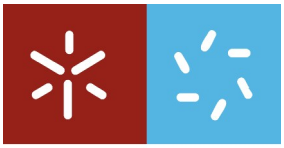


Universidade do Minho
Escola de Ciências

Centro de Matemática

Bolsa de Integração na Investigação 2010

Interpretações do λ-calculus em Redes de Interacção

Victor C. Miraldo

Licenciatura em Ciências da Computação

Orientadores: José Carlos Espírito Santo

Jorge Sousa Pinto

Conteúdo

Abstract

The interaction nets are a graphical formalism, invented by Yves Lafont in 1990 as a proof nets generalization. Since it constitutes a universal model of computation and the ease of a parallel implementation, they showed to be of direct interest in the implementation of functional languages. The λ-calculus, on the other hand, is the theoretical basis of functional programming languages, and can be translated into interaction nets. The project consists of studying interpretations of the known λ-calculus on interaction nets, considering to what extent such interpretations are sensitive to variations of the λ-calculus. Known translations were divided into two classes: direct and logical, and in this paper we present both a short survey of the translation process behind the logical interpretations and a detailed study of a direct interpretation. The project was supervised by Professors Jorge Sousa Pinto and José Carlos Espírito Santo as part of a Initiation into Research Scholarship, provided by the Center for Mathematics, University of Minho.

Resumo

As redes de interacção são um formalismo gráfico, inventadas por Yves Lafont em 1990 como uma generalização das redes de prova. Por constituirem um modelo universal de computação e pela facilidade de uma implementação paralela, mostraram-se imediatamente interessantes na implementação de linguagens funcionais. O λ-calculus, por outro lado, é a base teórica das linguagens de programação funcionais, e pode ser traduzido para redes de interacção. O projecto consiste em estudar as interpretações conhecidas do λ-calculus sobre redes de interacção, considerando em que medida tais interpretações são sensíveis à variantes do λ-calculus. Agrupou-se as traduções conhecidas em duas classes: lógicas e directas, e neste documento apresentamos um pequeno *survey* do processo de tradução por trás das interpretações lógicas e de um estudo detalhado de uma interpretação directa. O projecto foi orientado pelos Professores Jorge Sousa Pinto e José Carlos Espírito Santo no âmbito de uma Bolsa de Iniciação à Investigação, fornecida pelo Centro de Matemática da Universidade do Minho.

Capítulo 1

Introdução

O lambda calculus foi apresentado e criado por Alonzo Church, na década de 1930, num artigo [Chu32]. Um dos principais objectivos de Church era construir um sistema formal para os fundamentos da Matemática, criando um sistema que trabalhe tanto com funções como com noções lógicas. Foi descoberto que o sistema original era inconsistente, em 1935. Church abandonou a ideia em 1936, separando a parte consistente do sistema, que é hoje chamado de λ-calculus sem tipos, e concentrou os seus estudos sobre a computabilidade [Chu36b, Chu36a]. A noção de lambda definível é a base teórica por trás das linguagens de programação funcionais e o cálculo caracteriza um modelo universal de computação.

Em 1940, Church introduziu um sistema computacionalmente mais fraco, porém logicamente consistente, conhecido por λ-calculus com tipos simples [Chu40]. Posteriormente foi descoberto que, no entanto, existem outras maneiras consistentes de representar as noções lógicas no lambda calculus de forma a que uma base para a Matemática seja obtida. Tais sistemas são largamente utilizados em sistemas de dedução automática e fornecem a base para as linguagens de programação funcionais.

As redes de interacção são um formalismo gráfico inspirado nas redes de prova da Lógica Linear e constituem um modelo universal de computação. Este documento descreve o processo de tradução do lambda-calculus para as redes de interacção, e enquadra-se como relatório final do projecto *interpretações do λ-calculus em redes de interacção*, orientado pelos Professores Jorge Sousa Pinto e José Carlos Espírito Santo. O objectivo do projecto é estudar as interpretações conhecidas e modificá-las para se adequarem a variante do λ-calculus adoptada.

Este documento divide-se em duas grandes partes, inicialmente descrevemos o longo processo de tradução do λ-calculus para as redes de interacção,

começamos por apresentar o lambda-calculus, com alguns resultados importantes, nos capítulos 2 e 3. A tradução começa, de facto, pelo isomorfismo de Curry-Howard, secção 3.2, onde estabelecemos a ponte entre a Dedução Natural e o lambda-calculus, saindo do mundo dos termos e entrando no mundo das derivações. Da Dedução Natural traduz-se para o Cálculo de Sequentes, capítulo 3.3, donde traduzimos para a lógica linear, capítulo 4, e consequentemente, para as Redes de Prova. Uma segunda parte apresentamos um estudo detalhado de uma tradução directa do λ-calculus para as redes de interacção, secção 5.3.

Capítulo 2

O lambda-calculus sem tipos

2.1 Preliminares

Aquilo que normalmente chamamos de lambda-calculus é uma colecção de diversos sistemas formais baseados na notação inventada por Alonzo Church. Todos esses sistemas, apesar de sintaticamente diferentes, tem um principio básico em comum, foram todos desenhados para descrever as formas mais básicas pelas quais funções e operadores podem ser combinados para formarem outros operadores.

Neste capítulo vamos apresentar a variação mais simples do lambda-calculus, a noção de confluência sobre relação de redução que da o poder à essa versão mais simples com alguns resultados obtidos ao longo do semestre acerca do mesmo tema. Vamos estudar também as duas estratégias de redução mais simples, *call-by-name* e *call-by-value*.

