/x]W, z.[V/x]N), \text{ se } x \neq y \\
[V/x](C(M, y.N)) &= C([V/x]M, y.[V/x]N)
\end{aligned}
$$

Numa primeira parte, consideremos as seguintes relações binárias:

(B_v) $C(\uparrow(\lambda x.M), y.y(V, z.N)) \to C(C(\uparrow V, x.M), z.N)$, se $y \notin LIV(VN)$

(σ_{v1}) $C(\uparrow x, y.N) \to [x/y]N$

(id) $C(M, y.\uparrow y) \to M$

(π_1) $C(z(V, y.P), x.N) \to z(V, y.C(P, x.N))$

$(*)$ $C(C(\uparrow W, y.y(V, z.P)), x.N) \to C(\uparrow W, y.y(V, z.C(P, x.N)))$, se $y \notin LIV(VP)$

$(comp)$ $C(C(M, y.P), x.N) \to C(M, y.C(P, x.N))$, se o redex não for o redex da regra anterior

(σ_{v2}) $C(\uparrow(\lambda x.M), y.N) \to [(\lambda x.M)/y]N$, se N não for y-covalor

Seja (π_2) a regra $(comp)$ sem a condição lateral. Então, (π_1) e (π_2) derivam $(*)$. Com efeito, $C(C(\uparrow W, y.y(V, z.P)), x.N) \to_{\pi_2} C(\uparrow W, y.C(y(V, z.P), x.N)) \to_{\pi_1} C(W, y.y(V, z.C(P, x.N)))$. Ou seja, o sistema com (π_1), (π_2) e $(*)$ contém uma ambiguidade inócua, que não existia anteriormente.

Seja agora (σ_v) a regra: $C(\uparrow V, y.N) \to [V/y]N$, se V não é uma abstração ou N não é y-covalor. Então, as regras (σ_{v1}) e (σ_{v2}) são casos particulares da regra (σ_v), quando V é uma variável ou uma abstração, respetivamente. Ora, $C(\uparrow x, y.\uparrow y) \to_{\sigma_v} [x/y](\uparrow y) = \uparrow x$ e $C(\uparrow x, y.\uparrow y) \to_{id} \uparrow x$, pelo

que o sistema com (σ_v) e (id) tem uma ambiguidade inócua. No entanto, esta ambiguidade já existia no sistema original, entre (σ_{v1}) e (id).

O sistema simplificado obtém-se adotando (σ_v) e (π_2) e deixando cair $(*)$.

Definição 74 (**Regras de redução para λ_{Q^*}**). *As relações binárias de λ_{Q^*} são apresentadas sobre a forma das seguintes regras:*

$$(B_v) \quad C(\uparrow(\lambda x.M), y.y(V, z.N)) \to C(C(\uparrow V, x.M), z.N), \text{ se } y \notin LIV(VN)$$

$$(id) \quad C(M, y.\uparrow y) \to M$$

$$(\pi_1) \quad C(z(V, y.P), x.N) \to z(V, y.C(P, x.N))$$

$$(\pi_2) \quad C(C(M, y.P), x.N) \to C(M, y.C(P, x.N))$$

$$(\sigma_v) \quad C(\uparrow V, y.N) \to [V/y]N, \text{ se } V \text{ não é abstração ou } N \text{ não é } y\text{-covalor}$$

5.3 Relação com o cálculo computacional

Informalmente, o núcleo de um cálculo resulta da normalização de certas regras de redução que constituem esse sistema. Por exemplo, o núcleo do cálculo-λ_c, que pode ser encontrado em [16], resulta da normalização das regras (let_1), (let_2) e $(comp)$ e é representado por $\lambda_{c^{**}}$. Além disso, o cálculo-$\lambda_{c^{**}}$ e o cálculo-λ_{cps}, apresentado na secção 4.2, formam um isomorfismo.

Para estudar a relação entre os cálculos λ_{Q^*} e λ_c, vamos estudar a relação entre os seus núcleos, sendo que o núcleo do cálculo-λ_{Q^*}, representado por $\lambda_{Q^{**}}$, é definido por nós.

Definição 75 (**Núcleo de λ_c**). *Seja $\mathcal{G} = \mathcal{A} \cup \{let,\ in,\ =,\ [\,]\}$ e $\mathcal{G}^*$ o conjunto de todas as palavras em $\mathcal{G}$. O conjunto de termos-$\lambda_{c^{**}}$ é definido indutivamente sobre $\mathcal{G}^*$ por:*

$$M, N \in \Lambda_{c^{**}} \subseteq \Lambda_c ::= K[V] \mid K[VW]$$

$$V, W ::= x \mid \lambda x.M$$

$$K \subseteq C ::= [\,] \mid let\ x = [\,]\ in\ M$$

Podemos também simplificar a sintaxe da seguinte forma:

$$M, N \in \Lambda_{c^{**}} ::= V \mid VW \mid let\ x = V\ in\ M \mid let\ x = VW\ in\ M$$

$$V, W ::= x \mid \lambda x.M$$

As relações deste sistema são apresentadas sobre a forma das seguintes regras:

$$(\beta_v) \qquad K[(\lambda x.M)V] \to [V/x]M : K, \text{ se } K \text{ é maximal}$$

$$(\eta_v) \qquad \lambda x.Vx \to V, \text{ se } x \notin LIV(V)$$

$$(\beta_{let}) \qquad let\ x = V\ in\ M \to [V/x]M$$

$$(\eta_{let}) \qquad M : let\ x = [\,]\ in\ K[x] \to M : K, \text{ se } x \notin LIV(K)$$

onde $M : K \in \Lambda_{c^{**}}$ pode ser visto como $K[M]$ e se define por recursão em $M \in \Lambda_{c^{**}}$ do seguinte modo:

$$V : K = K[V]$$

$$VW : K = K[VW]$$

$$(let\ x = V\ in\ M) : K = (let\ x = V\ in\ (M : K))$$

$$(let\ x = VW\ in\ M) : K = (let\ x = VW\ in\ (M : K))$$

Um contexto K é maximal se for o contexto maior. Consideremos o termo $let\ y = (\lambda x.M)V\ in\ P$, que é um redex-$\beta_v$. Este termo pode ser lido da forma $K[(\lambda x.M)V]$, com $K = [\,]$ ou $K = let\ y = [\,]\ in\ P$. Nesse caso, deve-se tomar $K = let\ y = [\,]\ in\ P$, isto é, K maior.

Definição 76 (**Relação compatível em** $\lambda_{c^{**}}$)**.** *Uma relação binária $R \subseteq \Lambda^2_{c^{**}}$ diz-se compatível se:*

$$\frac{(M,N) \in R}{(\lambda x.M, \lambda x.N) \in R}\ \xi \qquad \frac{(M,N) \in R}{(MP,NP) \in R}\ \nu \qquad \frac{(M,N) \in R}{(PM,PN) \in R}\ \mu$$

$$\frac{(M,N) \in R}{(let\ x = V\ in\ M,\ let\ x = V\ in\ N) \in R}\ let_{\mu 1} \qquad \frac{(M,N) \in R}{(let\ x = VW\ in\ M,\ let\ x = VW\ in\ N) \in R}\ let_{\mu 2}$$

$$\frac{(V,W) \in R}{(let\ x = V\ in\ P,\ let\ x = W\ in\ P) \in R}\ let_{\nu 1} \qquad \frac{(V,V') \in R}{(let\ x = VW\ in\ P,\ let\ x = V'W\ in\ P) \in R}\ let_{\nu 2}$$

$$\frac{(V,V') \in R}{(let\ x = WV\ in\ P,\ let\ x = WV'\ in\ P) \in R}\ let_{\nu 3}$$

Para simplificar a notação, utilizaremos $M \to_{c^{**}} N$ para representar $M \to_{\beta_v \beta_{let} \eta_v \eta_{let}} N$. O mesmo acontece para o fecho reflexivo e transitivo e para o fecho de congruência.

Definição 77 (**Núcleo de** λ_{Q^*})**.** *O conjunto de termos-$\lambda_{Q^{**}}$, representado por $\Lambda_{Q^{**}}$, é definido indutivamente por:*

$$M, N \in \Lambda_{Q^{**}} \subseteq \Lambda_{Q^*} ::= \uparrow V \mid x(V, y.N) \mid C(\uparrow V, x.N)$$

$$V, W ::= x \mid \lambda x.M$$

As relações deste sistema são apresentadas sobre a forma das seguintes regras:

(β_v) $\quad C(\uparrow(\lambda x.M), y.y(V, z.N)) \to C([V/x]M : z.N)$, *se* $y \notin LIV(VN)$

(id_v) $\quad C(\uparrow V, y.\uparrow y) \to \uparrow V$

(σ_v) $\quad C(\uparrow V, y.N) \to [V/y]N$, *se* V *não é abstração ou* N *não é* y*-covalor*

(η_v) $\quad \lambda x.y(x, z.\uparrow z) \to y$

onde $C(M : x.N) \in \Lambda_{Q^{**}}$ *pode ser visto como* $C(M, x.N)$ *e se define por recursão em* $M \in \Lambda_{Q^{**}}$ *do seguinte modo:*

$$C(\uparrow V : x.N) = C(\uparrow V, x.N)$$

$$C(y(V, z.P) : x.N) = y(V, z.C(P : x.N))$$

$$C(C(\uparrow V, y.P) : x.N) = C(\uparrow V, y.C(P : x.N))$$

A adição da regra (η_v) pode ser justificada através da confluência do sistema.

Definição 78 (**Relação compatível em** $\lambda_{Q^{**}}$). *Uma relação binária* $R \subseteq \Lambda^2_{Q^{**}}$ *diz-se compatível se:*

$$\frac{(M, N) \in R}{(\lambda x.M, \lambda x.N) \in R}\ \xi \qquad \frac{(V, W) \in R}{(\uparrow V, \uparrow W) \in R}\ \uparrow$$

$$\frac{(\uparrow V, \uparrow W) \in R}{(C(\uparrow V, x.P), C(\uparrow W, x.P)) \in R}\ C_\nu \qquad \frac{(M, N) \in R}{((C(\uparrow V, x.M), C(\uparrow V, x.N)) \in R}\ C_\mu$$

$$\frac{(V, W) \in R}{(x(V, y.N), x(W, y.N)) \in R}\ Ap_\nu \qquad \frac{(M, N) \in R}{(x(V, y.M), x(V, y.N)) \in R}\ Ap_\mu$$

Para simplificar a notação, utilizaremos $M \to_{Q^{**}} N$ para representar $M \to_{\beta_v id_v \sigma_v \eta} N$. O mesmo acontece para o fecho reflexivo e transitivo e para o fecho de congruência.

5.3.1 Correspondência equacional entre núcleos

Os mapas que vamos apresentar de seguida são inspirados nas propriedades que Dyckhoff e Lengrand estabelecem entre os sistemas λ_c e λ_Q em [6].

Definição 79 (**Tradução** $\flat$)**.** *Seja* $(\cdot)^\flat : \Lambda_{Q^{**}} \to \Lambda_{c^{**}}$ *o mapeamento definido por:*

$$
\begin{aligned}
x^\flat &= x \\
(\lambda x.M)^\flat &= \lambda x.M^\flat \\
(\uparrow V)^\flat &= V^\flat \\
(x(V, y.M))^\flat &= let\ y = xV^\flat\ in\ M^\flat \\
(C(\uparrow V, y.M))^\flat &= let\ y = V^\flat\ in\ M^\flat
\end{aligned}
$$

Lema 11. *Sejam* V, W *e* $M \in \Lambda_{Q^{**}}$. *Então,*

$$
\begin{aligned}
&(i) \qquad ([V/x]W)^\flat \twoheadrightarrow_{c^{**}} [V^\flat/x]W^\flat; \\
&(ii) \qquad ([V/x]M)^\flat \twoheadrightarrow_{c^{**}} [V^\flat/x]M^\flat.
\end{aligned}
$$

Demonstração. A prova segue por indução simultânea em W e $M \in \Lambda_{Q^{**}}$.