Definição 2.1 (lambda-termos)**.** Sejam $\mathcal{V} = \{v_1, v_2, \ldots\}$ um conjunto infinito de variáveis, $\mathcal{C} = \{c_1, c_2, \ldots\}$ um conjunto de constantes tal que $\mathcal{V} \cap \mathcal{C} = \emptyset$ (Se $\mathcal{C} = \emptyset$ dizemos que o sistema é *puro*, dizemos que o sistema é *aplicado* caso contrário). O conjunto Λ dos lambda-termos é definido indutivamente por:

(i) Todas as variáveis e todas as constantes são lambda-termos. Tais termos são chamados *átomos*.

(ii) Sejam M, N lambda-termos, então (MN) também é um lambda-termo. Tais termos são chamados *aplicações*.

(iii) Sejam M um lambda-termo e x uma variável, então $(\lambda x.M)$ é um lambda termo, chamado *abstracção*.

Notação 2.1. De forma a simplificar a escrita e a leitura de lambda-termos, vamos fixar alguma notação:

(i) Letras minúsculas $x, y, z, \ldots$ usualmente representam variáveis, enquanto letras maiúsculas $M, N, P, Q, \ldots$ representam termos.

(ii) A aplicação é associativa à esquerda, isto é, o termo $M_1M_2M_3M_4$ representa $(((M_1M_2)M_3)M_4)$.

(iii) A abstracção é associativa à direita, isto é, o termo $\lambda xyz.K$ representa $(\lambda x.(\lambda y.(\lambda z.K)))$

(iv) Dois termos M, N sintáticamente iguais são notados por $M \equiv N$

Num termo como $\lambda yz.x(yz)$ dizemos que as variáveis y e z ocorrem como variáveis ligadas e a variável x possui uma ocorrência livre. Tais categorizações de ocorrências de variáveis motivam a seguinte definição:

Definição 2.2 (Variáveis Livres e Ligadas)**.** Seja M um lambda-termo. Os conjuntos $FV(M)$ e $BV(M)$ são definidos recursivamente por:

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

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

Pela definição acima, podemos dizer que o operador λ *liga* uma variável, mas como mencionamos na definição 2.1 possuímos infinitas variáveis. Note que a mesma variável pode possuir ocorrências livres e ligadas no mesmo termo, por exemplo: $\lambda k.n(\lambda n.kn)$. É cabível então nos perguntarmos se os termos $M = \lambda yz.x(yz)$ e $N = \lambda yk.x(yk)$ são iguais. Óbviamente que não são sintáticamente iguais, mas é evidente que a sua construção é muito semelhante e difere apenas no nome de uma ou mais variáveis. Nesta condição, dizemos que M e N são α-equivalentes e notamos por $M \equiv_\alpha N$.

É importante notar que as α-converssões podem trazer alguns problemas com elas, tal como a captura de nomes. Por exemplo, se renomearmos a variável x do termo $(\lambda y.yx)$ para y, vamos obter um termo com um significado completamnte diferente, uma vez que y possui uma ocorrência ligada no termo. É preciso ter um certo nível de atenção ao utilizá-las.

Definição 2.3 (Substituição)**.** Sejam M e N dois lambda-termos tais que $FV(N) \cap BV(M) = \emptyset$ e x uma variável que ocorra livre em M. Então, a

substituiçao de N por x em M, notada por $[N/x]M$ é, informalmente, o resultado de substituir todas as ocorrências livres de x em M por N. Tal operação pode ser definida recursivamente por:

$$
\begin{aligned}
&[N/x]x = N \\
&[N/x]y = y,\ y \neq x \\
&[N/x](M_1 M_2) = [N/x]M_1[N/x]M_2 \\
&[N/x](\lambda y.M) = (\lambda y.[N/x]M), y \neq x
\end{aligned}
$$

2.2 Redução Beta e Confluência

Prosseguiremos por estudar o processo de cálculo. Tal processo pode ser visto como uma computação num termo. Se imaginarmos uma função $f(x) = x + 3$ e calcularmos $f(2)$, o processo que fazemos implicitamente é procurar pela variável x no corpo da função e substituir todas as ocorrências encontradas por 2, resultando em $2 + 3$.

No lambda calculus não é diferente, um termo com a forma $(\lambda x.M)N$ é chamado um *β-redex*, e pode ser reduzido para $[N/x]M$. Se um dado lambda termo não contém nenhum β-redex, dizemos que esse termo está na *forma β-normal*.

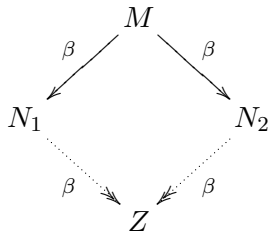
Definição 2.4 (β-redução)**.** Sejam $M, N, M' \in \Lambda$ e x uma variável. Vamos definir por indução em M a seguinte relação binária em Λ.

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

$$
\frac{M \to_\beta M'}{MN \to_\beta M'N} \qquad \frac{M \to_\beta M'}{NM \to_\beta NM'}
$$

Notamos por $M \twoheadrightarrow_\beta N$ se N é obtido a partir de zero ou mais β-reduções de M. Define-se uma β-igualdade de M e N e nota-se por $M =_\beta N$ se $M \twoheadrightarrow_\beta N$ ou $N \twoheadrightarrow_\beta M$.