- Caso $W = x$:

$$
\begin{aligned}
([V/x]x)^\flat &= V^\flat \\
&= [V^\flat/x]x \\
&= [V^\flat/x]x^\flat.
\end{aligned}
$$

- Caso $W = y$:

$$
\begin{aligned}
([V/x]y)^\flat &= y^\flat \\
&= y \\
&= [V^\flat/x]y \\
&= [V^\flat/x]y^\flat.
\end{aligned}
$$

- Caso $W = \lambda y.P$:

$$
\begin{aligned}
([V/x](\lambda y.P))^\flat &= (\lambda y.[V/x]P)^\flat \\
&= \lambda y.([V/x]P)^\flat \\
&\twoheadrightarrow_{c^{**}} \lambda y.[V^\flat/x]P^\flat, \text{ por H.I. em } P, \\
&= [V^\flat/x](\lambda y.P^\flat) \\
&= [V^\flat/x](\lambda y.P)^\flat.
\end{aligned}
$$

- Caso $M = \uparrow W$:

$$\begin{array}{rcl}
([V/x](\uparrow W))^\flat & = & (\uparrow([V/x]W))^\flat \\
& = & ([V/x]W)^\flat \\
& \twoheadrightarrow_{c^{**}} & [V^\flat/x]W^\flat, \text{ por H.I. em } W, \\
& = & [V^\flat/x](\uparrow W)^\flat.
\end{array}$$

- Caso $M = y(W, z.P)$:

$$\begin{array}{rcl}
([V/x](y(W, z.P)))^\flat & = & (y([V/x]W, z.[V/x]P))^\flat \\
& = & let\ z = y([V/x]W)^\flat\ in\ ([V/x]P)^\flat \\
& \twoheadrightarrow_{c^{**}} & let\ z = y[V^\flat/x]W^\flat\ in\ [V^\flat/x]P^\flat, \text{ por H.I. em } W \text{ e } P, \\
& = & [V^\flat/x](let\ z = yW^\flat\ in\ P^\flat) \\
& = & [V^\flat/x](y(W, z.P))^\flat.
\end{array}$$

- Caso $M = x(W, z.P)$:

$$\begin{array}{rcl}
([V/x](x(W, z.P)))^\flat & = & (C(\uparrow V, x.x([V/x]W, z.[V/x]P)))^\flat \\
& = & let\ x = V^\flat\ in\ (x([V/x]W, z.[V/x]P))^\flat \\
& = & let\ x = V^\flat\ in\ (let\ z = x([V/x]W)^\flat\ in\ [V/x]P)^\flat) \\
& \twoheadrightarrow_{c^{**}} & let\ x = V^\flat\ in\ (let\ z = x[V^\flat/x]W^\flat\ in\ [V^\flat/x]P^\flat)\ , \\
& & \text{por H.I. em } W \text{ e } P, \\
& \rightarrow_{\beta_{let}} & let\ z = V^\flat[V^\flat/x]W^\flat\ in\ [V^\flat/x]P^\flat \\
& = & [V^\flat/x](let\ z = xW^\flat\ in\ P^\flat) \\
& = & [V^\flat/x](x(W, z.P))^\flat.
\end{array}$$

- Caso $M = C(\uparrow W, y.P)$:

$$\begin{array}{rcl}
([V/x](C(\uparrow W, y.P)))^\flat & = & (C(\uparrow([V/x]W), y.[V/x]P))^\flat \\
& = & let\ y = ([V/x]W)^\flat\ in\ ([V/x]P)^\flat \\
& \twoheadrightarrow_{c^{**}} & let\ y = [V^\flat/x]W^\flat\ in\ [V^\flat/x]P^\flat, \text{ por H.I. em } W \text{ e } P, \\
& = & [V^\flat/x](let\ y = W^\flat\ in\ P^\flat) \\
& = & [V^\flat/x](C(\uparrow W, y.P))^\flat.
\end{array}$$

□

Lema 12. *Sejam M e $N \in \Lambda_{Q^{**}}$. Então,*

$$(M^\flat : let\ x = [\,]\ in\ N^\flat) = (C(M : x.N))^\flat.$$

Demonstração. A prova segue por indução em $M \in \Lambda_{Q^{**}}$.

- Caso $M = \uparrow V$:

$$\begin{aligned}(\uparrow V)^\flat : let\ x = [\,]\ in\ N^\flat &= V^\flat : let\ x = [\,]\ in\ N^\flat \\ &= let\ x = V^\flat\ in\ N^\flat \\ &= (C(\uparrow V, x.N))^\flat \\ &= (C(\uparrow V : x.N))^\flat.\end{aligned}$$

- Caso $M = y(V, z.P)$:

$$\begin{aligned}(y(V, z.P))^\flat : let\ x = [\,]\ in\ N^\flat &= let\ z = yV^\flat\ in\ P^\flat : let\ x = [\,]\ in\ N^\flat \\ &= let\ z = yV^\flat\ in\ (P^\flat : let\ x = [\,]\ in\ N^\flat) \\ &= let\ z = yV^\flat\ in\ C(P : x.N)^\flat, \text{ por H.I. em } P, \\ &= (y(V, z.C(P : x.N)))^\flat \\ &= (C(y(V, z.P) : x.N))^\flat.\end{aligned}$$

- Caso $M = C(\uparrow V, z.P)$:

$$\begin{aligned}(C(\uparrow V, z.P))^\flat : let\ x = [\,]\ in\ N^\flat &= let\ z = V^\flat\ in\ P^\flat : let\ x = [\,]\ in\ N^\flat \\ &= let\ z = V^\flat\ in\ (P^\flat : let\ x = [\,]\ in\ N^\flat) \\ &= let\ z = V^\flat\ in\ C(P : x.N)^\flat, \text{ por H.I. em } P, \\ &= (C(\uparrow V, z.C(P : x.N)))^\flat \\ &= (C(C(\uparrow V, z.P) : x.N))^\flat.\end{aligned}$$

□

Proposição 4. *Sejam M e $N \in \Lambda_{Q^{**}}$. Então,*

$$M \twoheadrightarrow_{Q^{**}} N \implies M^\flat =_{c^{**}} N^\flat.$$

Demonstração. A prova segue por indução na relação $M \twoheadrightarrow_{Q^{**}} N$.

- Caso β_v:

  Queremos mostrar que $(C(\uparrow(\lambda x.M), y.y(V, z.N)))^\flat \twoheadrightarrow_{c^{**}} (C([V/x]M : z.N))^\flat$, onde $y \notin LIV(VN)$.

$$\begin{aligned}
(C(\uparrow(\lambda x.M), y.y(V, z.N)))^\flat &= let\ y = (\uparrow(\lambda x.M))^\flat\ in\ (y(V, z.N))^\flat \\
&= let\ y = \lambda x.N^\flat\ in\ (let\ z = yV^\flat\ in\ N^\flat) \\
&\to_{\beta_{let}} let\ z = (\lambda x.M^\flat)V^\flat\ in\ N^\flat \\
&\to_{\beta_v} [V^\flat/x](M^\flat) : let\ z = [\,]\ in\ N^\flat \\
&\twoheadleftarrow_{c^{**}} ([V/x]M)^\flat : let\ z = [\,]\ in\ N^\flat, \text{ pelo lema 11,} \\
&= (C([V/x]M : z.N))^\flat, \text{ pelo lema 12.}
\end{aligned}$$

- Caso id_v:

  Queremos mostrar que $(C(\uparrow V, y.\uparrow y))^\flat \twoheadrightarrow_{c^{**}} (\uparrow V)^\flat$.

$$\begin{aligned}
(C(\uparrow V, y.\uparrow y))^\flat &= let\ y = (\uparrow V)^\flat\ in\ (\uparrow y)^\flat \\
&= let\ y = V^\flat\ in\ y \\
&\to_{\eta_{let}} V^\flat \\
&= (\uparrow V)^\flat.
\end{aligned}$$

- Caso σ_v:

  Queremos mostrar que $(C(\uparrow V, y.N))^\flat \twoheadrightarrow_{c^{**}} ([V/y]N)^\flat$, se V não é abstração ou N não é y-covalor.

$$\begin{aligned}
(C(\uparrow V, y.N))^\flat &= (let\ y = V^\flat\ in\ N^\flat) \\
&\to_{\eta_v} [V^\flat/y](N^\flat) \\
&\twoheadleftarrow_{c^{**}} ([V/y]N)^\flat, \text{ pelo lema 11.}
\end{aligned}$$

- Caso η_v:

  Queremos mostrar que $(\lambda x.y(x, z.\uparrow z))^\flat \twoheadrightarrow_{c^{**}} y^\flat$.

$$(\lambda x.y(x, z.\uparrow z))^\flat = \lambda x.(y(x, z.\uparrow z))^\flat$$

$$\begin{aligned} &= && \lambda x.(let\ z = yx^\flat\ in\ z^\flat) \\ &= && \lambda x.(let\ z = yx\ in\ z) \\ &\to_{\eta_{let}} && \lambda x.yx \\ &\to_{\eta_v} && y \\ &= && y^\flat. \end{aligned}$$

- Os outros casos, relativos ao fecho compatível e ao fecho reflexivo e transitivo da relação, seguem por hipótese de indução.

□

Definição 80 (Tradução $\sharp$). *Seja $(\cdot)^\sharp : \Lambda_{c^{**}} \to \Lambda_{Q^{**}}$ o mapeamento definido recursivamente da seguinte forma:*

- *Para cada $M \in \Lambda_{c^{**}}$, $M^\sharp$ um termo de $\Lambda_{Q^{**}}$:*

$$\begin{aligned} V^\sharp &= \uparrow(V^\natural) \\ (yW)^\sharp &= y(W^\natural, z.z^\sharp) \\ (VW)^\sharp &= C(V^\sharp, y.y(W^\natural, z.z^\sharp)),\ \text{se } V \text{ é uma abstração} \\ (let\ x = V\ in\ N)^\sharp &= C(V^\sharp, x.N^\sharp) \\ (let\ x = yW\ in\ N)^\sharp &= y(W^\natural, x.N^\sharp) \\ (let\ x = VW\ in\ N)^\sharp &= C(V^\sharp, y.y(W^\natural, x.N^\sharp)),\ \text{se } V \text{ é uma abstração} \end{aligned}$$

- *Para cada $V \in \Lambda_{c^{**}}$, $V^\natural$ um valor de $\Lambda_{Q^{**}}$:*

$$\begin{aligned} x^\natural &= x \\ (\lambda x.M)^\natural &= \lambda x.M^\sharp \end{aligned}$$

Lema 13. *Sejam V, W e $M \in \Lambda_{c^{**}}$. Então,*

$$\begin{aligned} &(i) && [V^\natural/x]W^\natural \twoheadrightarrow_{Q^{**}} ([V/x]W)^\natural; \\ &(ii) && [V^\natural/x]M^\sharp \twoheadrightarrow_{Q^{**}} ([V/x]M)^\sharp. \end{aligned}$$

Demonstração. A prova segue por indução simultânea em W e $M \in \Lambda_{c^{**}}$.

- Caso $W = x$:

$$\begin{aligned}[V^\natural/x]x^\natural &= [V^\natural/x]x \\ &= V^\natural \\ &= ([V/x]x)^\natural.\end{aligned}$$

- Caso $W = y$:

$$\begin{aligned}[V^\natural/x]y^\natural &= [V^\natural/x]y \\ &= y \\ &= y^\natural \\ &= ([V/x]y)^\natural.\end{aligned}$$

- Caso $W = \lambda y.P$:

$$\begin{aligned}[V^\natural/x](\lambda y.P)^\natural &= [V^\natural/x](\lambda y.P^\sharp) \\ &= \lambda y.[V^\natural/x]P^\sharp \\ &\twoheadrightarrow_{Q^{**}} \lambda y.([V/x]P)^\sharp, \text{ por H.I. em } P, \\ &= (\lambda y.[V/x]P)^\natural \\ &= ([V/x](\lambda y.P))^\natural.\end{aligned}$$

- Caso $M = W$:

$$\begin{aligned}[V^\natural/x]W^\sharp &= [V^\natural/x](\uparrow W^\natural) \\ &= \uparrow([V^\natural/x]W^\natural) \\ &\twoheadrightarrow_{Q^{**}} \uparrow([V/x]W)^\natural, \text{ por H.I. em } W, \\ &= ([V/x]W)^\sharp.\end{aligned}$$

- Caso $M = yW$:

$$\begin{aligned}[V^\natural/x](yW)^\sharp &= [V^\natural/x](y(W^\natural, z.z^\sharp)) \\ &= y([V^\natural/x]W^\natural, z.[V^\natural/x]z^\sharp) \\ &= y([V^\natural/x]W^\natural, z.z^\sharp)\end{aligned}$$

$$\begin{aligned}
&\twoheadrightarrow_{Q^{**}} && y(([V/x]W)^\natural, z.z^\sharp), \text{ por H.I. em } W,\\
&= && (y[V/x]W)^\sharp\\
&= && ([V/x](yW))^\sharp.
\end{aligned}$$

- Caso $M = xW$:

  - Se $V = y$:

$$\begin{aligned}
[y^\natural/x](xW)^\sharp &= [y^\natural/x](x(W^\natural, z.z^\sharp))\\
&= C(\uparrow y^\natural, x.x([y^\natural/x]W^\natural, z.[y^\natural/x]z^\sharp))\\
&= C(\uparrow y, x.x([y^\natural/x]W^\natural, z.[y^\natural/x]z^\sharp))\\
&\to_{\sigma_v} y([y^\natural/x]W^\natural, z.[y/x]z^\sharp)\\
&= y([y^\natural/x]W^\natural, z.z^\sharp)\\
&\twoheadrightarrow_{Q^{**}} y(([y/x]W)^\natural, z.z^\sharp), \text{ por H.I. em } W,\\
&= (y[y/x]W)^\sharp\\
&= ([y/x](xW))^\sharp.
\end{aligned}$$

  - Se V é uma abstração:

$$\begin{aligned}
[V^\natural/x](xW)^\sharp &= [V^\natural/x](x(W^\natural, z.z^\sharp))\\
&= C(\uparrow V^\natural, x.x([V^\natural/x]W^\natural, z.[V^\natural/x]z^\sharp))\\
&= C(V^\sharp, x.x([V^\natural/x]W^\natural, z.z^\sharp))\\
&= C(V^\sharp, y.y([V^\natural/x]W^\natural, z.z^\sharp))\\
&\twoheadrightarrow_{Q^{**}} C(V^\sharp, y.y(([V/x]W)^\natural, z.z^\sharp)), \text{ por H.I. em } W,\\
&= (V[V/x]W)^\sharp\\
&= ([V/x](xW))^\sharp.
\end{aligned}$$

- Caso $M = W_1W_2$, onde W_1 é uma abstração:

$$\begin{aligned}
[V^\natural/x](W_1W_2)^\sharp &= [V^\natural/x](C(W_1^\sharp, y.y(W_2^\natural, z.z^\sharp)))\\
&= C([V^\natural/x]W_1^\sharp, y.y([V^\natural/x]W_2^\natural, z.[V^\natural/x]z^\sharp))\\
&= C([V^\natural/x]W_1^\sharp, y.y([V^\natural/x]W_2^\natural, z.z^\sharp))
\end{aligned}$$

$$
\begin{array}{rcl}
 & \twoheadrightarrow_{Q^{**}} & C(([V/x]W_1)^\sharp, y.y(([V/x]W_2)^\natural, z.z^\sharp)), \text{ por H.I. em } W_1 \text{ e } W_2, \\
 & = & ([V/x]W_1[V/x]W_2)^\sharp \\
 & = & ([V/x](W_1W_2))^\sharp.
\end{array}
$$

- Caso $M = (let\ y = W\ in\ N)$:

$$
\begin{array}{rcl}
[V^\natural/x](let\ y = W\ in\ N)^\sharp & = & [V^\natural/x](C(W^\sharp, y.N^\sharp)) \\
 & = & C([V^\natural/x]W^\sharp, y.[V^\natural/x]N^\sharp) \\
 & \twoheadrightarrow_{Q^{**}} & C(([V/x]W)^\sharp, y.([V/x]N)^\sharp), \text{ por H.I. em } W \text{ e } N, \\
 & = & (let\ y = [V/x]W\ in\ [V/x]N)^\sharp \\
 & = & ([V/x](let\ y = W\ in\ N))^\sharp.
\end{array}
$$

- Caso $M = (let\ z = yW\ in\ N)$:

$$
\begin{array}{rcl}
[V^\natural/x](let\ z = yW\ in\ N)^\sharp & = & [V^\natural/x](y(W^\natural, z.N^\sharp)) \\
 & = & y([V^\natural/x]W^\natural, z.[V^\natural/x]N^\sharp) \\
 & \twoheadrightarrow_{Q^{**}} & y(([V/x]W)^\natural, z.([V/x]N)^\sharp), \text{ por H.I. em } W \text{ e } N, \\
 & = & (let\ z = y[V/x]W\ in\ [V/x]N)^\sharp \\
 & = & ([V/x](let\ z = yW\ in\ N))^\sharp.
\end{array}
$$

- Caso $M = (let\ z = xW\ in\ N)$:

  - Caso $V = y$:

$$
\begin{array}{rcl}
[y^\natural/x](let\ z = xW\ in\ N)^\sharp & = & [y^\natural/x](x(W^\natural, z.N^\sharp)) \\
 & = & C(\uparrow y^\natural, x.x([y^\natural/x]W^\natural, z.[y^\natural/x]N^\sharp)) \\
 & = & C(\uparrow y, x.x([y^\natural/x]W^\natural, z.[y^\natural/x]N^\sharp)) \\
 & \to_{\sigma_v} & y([y^\natural/x]W^\natural, z.[y^\natural/x]N^\sharp) \\
 & \twoheadrightarrow_{Q^{**}} & y(([y/x]W)^\natural, z.([y/x]N)^\sharp), \text{ por H.I. em } W \text{ e } N, \\
 & = & (let\ z = y[y/x]W\ in\ [y/x]N)^\sharp \\
 & = & ([y/x](let\ z = xW\ in\ N))^\sharp.
\end{array}
$$

– Caso V é uma abstração:

$$\begin{aligned}
[V^\natural/x](let\ z = xW\ in\ N)^\sharp &= [V^\natural/x](x(W^\natural, z.N^\sharp)) \\
&= C(\uparrow V^\natural, x.x([V^\natural/x]W^\natural, z.[V^\natural/x]N^\sharp)) \\
&= C(V^\sharp, x.x([V^\natural/x]W^\natural, z.[V^\natural/x]N^\sharp)) \\
&= C(V^\sharp, y.y([V^\natural/x]W^\natural, z.[V^\natural/x]N^\sharp)) \\
&\twoheadrightarrow_{Q^{**}} C(V^\sharp, y.y(([V/x]W)^\natural, z.([V/x]N)^\sharp)), \\
&\quad \text{por H.I. em } W \text{ e } N, \\
&= (let\ z = V[V/x]W\ in\ [V/x]N)^\sharp \\
&= ([V/x](let\ z = xW\ in\ N))^\sharp.
\end{aligned}$$

- Caso $M = (let\ z = W_1W_2\ in\ N)$:

$$\begin{aligned}
[V^\natural/x](let\ z = W_1W_2\ in\ N)^\sharp &= [V^\natural/x](C(W_1^\sharp, y.y(W_2^\natural, z.N^\sharp))) \\
&= C([V^\natural/x]W_1^\sharp, y.y([V^\natural/x]W_2^\natural, z.[V^\natural/x]N^\sharp)) \\
&\twoheadrightarrow_{Q^{**}} C(([V/x]W_1)^\sharp, y.y(([V/x]W_2)^\natural, z.([V/x]N)^\sharp)), \\
&\quad \text{por H.I. em } W_1, W_2 \text{ e } N, \\
&= (let\ z = [V/x]W_1[V/x]W_2\ in\ [V/x]N)^\sharp \\
&= ([V/x](let\ z = W_1W_2\ in\ N))^\sharp.
\end{aligned}$$

□

Lema 14. *Seja $M \in \Lambda_{Q^{**}}$. Então,*

$$C(M : y.\uparrow y) \twoheadrightarrow_{Q^{**}} M.$$

Demonstração. A prova segue por indução em $M \in \Lambda_{Q^{**}}$.

- Caso $M = \uparrow V$:

$$\begin{aligned}
C(\uparrow V : y.\uparrow y) &= C(\uparrow V, y.\uparrow y) \\
&\rightarrow_{id_v} \uparrow V.
\end{aligned}$$

- Caso $M = x(V, z.N)$:

$$\begin{aligned}
C(x(V, z.N) : y.\uparrow y) &= x(V, z.C(N : y.\uparrow y)) \\
&\twoheadrightarrow_{Q^{**}} x(V, z.N), \text{ por H.I. em } N.
\end{aligned}$$

- Caso $M = C(\uparrow V, x.N)$:

$$\begin{array}{rcl} C(C(\uparrow V, x.N) : y.\uparrow y) & = & C(\uparrow V, x.C(N : y.\uparrow y)) \\ & \twoheadrightarrow_{Q^{**}} & C(\uparrow V, x.N), \text{ por H.I. em } N. \end{array}$$

□

Lema 15. *Seja M e $N \in \Lambda_{c^{**}}$. Então,*

$$C(M^\sharp : x.N^\sharp) \twoheadrightarrow_{Q^{**}} (M : let\, x = [\,]\, in\, N)^\sharp.$$

Demonstração. A prova segue por indução em $M \in \Lambda_{c^{**}}$.