Teorema 2.1 (Confluência)**.** *Sejam M, N_1 e N_2 lambda-termos. Então se $M \to_\beta N_1$ e $M \to_\beta N_2$ então existe um termo T tal que $N_1 \twoheadrightarrow_\beta Z$ e $N_2 \twoheadrightarrow_\beta T$.*



Teorema 2.2 (Church-Rosser)**.** *Sejam M e N lambda termos tais que $M =_\beta N$. Então existe um Z tal que $M \twoheadrightarrow_\beta Z$ e $N \twoheadrightarrow_\beta Z$.*

Note que os resultados acima são de grande relevância no lambda-calculus. Permitem provar, por exemplo, que se um lambda-termo possui uma forma normal, ela é única. Outro resultado de grande relevância demonstrado a custa dos mesmos teoremas é a consistência do lambda-calculus [Bar84]. Note que existem termos que não possuem formas normais, como por exemplo $(\lambda x.x\ x)(\lambda x.x\ x)$.

Duas demonstrações do teorema 2.1 estão disponíveis, respectivamente, em [HS86] e [Bar84]. A primeira usa um conceito chamado *minimal complete development* e a segunda utiliza uma relação de redução $\twoheadrightarrow_1$. Ambos conceitos são análogos como demonstrado a seguir.

Definição 2.5 (MCD)**.** Define-se por *minimal complete develpment* de um conjunto de redexes $\mathcal{R} = \{r_1, \cdots, r_n\}$ de um λ-termo M a seguinte sequência de β-reduções:

- Primeiro, reduz-se um redex minimal r_i em M (por conveniência, vamos assumir $i = 1$). Tal operação produz no máximo $n-1$ resíduos [1] $r'_2, \cdots, r'_n$ de $r_2, \cdots, r_n$.
- Continua-se por reduzir um resíduo r'_j minimal qualquer, produzindo $n-2$ resíduos.
- Repete-se os passos acima até não sobrarem mais resíduos.
- Efectuam-se zero ou mais α-conversões no termo obtido.

Definição 2.6. Sejam $M, M' \in \Lambda$, então nota-se por:

- $M \rhd^{\mathcal{R}}_{mcd} M'$ se $M' \in \Lambda$ é obtido por um *minimal complete development* de um conjunto $\mathcal{R}$ de redexes de $M \in \Lambda$.
- $M \rhd_{mcd} M'$ um *minimal complete development* de um conjunto de redexes arbitrário.

[1] A definição precisa de reíduos encontra-se em [HS86, Appendix A2]

Definição 2.7. Também podemos definir os passos apresentados na definição 2.5 por meio de uma função recursiva. Seja $M \in \Lambda$, define-se a função MCD por:

$$\begin{aligned}
&\mathrm{MCD}(M, \emptyset) = N, \qquad N \equiv_\alpha \mathrm{M}, \\
&\mathrm{MCD}(M, \{r\} \cup \mathcal{R}) = \mathrm{MCD}(M', \mathcal{R}') \\
&\qquad\qquad r \text{ um redex minimal reduzido em } M \to_\beta M' \\
&\qquad\qquad \text{e } \mathcal{R}' \text{ o conjunto de resíduos dessa redução em } \mathcal{R}.
\end{aligned}$$

E então dizemos que $M \rhd^{\mathcal{R}}_{mcd} M'$ se $\mathrm{MCD}(M, \mathcal{R}) = M'$.

Definição 2.8. Sejam $M, M', N, N' \in \Lambda$ a relação $\twoheadrightarrow_1$ em Λ é definida por:

$$\frac{}{M \twoheadrightarrow_1 M'} \ (1) \qquad \frac{M \twoheadrightarrow_1 M'}{(\lambda x.M) \twoheadrightarrow_1 (\lambda x.M')} \ (2)$$

$$\frac{M \twoheadrightarrow_1 M' \quad N \twoheadrightarrow_1 N'}{MN \twoheadrightarrow_1 M'N'} \ (3) \qquad \frac{M \twoheadrightarrow_1 M' \quad N \twoheadrightarrow_1 N'}{(\lambda x.M)N \twoheadrightarrow_1 [N'/x]M'} \ (4)$$

Teorema 2.3. *Sejam $M, N \in \Lambda$ então $M \rhd_{mcd} N \iff M \twoheadrightarrow_1 N$.*

Dem. Vamos dividir a demonstração em duas partes.

1. Queremos provar que $\forall M, N \in \Lambda \quad M \twoheadrightarrow_1 N \Rightarrow M \rhd_{mcd} N$

Para tal, vamos primeiro demonstrar alguns lemas auxiliares.

Lema 2.1. Seja $M \in \Lambda$ então $M \rhd_{mcd} M$.

Dem. Imediata a partir da definição de MCD. (Basta considerar um MCD de tamanho 0 com nenhuma redução alpha.) □

Lema 2.2. Sejam $M, M' \in \Lambda$ tais que $M \rhd_{mcd} M'$.
Então $(\lambda x.M) \rhd_{mcd} (\lambda x.M')$.

Dem. Uma vez que M e $(\lambda x.M)$ possuem os mesmos redexes a demonstração torna-se imediata. □

Lema 2.3. Sejam $M, M', N, N' \in \Lambda$ tais que $M \rhd^{\mathcal{R}}_{mcd} M'$ e $N \rhd^{\mathcal{S}}_{mcd} N'$.
Então $MN \rhd_{mcd} M'N'$.