- Caso $M = V$:

$$\begin{array}{rcl} C(V^\sharp : x.N^\sharp) & = & C(V^\sharp, x.N^\sharp) \\ & = & (let\, x = V\, in\, N)^\sharp \\ & = & (V : let\, x = [\,]\, in\, N)^\sharp. \end{array}$$

- Caso $M = yW$:

$$\begin{array}{rcl} C((yW)^\sharp : x.N^\sharp) & = & C(y(W^\natural, z.z^\sharp) : x.N^\sharp) \\ & = & y(W^\natural, z.C(z^\sharp : x.N^\sharp)) \\ & = & y(W^\natural, z.C(z^\sharp, x.N^\sharp)) \\ & \to_{\sigma_v} & y(W^\natural, z.[z/x]N^\sharp) \\ & = & y(W^\natural, x.N^\sharp) \\ & = & (let\, x = yW\, in\, N)^\sharp \\ & = & (yW : let\, x = [\,]\, in\, N)^\sharp. \end{array}$$

- Caso $M = VW$, onde V é uma abstração:

$$\begin{array}{rcl} C((VW)^\sharp : x.N^\sharp) & = & C(C(V^\sharp, y.y(W^\natural, z.z^\sharp)) : x.N^\sharp) \\ & = & C(V^\sharp, y.C(y(W^\natural, z.z^\sharp) : x.N^\sharp)) \\ & = & C(V^\sharp, y.y(W^\natural, z.C(z^\sharp : x.N^\sharp))) \\ & = & C(V^\sharp, y.y(W^\natural, z.C(z^\sharp, x.N^\sharp))) \end{array}$$

$$
\begin{aligned}
&\to_{\sigma_v} && C(V^\sharp, y.y(W^\natural, z.[z/x]N^\sharp)) \\
&= && C(V^\sharp, y.y(W^\natural, x.N^\sharp)) \\
&= && (let\ x = VW\ in\ N)^\sharp \\
&= && (VW : let\ x = [\,]\ in\ N)^\sharp.
\end{aligned}
$$

- Caso $M = (let\ y = V\ in\ P)$:

$$
\begin{aligned}
C((let\ y = V\ in\ P)^\sharp : x.N^\sharp) \quad &= && C(C(V^\sharp, y.P^\sharp) : x.N^\sharp) \\
&= && C(V^\sharp, y.C(P^\sharp : x.N^\sharp)) \\
&\twoheadrightarrow_{Q^{**}} && C(V^\sharp, y.(P : let\ x = [\,]\ N)^\sharp), \text{ por H.I. em } P, \\
&= && (let\ y = V\ in\ (P : let\ x = [\,]\ N))^\sharp \\
&= && (let\ y = V\ in\ P : let\ x = [\,]\ in\ N)^\sharp.
\end{aligned}
$$

- Caso $M = (let\ z = yW\ in\ P)$:

$$
\begin{aligned}
C((let\ z = yW\ in\ P)^\sharp : x.N^\sharp) \quad &= && C(y(W^\natural, z.P^\sharp) : x.N^\sharp) \\
&= && y(W^\natural, z.C(P^\sharp : x.N^\sharp)) \\
&\twoheadrightarrow_{Q^{**}} && y(W^\natural, z.(P : let\ x = [\,]\ N)^\sharp), \text{ por H.I. em } P, \\
&= && (let\ z = yW\ in\ (P : let\ x = [\,]\ N))^\sharp \\
&= && (let\ z = yW\ in\ P : let\ x = [\,]\ in\ N)^\sharp.
\end{aligned}
$$

- Caso $M = (let\ z = VW\ in\ P)$, onde V é uma abstração:

$$
\begin{aligned}
C((let\ z = VW\ in\ P)^\sharp : x.N^\sharp) \quad &= && C(C(V^\sharp, y.y(W^\natural, z.P^\sharp)) : x.N^\sharp) \\
&= && C(V^\sharp, y.C(y(W^\natural, z.P^\sharp) : x.N^\sharp)) \\
&= && C(V^\sharp, y.y(W^\natural, z.C(P^\sharp : x.N^\sharp))) \\
&= && C(V^\sharp, y.y(W^\natural, z.C(P^\sharp : x.N^\sharp))) \\
&\twoheadrightarrow_{Q^{**}} && C(V^\sharp, y.y(W^\natural, z.(P : let\ x = [\,]\ N)^\sharp)), \\
& && \text{por H.I. em } P, \\
&= && (let\ z = VW\ in\ (P : let\ x = [\,]\ N))^\sharp \\
&= && (let\ z = VW\ in\ P : let\ x = [\,]\ in\ N)^\sharp.
\end{aligned}
$$

□

Proposição 5. *Sejam M e $N \in \Lambda_{c^{**}}$. Então,*

$$M \twoheadrightarrow_{c^{**}} N \implies M^\sharp \twoheadrightarrow_{Q^{**}} N^\sharp.$$

Demonstração. A prova segue por indução na relação $M =_{c^{**}} N$.

- Caso β_v:
  - Caso $K = [\,]$:

    Queremos mostrar que $((\lambda x.M)V)^\sharp \twoheadrightarrow_{Q^{**}} ([V/x]M)^\sharp$.

$$\begin{array}{rcl} ((\lambda x.M)V)^\sharp & = & C((\lambda x.M)^\sharp, y.y(V^\natural, z.z^\sharp)) \\ & \to_{\beta_v} & C([V^\natural/x]M^\sharp : z.z^\sharp) \\ & \twoheadrightarrow_{Q^{**}} & C(([V/x]M)^\sharp : z.z^\sharp), \text{ pelo lema 13,} \\ & = & C(([V/x]M)^\sharp : z. \uparrow z) \\ & \twoheadrightarrow_{Q^{**}} & ([V/x]M)^\sharp, \text{ pela lema 14.} \end{array}$$

  - Caso $K = (let\ y = [\,]\ in\ P)$:

    Queremos mostrar que $(let\ y = (\lambda x.M)V\ in P)^\sharp \twoheadrightarrow_{Q^{**}} ([V/x]M : let\ y = [\,]\ in\ P)^\sharp$.

$$\begin{array}{rcl} (let\ y = (\lambda x.M)V\ in\ P)^\sharp & = & C((\lambda x.M)^\sharp, z.z(V^\natural, y.P^\sharp)) \\ & = & C(\uparrow(\lambda x.M^\sharp), z.z(V^\natural, y.P^\sharp)) \\ & \to_{\beta_v} & C([V^\natural/x]M^\sharp : y.P^\sharp) \\ & \twoheadrightarrow_{Q^{**}} & C(([V/x]M)^\sharp : y.P^\sharp), \text{ pelo lema 13,} \\ & \twoheadrightarrow_{Q^{**}} & ([V/x]M : let\ y = [\,]\ in\ P)^\sharp, \text{ pelo lema 15.} \end{array}$$

- Caso η_v:

  Queremos mostrar que $(\lambda x.Vx)^\sharp \twoheadrightarrow_{Q^{**}} y^\sharp$, onde $x \notin LIV(V)$.

  - Caso $V = y$:

$$\begin{array}{rcl} (\lambda x.yx)^\sharp & = & \uparrow(\lambda x.(yx)^\sharp) \\ & = & \uparrow(\lambda x.y(x^\sharp, z.z^\sharp)) \end{array}$$

$$\begin{array}{rl} = & \uparrow(\lambda x.y(x, z.\uparrow z)) \\ \rightarrow_{\eta_v} & \uparrow y \\ = & y^\sharp. \end{array}$$

- Caso $V = \lambda y.P$:

$$\begin{array}{rl} (\lambda x.(\lambda y.P)x)^\sharp = & \uparrow(\lambda x.((\lambda y.P)x)^\sharp) \\ = & \uparrow(\lambda x.C((\lambda y.P)^\sharp, z.z(x^\natural, u.u^\sharp))) \\ = & \uparrow(\lambda x.C(\uparrow(\lambda y.P^\sharp), z.z(x, u.u^\sharp))) \\ \rightarrow_{\beta_v} & \uparrow(\lambda x.C([x/y]P^\sharp : u.u^\sharp)) \\ = & \uparrow(\lambda x.C([x/y]P^\sharp : u.\uparrow u)) \\ \twoheadrightarrow_{Q^{**}} & \uparrow(\lambda x.[x/y]P^\sharp)), \text{ pelo lema 14,} \\ = & \uparrow(\lambda y.P^\sharp) \\ = & (\lambda y.P)^\sharp. \end{array}$$

- Caso β_{let}:

  Queremos mostrar que $(let\ x = V\ in\ M)^\sharp \twoheadrightarrow_{Q^{**}} ([V/x]M)^\sharp$.

  Ora, $(let\ x = V\ in\ M)^\sharp = C(V^\sharp, x.M^\sharp)$. Falta ver que $C(V^\sharp, x.M^\sharp) \twoheadrightarrow_{Q^{**}} ([V/x]M)^\sharp$.

  Para aplicar a regra (σ_v), $V^\sharp$ não pode ser abstração ou $M^\sharp$ não pode ser x-covalor, isto é, não pode ser da forma $\uparrow x$ ou $x(W, y.N)$ com $x \notin LIV(WN)$. Desta forma, é necessário estudar os casos que se seguem.

  - Caso V é uma abstração e $M = let\ z = xW\ in\ N$, com $x \notin LIV(WN)$:

$$\begin{array}{rl} C(V^\sharp, x.(let\ z = xW\ in\ N)^\sharp) = & C(V^\sharp, x.x(W^\natural, z.N^\sharp)) \\ = & (let\ z = VW\ in\ N)^\sharp \\ = & ([V/x](let\ z = xW\ in\ N)), \text{ pois } x \notin LIV(WN). \end{array}$$

  - Caso V é uma abstração e $M = xW$, com $x \notin LIV(W)$:

$$\begin{array}{rl} C(V^\sharp, x.(xW)^\sharp) = & C(V^\sharp, x.x(W^\natural, z.z^\sharp)) \\ = & (let\ z = VW\ in\ z)^\sharp \\ = & ([V/x](let\ z = xW\ in\ z)), \text{ pois } x \notin LIV(W). \end{array}$$

- Caso V é uma abstração e $M = x$:

$$\begin{array}{rll} C(V^\sharp, x.x^\sharp) & = & C(\uparrow(V^\natural), x.\uparrow x) \\ & \rightarrow_{id_v} & \uparrow(V^\natural) \\ & = & V^\sharp. \end{array}$$

- Caso contrário:

$$\begin{array}{rll} C(V^\sharp, x.M^\sharp) & = & C(\uparrow(V^\natural), x.M^\sharp) \\ & \rightarrow_{\sigma_v} & [V^\natural/x]M^\sharp \\ & \twoheadrightarrow_{Q^{**}} & ([V/x]M)^\sharp, \text{ pelo lema 13.} \end{array}$$

- Caso η_{let}:

  - Caso $K = [\,]$:

    Queremos mostrar que $(M : let\, x = [\,]\, in\, x)^\sharp \twoheadrightarrow_{Q^{**}} (M : [\,])^\sharp = M^\sharp$.

    * Caso $M = V$:

$$\begin{array}{rll} (V : let\, x = [\,]\, in\, x)^\sharp & = & (let\, x = V\, in\, x)^\sharp \\ & = & C(V^\sharp, x.x^\sharp) \\ & = & C(\uparrow(V^\natural), x.\uparrow x) \\ & \rightarrow_{id_v} & \uparrow(V^\natural) \\ & = & V^\sharp. \end{array}$$

    * Caso $M = yW$:

$$\begin{array}{rll} (yW : let\, x = [\,]\, in\, x)^\sharp & = & (let\, x = yW\, in\, x)^\sharp \\ & = & y(W^\natural, x.x^\sharp) \\ & = & (yW)^\sharp. \end{array}$$

    * Caso $M = VW$, onde V é uma abstração:

$$\begin{array}{rll} (VW : let\, x = [\,]\, in\, x)^\sharp & = & (let\, x = VW\, in\, x)^\sharp \\ & = & C(V^\sharp, y.y(W^\natural, x.x^\sharp)) \\ & = & (VW)^\sharp. \end{array}$$

* Caso $M = (let\ y = V\ in\ N)$:

$$\begin{array}{rcl}((let\ y = V\ in\ N) : let\ x = [\,]\ in\ x)^\sharp & = & (let\ y = V\ in\ (N : let\ x = [\,]\ in\ x))^\sharp \\ & = & C(V^\sharp, y.(N : let\ x = [\,]\ in\ x)^\sharp) \\ & \twoheadrightarrow_{Q^{**}} & C(V^\sharp, y.N^\sharp), \text{ por H.I. em } N \\ & = & (let\ y = V\ in\ N)^\sharp.\end{array}$$

* Caso $M = (let\ z = yW\ in\ N)$:

$$\begin{array}{rcl} & & ((let\ z = yW\ in\ N) : let\ x = [\,]\ in\ x)^\sharp \\ & = & (let\ z = yW\ in\ (N : let\ x = [\,]\ in\ x))^\sharp \\ & = & y(W^\natural, z.(N : let\ x = [\,]\ in\ x)^\sharp) \\ & \twoheadrightarrow_{Q^{**}} & y(W^\natural, y.N^\sharp), \text{ por H.I. em } N, \\ & = & (let\ z = yW\ in\ N)^\sharp.\end{array}$$

* Caso $M = (let\ y = VW\ in\ N)$, onde V é uma abstração:

$$\begin{array}{rcl} & & ((let\ y = VW\ in\ N) : let\ x = [\,]\ in\ x)^\sharp \\ & = & (let\ y = VW\ in\ (N : let\ x = [\,]\ in\ x))^\sharp \\ & = & C(V^\sharp, z.z(W^\natural, y.(N : let\ x = [\,]\ in\ x)^\sharp)) \\ & \twoheadrightarrow_{Q^{**}} & C(V^\sharp, z.z(W^\natural, y.N^\sharp)), \text{ por H.I. em } N, \\ & = & (let\ y = VW\ in\ N)^\sharp.\end{array}$$

– Caso $K = (let\ y = [\,]\ in\ P)$:

Queremos mostrar que $(M : let\ x = [\,]\ in\ (let\ y = x\ in\ P))^\sharp \twoheadrightarrow_{Q^{**}} (M : let\ y = [\,]\ in\ P)^\sharp$, onde $x \notin LIV(K)$.

* Caso $M = V$:

$$\begin{array}{rcl}(V : let\ x = [\,]\ in\ (let\ y = x\ in\ P))^\sharp & = & (let\ x = V\ in\ (let\ y = x\ in\ P))^\sharp \\ & = & C(V^\sharp, x.(let\ y = x\ in\ P)^\sharp) \\ & = & C(V^\sharp, x.C(x^\sharp, y.P^\sharp)) \\ & \to_{\sigma_v} & C(V^\sharp, x.[x/y]P^\sharp) \\ & = & C(V^\sharp, y.P^\sharp)\end{array}$$

$$
\begin{array}{rl}
= & (let\ y = V\ in\ P)^\sharp \\
= & (V : let\ y = [\,]\ in\ P)^\sharp.
\end{array}
$$

* Caso $M = zW$:

$$
\begin{array}{rl}
(zW : let\ x = [\,]\ in\ (let\ y = x\ in\ P))^\sharp & = (let\ x = zW\ in\ (let\ y = x\ in\ P))^\sharp \\
& = z(W^\natural, x.(let\ y = x\ in\ P)^\sharp) \\
& = z(W^\natural, x.C(x^\sharp, y.P^\sharp)) \\
& \to_{\sigma_v} z(W^\natural, x.[x/y]P^\sharp) \\
& = z(W^\natural, y.P^\sharp) \\
& = (let\ y = zW\ in\ P)^\sharp \\
& = (zW : let\ y = [\,]\ in\ P)^\sharp.
\end{array}
$$

* Caso $M = VW$, onde V é uma abstração:

$$
\begin{array}{rl}
& (VW : let\ x = [\,]\ in\ (let\ y = x\ in\ P))^\sharp \\
= & (let\ x = VW\ in\ (let\ y = x\ in\ P))^\sharp \\
= & C(V^\sharp, z.z(W^\natural, x.(let\ y = x\ in\ P)^\sharp)) \\
= & C(V^\sharp, z.z(W^\natural, x.C(x^\sharp, y.P^\sharp))) \\
\to_{\sigma_v} & C(V^\sharp, z.z(W^\natural, x.[x/y]P^\sharp)) \\
= & C(V^\sharp, z.z(W^\natural, y.P^\sharp)) \\
= & (let\ y = VW\ in\ P)^\sharp \\
= & (VW : let\ y = [\,]\ in\ P)^\sharp.
\end{array}
$$

* Caso $M = (let\ z = V\ in\ N)$:

$$
\begin{array}{rl}
& ((let\ z = V\ in\ N) : let\ x = [\,]\ in\ (let\ y = x\ in\ P))^\sharp \\
= & (let\ z = V\ in\ (N : let\ x = [\,]\ in\ x))^\sharp \\
= & C(V^\sharp, z.(N : let\ x = [\,]\ in\ (let\ y = x\ in\ P))^\sharp) \\
\twoheadrightarrow_{Q^{**}} & C(V^\sharp, z.(N : let\ y = [\,]\ in\ P)^\sharp), \text{ por H.I. em } N, \\
= & (let\ z = V\ in\ (N : let\ y = [\,]\ in\ P))^\sharp \\
= & ((let\ z = V\ in\ N) : let\ y = [\,]\ in\ P)^\sharp.
\end{array}
$$

* Caso $M = (let\ z = uW\ in\ N)$:

$$\begin{array}{rl} & ((let\ z = uW\ in\ N) : let\ x = [\,]\ in\ (let\ y = x\ in\ P))^\sharp \\ = & (let\ z = uW\ in\ (N : let\ x = [\,]\ in\ x))^\sharp \\ = & u(W^\natural, z.(N : let\ x = [\,]\ in\ (let\ y = x\ in\ P))^\sharp) \\ \twoheadrightarrow_{Q^{**}} & u(W^\natural, z.(N : let\ y = [\,]\ in\ P)^\sharp), \text{ por H.I. em } N, \\ = & (let\ z = uW\ in\ (N : let\ y = [\,]\ in\ P))^\sharp \\ = & ((let\ z = uW\ in\ N) : let\ y = [\,]\ in\ P)^\sharp. \end{array}$$

* Caso $M = (let\ y = VW\ in\ N)$, onde V é uma abstração:

$$\begin{array}{rl} & ((let\ z = VW\ in\ N) : let\ x = [\,]\ in\ (let\ y = x\ in\ P))^\sharp \\ = & (let\ z = VW\ in\ (N : let\ x = [\,]\ in\ x))^\sharp \\ = & C(V^\sharp, u.u(W^\natural, z.(N : let\ x = [\,]\ in\ (let\ y = x\ in\ P))^\sharp)) \\ \twoheadrightarrow_{Q^{**}} & C(V^\sharp, u.u(W^\natural, z.(N : let\ y = [\,]\ in\ P)^\sharp)), \text{ por H.I. em } N, \\ = & (let\ z = VW\ in\ (N : let\ y = [\,]\ in\ P))^\sharp \\ = & ((let\ z = VW\ in\ N) : let\ y = [\,]\ in\ P)^\sharp. \end{array}$$

- Os outros casos, que dizem respeito ao fecho reflexivo e transitivo, seguem por hipótese de indução.

□

Teorema 21 (Correspondência equacional). *As traduções $\flat$ e $\sharp$ formam uma correspondência equacional entre $\lambda_{Q^{**}}$ e $\lambda_{c^{**}}$, isto é,*

- $M \twoheadrightarrow_{Q^{**}} N \implies M^\flat =_{c^{**}} N^\flat$;
- $M \twoheadrightarrow_{c^{**}} N \implies M^\sharp \twoheadrightarrow_{Q^{**}} N^\sharp$;
- $M \twoheadrightarrow_{Q^{**}} (M^\flat)^\sharp$;
- $(M^\sharp)^\flat \twoheadrightarrow_{c^{**}} M$.

Em particular, $M = (M^\flat)^\sharp$.

Demonstração. A prova segue segundo a definição de correspondência equacional.

Os dois primeiros pontos são diretos pelas proposições 4 e 5, respetivamente.

Falta-nos então provar os dois últimos pontos. Mostraremos, primeiramente, que $(M^\flat)^\sharp = M$. Para isso, é necessário ver que $(V^\flat)^\natural = V$. A prova segue por indução simultânea em M e $V \in \Lambda_{Q^{**}}$.

- Caso $V = x$:

$$(x^\flat)^\natural = x^\natural = x.$$

- Caso $V = \lambda x.P$:

$$\begin{aligned}((\lambda x.P)^\flat)^\natural &= (\lambda x.P^\flat)^\natural \\ &= \lambda x.(P^\flat)^\natural \\ &= \lambda x.P, \text{ por H.I. em } P.\end{aligned}$$

- Caso $M = \uparrow V$:

$$\begin{aligned}((\uparrow V)^\flat)^\sharp &= (V^\flat)^\sharp \\ &= \uparrow((V^\flat)^\natural) \\ &= \uparrow V, \text{ por H.I. em } P.\end{aligned}$$

- Caso $M = x(V, y.N)$:

$$\begin{aligned}((x(V, y.N))^\flat)^\sharp &= (let\ y = xV^\flat\ in\ N^\flat)^\sharp \\ &= x((V^\flat)^\natural, y.(N^\flat)^\sharp) \\ &= x(V, y.N), \text{ por H.I. em } V \text{ e } N.\end{aligned}$$

- Caso $M = C(\uparrow V, y.N)$:

$$\begin{aligned}((C(\uparrow V, y.N))^\flat)^\sharp &= (let\ x = V^\flat\ in\ N^\flat)^\sharp \\ &= C((V^\flat)^\sharp, x.(N^\flat)^\sharp) \\ &= C(\uparrow((V^\flat)^\natural), x.(N^\flat)^\sharp) \\ &= C(\uparrow V, y.N), \text{ por H.I. em } V \text{ e } N.\end{aligned}$$

Por fim, vamos mostrar que $(M^\sharp)^\natural \twoheadrightarrow_{c^{**}} M$. Para isso, é necessário ver que $(V^\natural)^\flat \twoheadrightarrow_{c^{**}} V$. A prova segue por indução simultânea em M e $V \in \Lambda_{c^{**}}$.