Dem. Por hipótese temos que $M \twoheadrightarrow_\beta M^* \equiv_\alpha M'$ e $N \twoheadrightarrow_\beta N^* \equiv_\alpha N'$. Vamos então considerar a seguinte cadeia de reduções:

$$\begin{aligned} MN &\twoheadrightarrow_\beta M^*N && (2.1)\\ &\twoheadrightarrow_\beta M^*N^* && (2.2)\\ &\equiv_\alpha M'N' && (2.3) \end{aligned}$$

Ora, uma vez que os conjuntos $\mathcal{R}$ e $\mathcal{S}$ possuem redexes disjuntos 2 a 2, uma redução de um redex de $\mathcal{R}$ não alterará os redexes de $\mathcal{S}$, o mesmo é válido para o contrário, reduções de $\mathcal{S}$ não alteram $\mathcal{R}$. Portanto podemos unificar as duas β-reduções dos passos (1) e (2), resultando em:

$$\begin{aligned} MN &\twoheadrightarrow_\beta M^*N^*\\ &\equiv_\alpha M'N' \end{aligned}$$

Logo $MN \rhd_{mcd} M'N'$. □

Lema 2.4. Sejam $M, M', N, N' \in \Lambda$ tais que $M \rhd_{mcd} M'$ e $N \rhd_{mcd} N'$. Então $(\lambda x.M)N \rhd_{mcd} [N'/x]M'$.

Dem. Por hipótese temos que $M \twoheadrightarrow_\beta M^* \equiv_\alpha M'$ e $N \twoheadrightarrow_\beta N^* \equiv_\alpha N'$. Segue que

$$\begin{aligned} (\lambda x.M)N &\twoheadrightarrow_\beta (\lambda x.M^*)N^*\\ &\to_\beta [N^*/x]M^*\\ &\equiv_\alpha [N'/x]M' \end{aligned}$$

A redução acima ainda é um MCD, portanto $(\lambda x.M)N \rhd_{mcd} [N'/x]M'$. □

Munido dos lemas (2.1), (2.2), (2.3) e (2.4) e uma indução estrutural em $M \twoheadrightarrow_1 N$ é fácil provar que se $M \twoheadrightarrow_1 N$ então $M \rhd_{mcd} N$, para quaisquer $M, N \in \Lambda$.

Vamos agora provar o outro sentido da equivalência:

2. Queremos provar que $\forall M, N \in \Lambda \quad M \rhd_{mcd} N \Rightarrow M \twoheadrightarrow_1 N$

Para tal, vamos utilizar o calculo λ', com a possibilidade de marcar redexes. O alfabeto é o mesmo do λ-calculus acrescido do simbolo λ'. A linguagem Λ' é definida indutivamente, da mesma forma que o λ-calculus, com apenas uma regra a mais:

$$\forall M, N \in \Lambda' \quad (\lambda' x.M)N \in \Lambda'$$

Seja $Red \subseteq \Lambda$ o conjunto de todos os redexes, vamos definir uma função $\theta : \Lambda \times \mathcal{P}(Red) \to \Lambda'$ recursivamente por:

$$\begin{array}{lcl} \theta(x, \mathcal{R}) & = & x \\ \theta(MN, \mathcal{R}) & = & \theta(M, \mathcal{R})\theta(N, \mathcal{R}), M \notin \mathcal{R} \\ \theta((\lambda x.M), \mathcal{R}) & = & (\lambda x.\theta(M, \mathcal{R})) \\ \theta((\lambda x.M)N, \mathcal{R}) & = & (\lambda' x.\theta(M, \mathcal{R}))\ \theta(N, \mathcal{R}), M \in \mathcal{R} \end{array}$$

(Note que θ é a função que marca os redexes de $\mathcal{R}$ em um termo M.)
E uma função $|\cdot| : \Lambda' \to \Lambda$ que remove as marcações dos redexes de um termo transformando-os em redexes normais.
Por último, define-se a relação $\twoheadrightarrow_\varphi$ em Λ' que reduza todos os redexes marcados de um λ'-termo:

$$\frac{}{x \twoheadrightarrow_\varphi x}\ (I) \qquad \frac{M \twoheadrightarrow_\varphi M'}{(\lambda x.M) \twoheadrightarrow_\varphi (\lambda x.M')}\ (II)$$

$$\frac{M \twoheadrightarrow_\varphi M' \quad N \twoheadrightarrow_\varphi N'}{MN \twoheadrightarrow_\varphi M'N'}\ (III) \qquad \frac{M \twoheadrightarrow_\varphi M' \quad N \twoheadrightarrow_\varphi N'}{(\lambda' x.M)N \twoheadrightarrow_\varphi [N'/x]M'}\ (IV)$$

Munidos das definições necessárias, prosseguimos com a demonstração.

Lema 2.5. Sejam $M, M' \in \Lambda'$ tais que $M \twoheadrightarrow_\varphi M'$. Então $M' \in \Lambda$ e $|M| \twoheadrightarrow_1 M'$.

Dem. Basta uma indução em $M \twoheadrightarrow_\varphi M'$, notando que M' não possui redexes marcados. □

Lema 2.6. Sejam $M, M' \in \Lambda$ tais que $M \rhd^{\mathcal{R}}_{mcd} M'$. Então $\theta(M, \mathcal{R}) \twoheadrightarrow_\varphi M'$.

Dem. Indução estrutural simples em M. □

Portanto, sejam $M, M' \in \Lambda$, tais que $M \rhd^{\mathcal{R}}_{mcd} M'$ segue que:

$$\begin{array}{rcl} M \rhd^{\mathcal{R}}_{mcd} M' & \stackrel{(2.6)}{\Longrightarrow} & \theta(M, \mathcal{R}) \twoheadrightarrow_\varphi M' \\ & \stackrel{(2.5)}{\Longrightarrow} & |\theta(M, \mathcal{R})| \twoheadrightarrow_1 M' \\ & \Longrightarrow & M \twoheadrightarrow_1 M' \end{array}$$

(O último passo é imediato uma vez que θ põe marcas e $|\cdot|$ retira marcas.)
Logo, se $M \rhd_{mcd} M'$ então $M \twoheadrightarrow_1 M'$ para quaisquer $M, M' \in \Lambda$.

De **1.** e **2.** vem que $\rhd_{mcd} = \twoheadrightarrow_1$. □

2.3 Call-by-Name

Uma estratégia de redução é um conjunto determinista de regras utilizadas para determinar uma ordem pela qual os redexes são reduzidos num lambda-termo, característica que a β-redução por si só não possui.

Antes de continuarmos, é preciso introduzir uma notação. Chamamos de *weak head normal forms (whnf)* à termos da forma $\lambda x.M$ ou $xM_1 \cdots M_n$, com $M, M_1, \cdots, M_n$ lambda-termos quaisquer. Dizemos que W é a whfn de M se $M \twoheadrightarrow_\beta W$

Definição 2.9 ($\Downarrow$)**.** Sejam $M, N, M' \in \Lambda$ e x uma variável. Vamos definir por indução em M a seguinte relação binária em Λ.

$$\frac{}{x \Downarrow x} \qquad \frac{}{(\lambda x.M) \Downarrow (\lambda x.M)}$$

$$\frac{M \Downarrow xT_1 \cdots T_n}{MN \Downarrow xT_1 \cdots T_n N} \qquad \frac{M \Downarrow (\lambda x.M') \quad [N/x]M' \Downarrow U}{MN \Downarrow U}$$

Definição 2.10 ($\rightarrow_{wh\beta}$)**.** Sejam $M, N, M' \in \Lambda$ e x uma variável. Vamos definir por indução em M a seguinte relação binária em Λ.

$$\frac{}{(\lambda x.M)N \rightarrow_{wh\beta} [N/x]M} \qquad \frac{M \rightarrow_{wh\beta} M'}{MN \rightarrow_{wh\beta} M'N}$$

Lema 2.7. Sejam M, N lambda-termos tais que $M \Downarrow N$. Então M possui whnf e N é uma whnf de M.

Dem. Indução em $M \Downarrow N$ □

Lema 2.8. Seja W um lambda-termo na whnf, então $W \Downarrow W$.

Lema 2.9. Sejam M, N lambda-termos. Se M não possui whnf então MN também não possui.

Dem. Assumindo que $MN \Downarrow W$, então, as duas regras que podemos aplicar encontram, recursivamente, a whnf de M. Mas M não possui whnf. □

Portanto, já possuímos conhecimento de que a relação $\Downarrow$ encontra a whnf de um termo, se ele à possuir. Podemos a seguir provar os mesmos resultados

para $\to_{wh\beta}$ ou tentar estabelecer uma equivalência. A segunda opção é claramente mais poderosa. Note que o teorema apresentado a seguir também é válido para a estratégia *call-by-value*, com apenas pequenas modificações na demosntrações.

Lema 2.10. Se $M \Downarrow N$ então $M \twoheadrightarrow_{wh\beta} N$.

Dem. Indução em $\Downarrow$. □

Lema 2.11. Se $M \to_{wh\beta} M'$ e $M' \Downarrow N$ então $M \Downarrow N$.

Dem. A demonstração segue por indução em $\to_{wh\beta}$.

Caso $\overline{(\lambda x.P)Q \to_{wh\beta} [Q/x]P}$ e $[Q/x]P \Downarrow R$,

queremos provar que $(\lambda x.P)Q \Downarrow R$. Mas isso é imediato.

Caso $\dfrac{P \to_{wh\beta} P'}{PQ \to_{wh\beta} P'Q}$ e $P'Q \Downarrow R$,

queremos provar que $PQ \Downarrow R$.

Por hipótese de indução temos que $P' \Downarrow R'$ implica $P \Downarrow R'$. Vamos provar dois subcasos de $P'Q \Downarrow R$:

Caso $P' \Downarrow xT_1 \cdots T_n$, da hipótese de indução e aplicação de uma regra segue que $PQ \Downarrow xT_1 \cdots T_nQ$.

Caso $P' \Downarrow (\lambda x.T)$ e $[Q/x]T \Downarrow R$, por hipótese de indução temos que $P \Downarrow (\lambda x.T)$, logo:

$$\frac{P \Downarrow (\lambda x.T) \quad [Q/x]T \Downarrow R}{PQ \Downarrow R}$$

□

Lema 2.12. Se $M \twoheadrightarrow_{wh\beta} N$ e N for uma whnf, então $M \Downarrow N$.

Dem. A demonstração segue por indução sobre o comprimento de $\twoheadrightarrow_{wh\beta}$, com os lemas 2.7 e 2.11. □

Teorema 2.4. *Sejam M, N lambda-termos, então $M \twoheadrightarrow_{wh\beta} N$ e N whnf se e só se $M \Downarrow N$.*

Dem. Com os lemas 2.10, 2.11 e 2.12 concluimos a demonstração do teorema proposto.

□

Capítulo 3

O lambda-calculus com tipos simples

Até agora, enquanto falamos do lambda-calculus sem tipos, não nos foram impostas restrições em como combinamos e utilizamos os objectos estudados, ou seja, qualquer termo pode ser aplicado a qualquer abstração. Esse comportamento pode ser indesejado, por exemplo, não faz sentido algum calcular $\int \mathbb{N}\,\mathrm{d}x$, os elementos possuem tipos diferentes, não são compatíveis.