- Caso $V = x$:

$$(x^\natural)^\flat = x^\flat = x.$$

- Caso $V = \lambda x.P$:

$$\begin{array}{rcl} ((\lambda x.P)^\natural)^\flat & = & (\lambda x.P^\sharp)^\flat \\ & = & \lambda x.(P^\sharp)^\flat \\ & \twoheadrightarrow_{c^{**}} & \lambda x.P, \text{ por H.I. em } P. \end{array}$$

- Caso $M = V$:

$$\begin{array}{rcl} (V^\sharp)^\flat & = & (\uparrow(V^\natural))^\flat \\ & = & (V^\natural)^\flat \\ & \twoheadrightarrow_{c^{**}} & V, \text{ por H.I. em } V. \end{array}$$

- Caso $M = yW$:

$$\begin{array}{rcl} ((yW)^\sharp)^\flat & = & (y(W^\natural, z.z^\flat))^\flat \\ & = & let\ z = y(W^\natural)^\flat\ in\ (z^\sharp)^\flat \\ & = & let\ z = y(W^\natural)^\flat\ in\ z \\ & \twoheadrightarrow_{c^{**}} & let\ z = yW\ in\ z, \text{ por H.I. em } W, \\ & \rightarrow_{\eta_{let}} & yW. \end{array}$$

- Caso $M = VW$, onde V é uma abstração:

$$\begin{array}{rcl} ((VW)^\sharp)^\flat & = & (C(V^\sharp, y.y(W^\natural, z.z^\sharp)))^\flat \\ & = & let\ y = (V^\sharp)^\flat\ in\ (y(W^\natural, z.z^\sharp))^\flat \\ & = & let\ y = (V^\sharp)^\flat\ in\ (let\ z = y(W^\natural)^\flat\ in\ (z^\sharp)^\flat) \\ & = & let\ y = (V^\sharp)^\flat\ in\ (let\ z = y(W^\natural)^\flat\ in\ z) \\ & \twoheadrightarrow_{c^{**}} & let\ y = V\ in\ (let\ z = yW\ in\ z), \text{ por H.I. em } V \text{ e } W, \end{array}$$

$$\begin{aligned} &\to_{\eta_{let}} \quad let\ y = V\ in\ yW \\ &\to_{\eta_{let}} \quad VW. \end{aligned}$$

- Caso $M = (let\ y = V\ in\ N)$:

$$\begin{aligned} ((let\ y = V\ in\ N)^\sharp)^\flat &= (C(V^\sharp, y.N^\sharp))^\flat \\ &= let\ y = (V^\sharp)^\flat\ in\ (N^\sharp)^\flat \\ &\twoheadrightarrow_{c^{**}} let\ y = V\ in\ N, \text{ por H.I. em } V \text{ e } W. \end{aligned}$$

- Caso $M = (let\ z = yW\ in\ N)$:

$$\begin{aligned} ((let\ z = yW\ in\ N)^\sharp)^\flat &= (y(W^\natural, z.N^\sharp))^\flat \\ &= let\ z = y(W^\natural)^\flat\ in\ (N^\sharp)^\flat \\ &\twoheadrightarrow_{c^{**}} let\ z = yW\ in\ N, \text{ por H.I. em } V \text{ e } W. \end{aligned}$$

- Caso $M = (let\ z = VW\ in\ N)$, onde V é abstração:

$$\begin{aligned} ((let\ z = VW\ in\ N)^\sharp)^\flat &= (C(V^\sharp, y.y(W^\natural, z.N^\sharp)))^\flat \\ &= let\ z = (V^\sharp)^\flat\ in\ (letz = y(W^\natural)^\flat\ in\ (N^\sharp)^\flat) \\ &\twoheadrightarrow_{c^{**}} let\ y = V\ in\ (letz = yW\ in\ N), \text{ por H.I.} \\ &\qquad \text{em } V, W \text{ e } N, \\ &\to_{\beta_{let}} let\ z = VW\ in\ N. \end{aligned}$$

□

5.3.2 Confluência dos núcleos

Sabendo que o cálculo computacional, λ_c, é confluente [13], vamos provar a confluência do núcleo $\lambda_{c^{**}}$. Para isso, consideremos a tradução $*$ apresentada em [16].

Definição 81 (Tradução $*$). *Seja $(\cdot)^* : \Lambda_c \to \Lambda_{c^{**}}$ o mapeamento definido recursivamente do seguinte modo:*

- *Para cada $M \in \Lambda_c$, para cada $K \in \lambda_{c^{**}}$, $M : K$ um termo de $\lambda_{c^{**}}$:*

$$V : K = K[V^\dagger]$$

$$MN : K = M : (let\ x = [\,]\ in\ (xN : K)),\ \textit{se } M \textit{ não é valor}$$

$$VN : K = N : (let\ x = [\,]\ in\ (Vx : K)),\ \textit{se } N \textit{ não é valor}$$

$$VW : K = K[V^\dagger W^\dagger]$$

$$(let\ x = M\ in\ N) : K = M : (let\ x = [\,]\ in\ (N : K))$$

- *Para cada $V \in \Lambda_c$, $V^\dagger$ um valor de $\lambda_{c^{**}}$:*

$$x^\dagger = x$$

$$(\lambda x.M)^\dagger = \lambda x.M^*$$

Por fim, define-se:

$$M^* = M : [\,].$$

Lema 16. *Sejam V, W e $M \in \Lambda_c$. Então,*

$$(i) \qquad ([V/x]W)^\dagger = [V^\dagger/x]W^\dagger;$$

$$(ii) \qquad [V^\dagger/x](M : K) = [V/x]M : [V^\dagger/x]K.$$

Demonstração. A prova segue por indução simultânea em W e $M \in \Lambda_c$.

- Caso $W = x$:

$$\begin{aligned} ([V/x]x)^\dagger &= V^\dagger \\ &= [V^\dagger/x]x \\ &= [V^\dagger/x]x^\dagger. \end{aligned}$$

- Caso $W = y$:

$$\begin{aligned} ([V/x]y)^\dagger &= y^\dagger \\ &= y \\ &= [V^\dagger/x]y \\ &= [V^\dagger/x]y^\dagger \end{aligned}$$

- Caso $W = \lambda y.P$:

$$([V/x](\lambda y.P))^\dagger = (\lambda y.[V/x]P)^\dagger$$

$$
\begin{aligned}
&= \lambda y.([V/x]P)^* \\
&= \lambda y.([V/x]P : [\,]) \\
&= \lambda y.([V/x]P : [V^\dagger/x][\,]) \\
&= \lambda y.[V^\dagger/x](P : [\,]), \text{ por H.I. em } P, \\
&= \lambda y.[V^\dagger/x]P^* \\
&= [V^\dagger/x](\lambda y.P^*) \\
&= [V^\dagger/x](\lambda y.P)^\dagger.
\end{aligned}
$$

- Caso $M = x$:
  - Caso $K = [\,]$:

$$
\begin{aligned}
[V^\dagger/x](x : [\,]) &= [V^\dagger/x]x^\dagger \\
&= [V^\dagger/x]x \\
&= V^\dagger \\
&= V : [\,] \\
&= [V/x]x : [\,] \\
&= [V/x]x : [V^\dagger/x]([\,]).
\end{aligned}
$$

  - Caso $K = (let\ y = [\,]\ in\ N)$:

$$
\begin{aligned}
[V^\dagger/x](x : K) &= [V^\dagger/x](K[x^\dagger]) \\
&= [V^\dagger/x](K[x]) \\
&= [V^\dagger/x](let\ y = x\ in\ N) \\
&= let\ y = [V^\dagger/x]x\ in\ [V^\dagger/x]N \\
&= let\ y = V^\dagger\ in\ [V^\dagger/x]N \\
&= V : (let\ y = [\,]\ in\ [V^\dagger/x]N) \\
&= V : (let\ y = [V^\dagger/x]([\,])\ in\ [V^\dagger/x]N) \\
&= [V/x]x : [V^\dagger/x]K.
\end{aligned}
$$

- Caso $M = z$:

– Caso $K = [\ ]$:

$$\begin{aligned}
[V^\dagger/x](z : [\ ]) &= [V^\dagger/x]z^\dagger \\
&= [V^\dagger/x]z \\
&= z \\
&= z^\dagger \\
&= z : [\ ] \\
&= [V/x]z : [V^\dagger/x]([\ ]).
\end{aligned}$$

– Caso $K = (let\ y = [\ ]\ in\ N)$:

$$\begin{aligned}
[V^\dagger/x](z : K) &= [V^\dagger/x](K[z^\dagger]) \\
&= [V^\dagger/x](K[z]) \\
&= [V^\dagger/x](let\ y = z\ in\ N) \\
&= let\ y = [V^\dagger/x]z\ in\ [V^\dagger/x]N \\
&= let\ y = z\ in\ [V^\dagger/x]N \\
&= z : (let\ y = [\ ]\ in\ [V^\dagger/x]N) \\
&= z : (let\ y = [V^\dagger/x]([\ ])\ in\ [V^\dagger/x]N) \\
&= [V/x]z : [V^\dagger/x]K.
\end{aligned}$$

- Caso $M = \lambda z.P$:

– Caso $K = [\ ]$:

$$\begin{aligned}
[V^\dagger/x](\lambda z.P : [\ ]) &= [V^\dagger/x](\lambda z.P)^\dagger \\
&= [V^\dagger/x](\lambda z.P^*) \\
&= \lambda z.[V^\dagger/x]P^* \\
&= \lambda z.[V^\dagger/x](P : [\ ]) \\
&= \lambda z.([V/x]P : [V^\dagger/x]([\ ])), \text{ por H.I. em } P, \\
&= \lambda z.([V/x]P : [\ ]) \\
&= \lambda z.([V/x]P)^* \\
&= (\lambda z.[V/x]P)^\dagger
\end{aligned}$$

$$\begin{aligned} &= \lambda z.[V/x]P : [\,] \\ &= [V/x](\lambda z.P) : [V^\dagger/x]([\,]). \end{aligned}$$

– Caso $K = (let\ y = [\,]\ in\ N)$:

$$\begin{aligned} [V^\dagger/x](\lambda z.P : K) &= [V^\dagger/x](K[(\lambda z.P)^\dagger]) \\ &= [V^\dagger/x](K[\lambda z.P^*]) \\ &= [V^\dagger/x](let\ y = \lambda z.P^*\ in\ N) \\ &= let\ y = [V^\dagger/x](\lambda z.P^*)\ in\ [V^\dagger/x]N \\ &= let\ y = \lambda z.[V^\dagger/x]P^*\ in\ [V^\dagger/x]N \\ &= let\ y = \lambda z.[V^\dagger/x](P : [\,])\ in\ [V^\dagger/x]N \\ &= let\ y = \lambda z.([V/x]P : [V^\dagger/x]([\,]))\ in\ [V^\dagger/x]N, \text{ por H.I. em } P, \\ &= let\ y = \lambda z.([V/x]P : [\,])\ in\ [V^\dagger/x]N \\ &= let\ y = \lambda z.([V/x]P)^*\ in\ [V^\dagger/x]N \\ &= let\ y = (\lambda z.[V/x]P)^\dagger\ in\ [V^\dagger/x]N \\ &= \lambda z.[V/x]P : let\ y = [\,]\ in\ [V^\dagger/x]N \\ &= \lambda z.[V/x]P : let\ y = [V^\dagger/x]([\,])\ in\ [V^\dagger/x]N \\ &= [V/x](\lambda z.P) : [V^\dagger/x]K. \end{aligned}$$

- Caso $M = W_1W_2$:

  – Caso $K = [\,]$:

$$\begin{aligned} [V^\dagger/x](W_1W_2 : [\,]) &= [V^\dagger/x](W_1^\dagger W_2^\dagger) \\ &= [V^\dagger/x]W_1^\dagger[V^\dagger/x]W_2^\dagger \\ &= ([V/x]W_1)^\dagger([V/x]W_2)^\dagger, \text{ por H.I. em } W_1 \text{ e } W_2, \\ &= [V/x]W_1[V/x]W_2 : [\,] \\ &= [V/x](W_1W_2) : [V^\dagger/x]([\,]). \end{aligned}$$

  – Caso $K = (let\ y = [\,]\ in\ N)$:

$$[V^\dagger/x](W_1W_2 : K) = [V^\dagger/x](K[W_1^\dagger W_2^\dagger])$$

$$
\begin{aligned}
&= [V^\dagger/x](K[\lambda z.P^*]) \\
&= [V^\dagger/x](let\ y = W_1^\dagger W_2^\dagger\ in\ N) \\
&= let\ y = [V^\dagger/x](W_1^\dagger W_2^\dagger)\ in\ [V^\dagger/x]N \\
&= let\ y = [V^\dagger/x]W_1^\dagger [V^\dagger/x]W_2^\dagger\ in\ [V^\dagger/x]N \\
&= let\ y = ([V/x]W_1)^\dagger([V/x]W_2)^\dagger\ in\ [V^\dagger/x]N, \\
&\quad \text{por H.I. em } W_1 \text{ e } W_2, \\
&= [V/x]W_1[V/x]W_2 : let\ y = [\,]\ in\ [V^\dagger/x]N \\
&= [V/x]W_1[V/x]W_2 : let\ y = [V^\dagger/x]([\,])\ in\ [V^\dagger/x]N \\
&= [V/x](W_1W_2) : [V^\dagger/x]K.
\end{aligned}
$$