De um certo ponto de vista, as diferentes variações tipadas do lambda-calculus [Bar93] podem ser vistas como refinamentos e especializações do mesmo. Tais variantes possuem um papel fundamental nos sistemas de tipos das linguagens de programação ou, como veremos posteriormente neste capítulo, o lambda-calculus tipado está intrinsecamente conectado à Lógica Matemática pelo *Isomorfismo de Curry-Howard*.

3.1 Tipos Simples

Dentro dos tipos simples existem dois estilos, os tipos implícitos ou explícitos, também conhecidos por *Curry Typing* e *Church Typing* respectivamente. Na variação explícita, ou de Church [Chu40], cada lambda-termo possui o seu tipo anotado, enquanto na variação implícita, ou de Curry [HCS72], o mesmo lambda-termo pode possuir vários tipos diferentes depententes de um contexto.

Neste capítulo vamos apresentar o lambda-calculus com tipos simples *a lá Curry*, alguns resultados estudados acerca do *subject reduction* e vamos apresentar a conexão entre os lambda-termos e a Lógica Matemática. Por motivos de simplificação vamos utilizar a convenção de variáveis de [Bar84],

onde convencionamos que α-converssões serão aplicadas sempre que preciso para garantir que todas as variáveis possuam nomes diferentes.

Definição 3.1 (Tipo)**.** Seja $\mathcal{C}_{\mathcal{T}} = \{\sigma, \sigma', \sigma'', \ldots\}$ um conjunto de tipos, chamados de *tipos atómicos*. Define-se então o conjunto $\mathbb{T}$ dos tipos simples indutivamente por:

(i) $\mathcal{C}_{\mathcal{T}} \subseteq \mathbb{T}$,

(ii) Sejam $\sigma, \tau \in \mathbb{T}$, então $(\sigma \to \tau) \in \mathbb{T}$.

Definição 3.2 (Contexto)**.** Define-se por contexto um conjunto $\Gamma \subseteq \mathcal{V} \times \mathbb{T}$. Podemos ver um contexto como um conjunto que atribui tipos à variáveis. Um elemento de Γ é da forma $(x : \sigma)$.

Definição 3.3 (Derivação)**.** Define-se indutivamente o conjunto de todas as derivações por:

(i) Sejam x uma variável, σ um tipo e Γ um contexto. Se $(x : \sigma) \in \Gamma$ então

$$\frac{}{\Gamma \vdash x : \sigma}\ (Ax) \text{ é uma derivação.}$$

(ii) Seja $\mathcal{D}$ uma derivação com conclusão $\Gamma, x : \tau \vdash M : \sigma$. Então

$$\frac{\begin{array}{c}\vdots\ \mathcal{D} \\ \Gamma, x : \tau \vdash M : \sigma\end{array}}{\Gamma \vdash (\lambda x.M) : (\tau \to \sigma)}\ (I \to) \text{ é uma derivação.}$$

(iii) Sejam $\mathcal{D}$ e $\mathcal{D}'$ derivações com conclusões $\Gamma \vdash M : (\tau \to \sigma)$ e $\Gamma \vdash N : \tau$ respectivamente. Então

$$\frac{\begin{array}{cc}\vdots\ \mathcal{D} & \vdots\ \mathcal{D}' \\ \Gamma \vdash M : (\tau \to \sigma) & \Gamma \vdash N : \tau\end{array}}{\Gamma \vdash MN : \sigma}\ (E \to) \text{ é uma derivação}$$

Diz-se que $\mathcal{D}$ é derivação de $\Gamma \vdash M : \sigma$ se esse sequente é a conclusão de $\mathcal{D}$.

Definição 3.4 (Derivabilidade)**.** Sejam Γ um contexto, $M \in \Lambda$ e σ um tipo. Dizemos que o sequente $\Gamma \vdash M : \sigma$ é derivável se existe uma derivação desse sequente.

O lema a seguir permite-nos olhar para a derivação de um sequente de forma inversa, partindo das conclusões.

Lema 3.1 (Lema da Geração)**.** Sejam $M, N \in \Lambda$ e Γ um contexto, então:

1. Se $\Gamma \vdash x : \sigma$ é derivável então $(x : \sigma) \in \Gamma$.
2. Se $\Gamma \vdash MN : \sigma$ é derivável então existe um $\tau \in \mathbb{T}$ tal que $\Gamma \vdash M : (\tau \to \sigma)$ e $\Gamma \vdash N : \tau$ são deriváveis.
3. Se $\Gamma \vdash (\lambda x.M) : (\tau \to \sigma)$ é derivável então $\Gamma, x : \tau \vdash M : \sigma$ é derivável.

Dem. Indução estrutural nas derivações de cada um dos casos. □

Lema 3.2 (Lema da Substituição)**.** Sejam M e N λ-termos e Γ um contexto tais que $\Gamma, x : \sigma \vdash M : \tau$ e $\Gamma \vdash N : \sigma$ são deriváveis.
Então

$$\Gamma \vdash [N/x]M : \tau \text{ é derivável.}$$

Ou seja, podemos dizer que a regra

$$\frac{\Gamma, x : \sigma \vdash M : \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash [N/x]M : \tau}\ (Subs)$$

É admissível.

Dem. Vamos fixar a derivação $\mathcal{D}'$ de $\Gamma \vdash N : \sigma$. Queremos provar que para qualquer derivação $\mathcal{D}$, se $\mathcal{D}$ é derivação de $\Gamma, x : \sigma \vdash M : \tau$ então existe uma derivação $\mathcal{D}''$ tal que $\mathcal{D}''$ é derivação de $\Gamma \vdash [N/x]M : \tau$.

A demonstração segue por indução na derivação $\mathcal{D}$.

1. Caso $\mathcal{D}$ seja a derivação:

$$\frac{}{\Gamma, x : \sigma \vdash x : \sigma}\ (Ax)$$

Queremos provar que existe $\mathcal{D}''$ derivação de $\Gamma \vdash [N/x]x : \sigma$.
Mas isso é evidente uma vez que $[N/x]x = N$ e por hipótese $\mathcal{D}'$ é derivação de $\Gamma \vdash N : \sigma$.
Segue que $\mathcal{D}'' = \mathcal{D}'$.

2. Caso $\mathcal{D}$ seja derivação:

$$\frac{}{\Gamma, x : \sigma \vdash y : \tau}\ (Ax)$$

Queremos provar que existe $\mathcal{D}''$ derivação de $\Gamma \vdash [N/x]y : \sigma$.
A prova é imediata uma vez que $[N/x]y = y$.

3. Caso $\mathcal{D}$ seja a derivação:

$$\dfrac{\begin{array}{c}\vdots\\ \Delta, y:\phi, x:\sigma \vdash M:\tau\end{array}}{\Delta, x:\sigma \vdash (\lambda y.M):(\phi\to\tau)}\ (I\to)$$

Queremos provar que $\Delta \vdash (\lambda y[N/x]M):(\phi\to\tau)$ é derivável.
Ora, $[N/x](\lambda y.M) = (\lambda y.[N/x]M)$ e por hipótese de indução sabemos que existe uma derivação $\mathcal{E}$ de $\Delta, y:\phi \vdash [N/x]M:\tau$.
Logo:

$$\dfrac{\begin{array}{c}\vdots\ \mathcal{E}\\ \Delta, y:\phi \vdash [N/x]M:\tau\end{array}}{\Delta \vdash (\lambda y.[N/x]M):(\phi\to\tau)}$$

é derivação de $\Delta \vdash (\lambda y.[N/x]M):(\phi\to\tau)$.

4. Caso $\mathcal{D}$ seja a derivação:

$$\dfrac{\begin{array}{c}\vdots\\ \Gamma, x:\sigma \vdash M_1:(\phi\to\tau)\end{array}\qquad\begin{array}{c}\vdots\\ \Gamma, x:\sigma \vdash M_2:\phi\end{array}}{\Gamma, x:\sigma \vdash (M_1M_2):\tau}\ (E\to)$$

Tal como no caso anterior o resultado segue directamente das hipóteses de indução notando que $[N/x](M_1M_2) = [N/x]M_1[N/x]M_2$.

□

Teorema 3.1. *Sejam M e M' λ-termos e Γ um contexto tais que $\Gamma \vdash M:\tau$ é derivável e $M \to_\beta M'$. Então $\Gamma \vdash M':\tau$ é derivável.*

Dem. A demonstração é feita por indução em $M \to_\beta M'$. Os casos indutivos são evidentes a partir das respectivas hipóteses de indução. Vejamos o caso de base:
Caso $M \to_\beta M' \equiv (\lambda x.M)N \to_\beta [N/x]M$,
Sabemos que $\Gamma \vdash (\lambda x.M)N:\tau$ é derivável, portanto, pelo lema da geração, podemos dizer que existe um σ tal que:

$$\Gamma \vdash (\lambda x.M):(\sigma\to\tau) \text{ e } \Gamma \vdash N:\sigma$$

são deriváveis.
Aplicando o mesmo lema outra vez, segue que:

$$\Gamma, x:\sigma \vdash M:\tau \text{ e } \Gamma \vdash N:\sigma$$

são deriváveis.
Pelo lema 3.2 concluímos que:

$$\Gamma \vdash [N/x]M:\tau$$

é derivável. □

Corolário 3.1 (Subject Reduction)**.** *Sejam M e M' λ-termos e Γ um contexto tais que $\Gamma \vdash M : \tau$ é derivável e $M \twoheadrightarrow_\beta M'$. Então $\Gamma \vdash M' : \tau$ é derivável.*

Teorema 3.2. *A regra estrutural de contração é admissível no sistema de derivação de tipos introduzido na definição 3.3:*

$$\frac{\Gamma, x : \sigma, y : \sigma \vdash M : \tau}{\Gamma, x : \sigma \vdash [x/y]M : \tau}\ (C)$$

Dem. Vamos considerar a seguinte derivação $\mathcal{D}$:

$$\frac{\dfrac{\Gamma, x : \sigma, y : \sigma \vdash M : \tau}{\Gamma, x : \sigma \vdash (\lambda y.M) : (\sigma \to \tau)}\ (I \to) \quad \dfrac{}{\Gamma, x : \sigma \vdash x : \sigma}\ (Ax)}{\Gamma, x : \sigma \vdash (\lambda y.M)x : \tau}\ (E \to)$$

Ora, pela derivação $\mathcal{D}$ de $\Gamma, x : \sigma \vdash (\lambda y.M)x : \tau$ e uma vez que $(\lambda y.M)x \to_\beta [x/y]M$, pelo corolário 3.1 (Subject Reduction) podemos garantir a existência de uma deriavação $\mathcal{D}'$ de $\Gamma, x : \sigma \vdash [x/y]M : \tau$. Segue que a regra (c) é admissível. □