- Caso $M = WP$, onde P não é valor:

$$
\begin{aligned}
[V^\dagger/x](WP : K) &= [V^\dagger/x](P : (let\ y = [\,]\ in\ (Wy : K))) \\
&= [V/x]P : [V^\dagger/x](let\ y = [\,]\ in\ (Wy : K)), \text{ por H.I. em } P, \\
&= [V/x]P : let\ y = [\,]\ in\ [V^\dagger/x](Wy : K) \\
&= [V/x]P : let\ y = [\,]\ in\ ([V/x](Wy) : [V^\dagger/x]K), \text{ por H.I. em } yW, \\
&= [V/x]P : let\ y = [\,]\ in\ ([V/x]Wy : [V^\dagger/x]K) \\
&= [V/x]W[V/x]P : [V^\dagger/x]K \\
&= [V/x](WP) : [V^\dagger/x]K.
\end{aligned}
$$

- Caso $M = PQ$, onde P não é valor:

$$
\begin{aligned}
[V^\dagger/x](PQ : K) &= [V^\dagger/x](P : (let\ y = [\,]\ in\ (yQ : K))) \\
&= [V/x]P : [V^\dagger/x](let\ y = [\,]\ in\ (yQ : K)), \text{ por H.I. em } P, \\
&= [V/x]P : let\ y = [\,]\ in\ [V^\dagger/x](yQ : K) \\
&= [V/x]P : let\ y = [\,]\ in\ ([V/x](yQ) : [V^\dagger/x]K), \text{ por H.I. em } yW, \\
&= [V/x]P : let\ y = [\,]\ in\ (y[V/x]Q : [V^\dagger/x]K) \\
&= [V/x]P[V/x]Q : [V^\dagger/x]K \\
&= [V/x](PQ) : [V^\dagger/x]K.
\end{aligned}
$$

- Caso $M = (let\ y = P\ in\ N)$:

$$
[V^\dagger/x]((let\ y = P\ in\ N) : K) = [V^\dagger/x](P : (let\ y = [\,]\ in\ (N : K)))
$$

$$
\begin{aligned}
&= [V/x]P : [V^\dagger/x](let\ y = [\,]\ in\ (N : K))),\\
&\quad \text{por H.I. em } P,\\
&= [V/x]P : let\ y = [\,]\ in\ [V^\dagger/x](N : K)\\
&= [V/x]P : let\ y = [\,]\ in\ ([V/x]N : [V^\dagger/x]K),\\
&\quad \text{por H.I. em } N,\\
&= (let\ y = [V/x]P\ in\ [V/x]N) : [V^\dagger/x]K\\
&= [V/x](let\ y = P\ in\ N) : [V^\dagger/x]K.
\end{aligned}
$$

□

Lema 17. *Seja V e $M \in \Lambda_c$. Então,*

$$([V/x]M)^* = [V^\dagger/x]M^*.$$

Demonstração.

$$
\begin{aligned}
\text{Ora, } ([V/x]M)^* &= [V/x]M : [\,]\\
&= [V/x]M : [V^\dagger/x]([\,])\\
&= [V^\dagger/x](M : [\,]), \text{ pelo lema 16,}\\
&= [V^\dagger/x]M^*.
\end{aligned}
$$

□

Proposição 6. *Sejam M e $N \in \Lambda_c$. Então,*

$$M \to_c N \implies M^* \twoheadrightarrow_{c^{**}} N^*.$$

Demonstração. A prova segue por indução na relação $M \to_c N$.

- Caso β_v:

  Queremos mostrar que $((\lambda x.M)V)^* \twoheadrightarrow_{c^{**}} ([V/x]M)^*$.

$$
\begin{aligned}
((\lambda x.M)V)^* &= (\lambda x.M)V : [\,]\\
&= (\lambda x.M)^\dagger V^\dagger\\
&= (\lambda x.M^*)V^\dagger
\end{aligned}
$$

$$\begin{array}{rl} \rightarrow_{\beta_v} & [V^\dagger/x]M^* \\ = & ([V/x]M)^*, \text{ pelo lema 17.} \end{array}$$

- Caso id:

  Queremos mostrar que $(let\ x = M\ in\ x)^* \twoheadrightarrow_{c^{**}} (M)^*$.

$$\begin{array}{rcl} (let\ x = M\ in\ x)^* & = & (let\ x = M\ in\ x) : [\,] \\ & = & M : let\ x = [\,]\ in\ (x : [\,]) \\ & = & M : let\ x = [\,]\ in\ x \\ \rightarrow_{\eta_{let}} & & M : [\,] \\ & = & M^*. \end{array}$$

- Caso $comp$:

  Queremos mostrar que $(let\ y = (let\ x = M\ in\ N)\ in\ P)^* \twoheadrightarrow_{c^{**}} (let\ x = M\ in\ (let\ y = N\ in\ P))^*$.

$$\begin{array}{rcl} (let\ y = (let\ x = M\ in\ N)\ in\ P)^* & = & (let\ y = (let\ x = M\ in\ N)\ in\ P) : [\,] \\ & = & (let\ x = M\ in\ N) : (let\ y = [\,]\ in\ (P : [\,])) \\ & = & M : (let\ x = [\,]\ in\ (N : (let\ y = [\,]\ in\ (P : [\,])))) \\ & = & M : (let\ x = [\,]\ in\ ((let\ y = N\ in\ P) : [\,])) \\ & = & (let\ x = M\ in\ (let\ y = N\ in\ P)) : [\,] \\ & = & (let\ x = M\ in\ (let\ y = N\ in\ P))^*. \end{array}$$

- Caso let_v:

  Queremos mostrar que $(let\ x = V\ in\ M)^* \twoheadrightarrow_{c^{**}} ([V/x]M)^*$.

$$\begin{array}{rcl} (let\ x = V\ in\ M)^* & = & (let\ x = V\ in\ M) : [\,] \\ & = & V : (let\ x = [\,]\ in\ (M : [\,])) \\ & = & let\ x = V^\dagger\ in\ (M : [\,]) \\ & = & let\ x = V^\dagger\ in\ M^* \\ \rightarrow_{\beta_{let}} & & [V^\dagger/x]M^* \\ & = & ([V/x]M)^*, \text{ pelo lema 17.} \end{array}$$

- Caso let_1:

  Queremos mostrar que $(MN)^* \twoheadrightarrow_{c^{**}} (let\ x = M\ in\ xN)^*$, onde M não é um valor.

$$\begin{aligned}(MN)^* &= MN : [\,] \\ &= M : (let\ x = [\,]\ in\ (xN : [\,])) \\ &= let\ x = M\ in\ xN) : [\,] \\ &= (let\ x = M\ in\ xN)^*.\end{aligned}$$

- Caso let_2:

  Queremos mostrar que $(VN)^* \twoheadrightarrow_{c^{**}} (let\ x = N\ in\ Vx)^*$, onde N não é um valor.

$$\begin{aligned}(VN)^* &= VN : [\,] \\ &= N : (let\ x = [\,]\ in\ (Vx : [\,])) \\ &= let\ x = N\ in\ Vx) : [\,] \\ &= (let\ x = N\ in\ Vx)^*.\end{aligned}$$

- Caso η_v:

  Queremos mostrar que $(\lambda x.Vx)^* \twoheadrightarrow_{c^{**}} V^*$, onde $x \notin LIV(V)$.

$$\begin{aligned}(\lambda x.Vx)^* &= \lambda x.(Vx)^* \\ &= \lambda x.(Vx : [\,]) \\ &= \lambda x.V^\dagger x \\ &\to_{\eta_v} V^\dagger \\ &= V : [\,] \\ &= V^*.\end{aligned}$$

□

Proposição 7. *Seja $M \in \Lambda_c$ e $K \in \Lambda_{c^{**}}$. Então,*

$$(i) \qquad V \twoheadrightarrow_c V^\dagger;$$

$$(ii) \qquad K[M] \twoheadrightarrow_c M : K.$$

Demonstração. A prova segue por indução simultânea em $M \in \Lambda_c$ e $K \in \Lambda_{c^{**}}$.

- Caso $V = x$:

$$x = x^\dagger.$$

- Caso $V = \lambda x.P$:

$$\begin{aligned} \lambda x.P &\twoheadrightarrow_c \lambda x.P : [\,], \text{ por H.I. em } P, \\ &= \lambda x.P^* \\ &= (\lambda x.P)^\dagger. \end{aligned}$$

- $M = V$:

  - Caso $K = [\,]$:

    Queremos mostrar que $V \twoheadrightarrow_c V : [\,]$.

$$\begin{aligned} V &= V^\dagger, \text{ por H.I. em } V, \\ &= V : [\,]. \end{aligned}$$

  - Caso $K = (let\ y = [\,]\ in\ N)$:

    Queremos mostrar que $let\ y = V\ in\ N \twoheadrightarrow_c V : let\ y = [\,]\ in\ N$.

$$\begin{aligned} let\ y = V\ in\ N &= let\ y = V^\dagger\ in\ N, \text{ por H.I. em V}, \\ &= V : let\ y = [\,]\ in\ N. \end{aligned}$$

- $M = VW$:

  - Caso $K = [\,]$:

    Queremos mostrar que $VW \twoheadrightarrow_c VW : [\,]$.

$$\begin{aligned} VW &\twoheadrightarrow_c V^\dagger W^\dagger, \text{ por H.I. em } V \text{ e } W, \\ &= VW : [\,]. \end{aligned}$$

  - Caso $K = (let\ y = [\,]\ in\ N)$:

    Queremos mostrar que $let\ y = VW\ in\ N \twoheadrightarrow_c VW : let\ y = [\,]\ in\ N$.

$$\begin{aligned} let\ y = VW\ in\ N &\twoheadrightarrow_c let\ y = V^\dagger W^\dagger\ in\ N \text{ , por H.I. em } V \text{ e } W, \\ &= VW : let\ y = [\,]\ in\ N. \end{aligned}$$

- $M = VQ$, onde Q não é um valor:
  - Caso $K = [\,]$:

    Queremos mostrar que $VQ \twoheadrightarrow_c VQ : [\,]$.

$$\begin{aligned} VQ &\rightarrow_{let_2} let\ x = Q\ in\ Vx \\ &\twoheadrightarrow_c Q : let\ x = [\,]\ in\ Vx, \text{ por H.I. em } Q, \\ &\twoheadrightarrow_c Q : let\ x = [\,]\ in\ (Vx : [\,]), \text{ por H.I. em } V, \\ &= VQ : [\,]. \end{aligned}$$

  - Caso $K = (let\ y = [\,]\ in\ N)$:

    Queremos mostrar que $let\ y = VQ\ in\ N \twoheadrightarrow_c VQ : let\ y = [\,]\ in\ N$.

$$\begin{aligned} let\ y = VQ\ in\ N &\rightarrow_{let_2} let\ y = (let\ x = Q\ in\ Vx)\ in\ N \\ &\rightarrow_{comp} let\ x = Q\ in\ (let\ y = Vx\ in\ N) \\ &\twoheadrightarrow_c Q : let\ x = [\,]\ in\ (let\ y = Vx\ in\ N), \text{ por H.I. em } Q, \\ &\twoheadrightarrow_c Q : let\ x = [\,]\ in\ (Vx : let\ y = [\,]\ in\ N), \text{ por H.I. em } V, \\ &= VQ : let\ y = [\,]\ in\ N. \end{aligned}$$

- $M = PQ$, onde P não é um valor:
  - Caso $K = [\,]$:

    Queremos mostrar que $PQ \twoheadrightarrow_c PQ : [\,]$.

$$\begin{aligned} PQ &\rightarrow_{let_1} let\ x = P\ in\ xQ \\ &\twoheadrightarrow_c P : let\ x = [\,]\ in\ xQ, \text{ por H.I. em } P, \\ &\twoheadrightarrow_c P : let\ x = [\,]\ in\ (xQ : [\,]), \text{ por H.I. em } Q, \\ &= PQ : [\,]. \end{aligned}$$

  - Caso $K = (let\ y = [\,]\ in\ N)$:

    Queremos mostrar que $let\ y = PQ\ in\ N \twoheadrightarrow_c PQ : let\ y = [\,]\ in\ N$.

$$\begin{aligned} let\ y = PQ\ in\ N &\rightarrow_{let_1} let\ y = (let\ x = P\ in\ xQ)\ in\ N \\ &\rightarrow_{comp} let\ x = P\ in\ (let\ y = xQ\ in\ N) \end{aligned}$$

$$\begin{aligned} &\twoheadrightarrow_c && P : let\ x = [\,]\ in\ (let\ y = xQ\ in\ N), \text{ por H.I. em } P, \\ &\twoheadrightarrow_c && P : let\ x = [\,]\ in\ (xQ : let\ y = [\,]\ in\ N), \text{ por H.I. em } Q, \\ &= && PQ : let\ y = [\,]\ in\ N. \end{aligned}$$

- $M = (let\ x = P\ in\ Q)$:
  - Caso $K = [\,]$:

    Queremos mostrar que $let\ x = P\ in\ Q \twoheadrightarrow_c let\ x = P\ in\ Q : [\,]$.

$$\begin{aligned} let\ x = P\ in\ Q &\twoheadrightarrow_c P : let\ x = [\,]\ in\ Q, \text{ por H.I. em } P, \\ &\twoheadrightarrow_c P : let\ x = [\,]\ in\ (Q : [\,]), \text{ por H.I. em } Q, \\ &= let\ x = P\ in\ Q : [\,]. \end{aligned}$$

  - Caso $K = (let\ y = [\,]\ in\ N)$:

    Queremos mostrar que $let\ y = (let\ x = P\ in\ Q)\ in\ N \twoheadrightarrow_c (let\ x = P\ in\ Q) : let\ y = [\,]\ in\ N$.

$$\begin{aligned} let\ y = (let\ x = P\ in\ Q)\ in\ N &\rightarrow_{comp} let\ x = P\ in\ (let\ y = Q\ in\ N) \\ &\twoheadrightarrow_c P : let\ x = [\,]\ in\ (let\ y = Q\ in\ N), \\ &\quad \text{por H.I. em } P, \\ &\twoheadrightarrow_c P : let\ x = [\,]\ in\ (Q : let\ y = [\,]\ in\ N), \\ &\quad \text{por H.I. em } Q, \\ &= (let\ x = P\ in\ Q) : let\ y = [\,]\ in\ N. \end{aligned}$$

□

Corolário 4. *Seja $M \in \Lambda_c$. Então,*

$$M \twoheadrightarrow_c M^*.$$

Demonstração. Segue diretamente da proposição 7, tomando $K = [\,]$. □

Teorema 22 (Confluência de $\lambda_{c^{**}}$). *As relações do cálculo-$\lambda_{c^{**}}$ são confluentes.*

Demonstração. Sejam M, N_1 e $N_2 \in \Lambda_{c^{**}}$. Suponhamos que $M \twoheadrightarrow_{c^{**}} N_1$ e $M \twoheadrightarrow_{c^{**}} N_2$. Queremos mostrar que existe $P \in \Lambda_{c^{**}}$ tal que $N_1 \twoheadrightarrow_{c^{**}} P$ e $N_2 \twoheadrightarrow_{c^{**}} P$.

Ora, se M, N_1 e $N_2 \in \Lambda_{c^{**}} \subseteq \Lambda_c$, pelo que $M \twoheadrightarrow_c N_1$ e $M \twoheadrightarrow_c N_2$. Uma vez que λ_c é confluente, existe $Q \in \Lambda_c$ tal que $N_1 \twoheadrightarrow_c Q$ e $N_2 \twoheadrightarrow_c Q$. Daqui segue que $Q \twoheadrightarrow_c Q^*$, pela corolário 4. Então, basta tomar $P = Q^*$.

Com efeito, de $N_1 \twoheadrightarrow_c Q$ segue que $N_1^* \twoheadrightarrow_{c^{**}} Q^*$, pela proposição 6. Uma vez que $N_1^* = N_1$ e $Q^* = P$, então $N_1 \twoheadrightarrow_{c^{**}} P$.

Seguindo a mesma linha de pensamento, de $N_2 \twoheadrightarrow_c Q$ segue que $N_2 \twoheadrightarrow_{c^{**}} P$. □

Neste capítulo definimos uma versão simplificada do cálculo de Dyckhoff-Lengrand, um sistema que vimos ter origem na teoria da demonstração, bem como o seu núcleo e provámos que este forma uma correspondência equacional com o núcleo do cálculo computacional.

Além disso, concluímos que $\lambda_{c^{**}}$ é confluente, mas nada se pode concluir relativamente à confluência de $\lambda_{Q^{**}}$ com os resultados obtidos. No entanto, é de notar que, se os núcleos formassem uma pré-conexão de Galois entre si, o resultado sairia diretamente pelo teorema 4.

Capítulo 6

Conclusão

Nesta dissertação, estudámos o cálculo-λ e o cálculo de Plotkin e vimos como estes modelam linguagens de programação *call-by-value* e *call-by-name*, respetivamente. De seguida, vimos como a linguagem *call-by-name* simula a linguagem *call-by-value* e vice-versa. Posto isso, estudámos a extensão Sabry-Felleisen, o cálculo computacional e o cálculo-λ_Q. Concluímos que os três sistemas são, de facto, completos para a semântica CPS e, portanto, neste aspeto, são melhorias equivalentes do cálculo de Plotkin. Em [6], Dyckhoff e Lengrand mostram que o cálculo-λ_Q é completo para a semântica CPS, pelo que não nos dedicamos a estudar esse tópico neste trabalho.

Sabry e Wadler fazem notar que a extensão Sabry-Felleisen e o cálculo computacional provam as mesmas igualdades, mas não provam as mesmas reduções, afirmando que o cálculo computacional é preferível [16]. Isto pode ser observado pelos teoremas 18 e 19. Não conhecemos uma prova direta deste facto. Com efeito, estudámos a relação entre ambos os sistemas e verificámos que formam uma correspondência equacional.

No que diz respeito ao cálculo-λ_Q, Dyckhoff e Lengrand mostram que existe uma reflexão do cálculo CPS no cálculo-λ_Q [16], tal como acontece no cálculo computacional. Dyckhoff e Lengrand provam ainda que existe uma correspondência equacional entre o cálculo-λ_Q e o cálculo computacional, recorrendo, para isso, ao cálculo CPS. Neste trabalho, estudámos diretamente a relação dos núcleos destes sistemas, sendo que primeiramente simplificámos o cálculo-λ_Q e depois definimos o seu núcleo. De seguida, provámos que existe uma correspondência equacional entre o núcleo do cálculo-λ_{Q^*} e o núcleo do cálculo computacional. Além disso, provámos que o núcleo do cálculo computacional é confluente, pelo que é consistente.

Infelizmente, devido à escassez do tempo, não conseguimos estudar a confluência da extensão Sabry-Felleisen, pelo que consideramos fazê-lo no futuro, para termos de comparação com os outros sistemas. Além disso, pretendemos continuar a estudar a relação entre o núcleo do cálculo computacional e o núcleo do cálculo-λ_{Q^*} e descobrir se, efetivamente, existe uma pré-conexão de Galois entre os mesmos, de forma a concluir que o cálculo-$\lambda_{Q^{**}}$ é confluente.

Por fim, na abordagem de Plotkin, um programa é um termo fechado e, por vezes, é necessário considerar termos abertos, isto é, termos com variáveis livres. A máquina abstrata do *Coq*, um assistente de prova, considera termos abertos, por exemplo. A generalização da abordagem de Plotkin a termos abertos levanta problemas imediatos, originando uma outra linha de investigação, que propõe outras alternativas ao cálculo de Plotkin [1]. Posto isto, seria de igual forma interessante seguir esta nova linha de investigação.

Bibliografia

[1] B. Accattoli e G. Guerrieri. *Open Call-by-Value (Extended Version)*. 2016.

[2] H. P. Barendregt. *The lambda calculus - its syntax and semantics*. Vol. 103. Studies in logic and the foundations of mathematics. North-Holland, 1985. isbn: 978-0-444-86748-3.

[3] H. Barendregt. "The impact of the lambda calculus in logic and computer science". Em: *Bull. Symb. Log.* 3.2 (1997), pp. 181–215.

[4] A. Church. "The calculi of lambda-conversion". Em: 1941.

[5] H. B. Curry. "Functionality in Combinatory Logic*". Em: *Proceedings of the National Academy of Sciences* 20.11 (1934), pp. 584–590.

[6] R. Dyckhoff e S. Lengrand. "Call-by-Value lambda-calculus and LJQ". Em: *J. Log. Comput.* 17.6 (2007), pp. 1109–1134.

[7] R. Dyckhoff e S. Lengrand. "LJQ: A Strongly Focused Calculus for Intuitionistic Logic". Em: *Logical Approaches to Computational Barriers, Second Conference on Computability in Europe, CiE 2006, Swansea, UK, June 30-July 5, 2006, Proceedings*. Ed. por A. Beckmann, U. Berger, B. Löwe e J. V. Tucker. Vol. 3988. Lecture Notes in Computer Science. Springer, 2006, pp. 173–185.

[8] G. Gentzen. "Untersuchungen über das logische Schließen I". Em: *Mathematische Zeitschrift* 39 (1935), pp. 176–210. url: `http://eudml.org/doc/168546`.

[9] J.-Y. Girard, P. Taylor e Y. Lafont. "Proofs and types". Em: 1989.

[10] J. Hatcliff e O. Danvy. "Thunks and the lambda-Calculus". Em: *J. Funct. Program.* 7.3 (1997), pp. 303–319.

[11] J. R. Hindley e J. P. Seldin. *Lambda-calculus and combinators, an introduction.* eng. Cambridge University Press, 2008. isbn: 9780521898850.

[12] P. J. Landin. “The Mechanical Evaluation of Expressions”. Em: *The Computer Journal* 6.4 (1964), pp. 308–320.

[13] E. Moggi. “Computational Lambda-Calculus and Monads”. Em: IEEE Computer Society Press, 1988, pp. 14–23.

[14] G. D. Plotkin. “Call-by-Name, Call-by-Value and the lambda-Calculus”. Em: *Theor. Comput. Sci.* 1.2 (1975), pp. 125–159.

[15] A. Sabry e M. Felleisen. “Reasoning about Programs in Continuation-Passing Style”. Em: *LISP Symb. Comput.* 6.3-4 (1993), pp. 289–360.

[16] A. Sabry e P. Wadler. “A Reflection on Call-by-Value”. Em: *ACM Trans. Program. Lang. Syst.* 19.6 (1997), pp. 916–941.

[17] M. H. Sørensen e P. Urzyczyn. *Lectures on the Curry-Howard isomorphism*. Amsterdam; Oxford: Elsevier, 2006. isbn: 9780444520777.