Teorema 3.3. *A regra estrutural de enfraquecimento é admissível no sistema de tipos introduzido na definição 3.3:*

$$\frac{\Gamma \vdash M : \tau}{\Gamma' \vdash M : \tau}\ (W),\ \Gamma \subseteq \Gamma'$$

Dem. Vamos começar por provar um lema auxiliar:

Lema 3.3. A seguinte regra é admissível:

$$\frac{\Gamma \vdash M : \tau}{\Gamma, x : \sigma \vdash M : \tau}\ (w),\ x \notin \operatorname{dom} \Gamma$$

Dem. A prova é feita por indução na derivação $\mathcal{D}$ de $\Gamma, x : \sigma \vdash M : \tau$. O único caso que não segue directamente das hipóteses de indução é:
Caso $\mathcal{D}$ seja

$$\frac{\Delta, y : \phi \vdash M : \tau}{\Delta, x : \sigma \vdash (\lambda y.M) : (\phi \to \tau)}\ ,\ \Gamma = \Delta \cup \{(y : \phi)\}$$

Por hipótese de indução sabemos que $\Delta, y : \phi, x : \sigma \vdash M : \tau$ é derivável, portanto, uma vez que x não ocorre livre em M (pois $x \notin \operatorname{dom} \Gamma$) e pela regra $(I \to)$ podemos afirmar que $\Delta, x : \sigma \vdash (\lambda y.M) : (\pi \to \tau)$ é derivável. □

É evidente que a regra (W) pode ser obtida com sucessivas aplicações de (w), portanto é admissível. □

3.2 Isomorfismo de Curry-Howard

O Isomorfismo de Curry-Howard estabelece a ponte entre duas famílias de formalismos aparentemente bastante diferentes. Por um lado temos os sistemas de prova e por outro os modelos de computação, que são estruturalmente o mesmo tipo de objectos.

Sejam M um termo e Γ um contexto tais que $\Gamma \vdash M : \sigma$ é derivável, podemos olhar para σ como uma fórmula proposicional sobre a lógica minimal e M como uma demonstração para essa fórmula. Vejamos a semelhança entre a dedução natural e as regras da definição 3.3: ($\tau \supset \sigma$ denota τ implica σ).

Dedução Natural	Derivação de Tipos
σ	$\dfrac{}{\Gamma, x : \sigma \vdash x : \sigma}\ (Ax)$
$\dfrac{\begin{array}{c}[\tau]\\ \vdots \\ \sigma\end{array}}{\tau \supset \sigma}\ (I \supset)$	$\dfrac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash (\lambda x.M) : (\tau \to \sigma)}\ (I \to)$
$\dfrac{\tau \supset \sigma \quad \tau}{\sigma}\ (E \supset)$	$\dfrac{\Gamma \vdash M : (\tau \to \sigma) \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \sigma}\ (E \to)$

Note que as regras de derivação de tipos são aplicadas de acordo com a forma do termo em questão. Podemos inclusive reescrever as regras de derivação para os tipos no formato da dedução natural. Para tal, vamos assumir que possuímos um conjunto de proposições verdadeiras Γ:

(i) Fórmula atómica:

$$x : \sigma$$

(ii) Abstracção:

$$\frac{\begin{array}{c}[x : \tau]\\ \vdots \\ M : \sigma\end{array}}{(\lambda x.M) : (\tau \to \sigma)}\ (I \to)$$

(iii) Aplicação:

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

Se analizarmos a demonstração do lema 3.2 com algum cuidado, podemos ver que ela possui um conteúdo computacional bem definido, devido à sua

natureza construtiva. Tal conteúdo computacional define uma operação de substituição sobre derivações.

Podemos ver a operação de substiuição como uma operação que concatena uma derivação $\mathcal{D}'$ nos axiomas (regras (Ax)) de uma derivação $\mathcal{D}$ sempre que necessário (caso 1 da demonstração do lema 3.2) ou descarta $\mathcal{D}'$ (caso 2). Note que os casos indutivos (casos 3 e 4) empurram a substituição em direção aos axiomas de uma derivação.

Definição 3.5. Sejam $\mathcal{D}$ uma derivação de um sequente $\Gamma' \vdash M : \sigma$, com $\Gamma' = \Gamma, x : \tau$ e $\mathcal{D}'$ uma derivação de $\Gamma \vdash N : \tau$. A operação de substituição de $\mathcal{D}'$ por x em $\mathcal{D}$, notada por $[\mathcal{D}'/x]\mathcal{D}$, que produz a derivação de $\Gamma \vdash [N/x]M : \sigma$ é definida recursivamente por:

$$[\mathcal{D}'/x](\overline{\Gamma' \vdash x : \tau}) = \mathcal{D}'$$

$$[\mathcal{D}'/x](\overline{\Gamma' \vdash y : \sigma}) = \overline{\Gamma' \vdash y : \sigma}$$

$$[\mathcal{D}'/x]\left(\dfrac{\begin{array}{c}\vdots\\ \Delta, y : \sigma \vdash M : \tau\end{array}}{\Delta \vdash (\lambda y.M) : (\sigma \to \tau)}\right) = \dfrac{[\mathcal{D}'/x]\left(\begin{array}{c}\vdots\\ \Delta, y : \sigma \vdash M : \tau\end{array}\right)}{\Delta \vdash [N/x](\lambda y.M) : (\sigma \to \tau)}$$

$$[\mathcal{D}'/x]\left(\dfrac{\begin{array}{c}\vdots\\ \Gamma' \vdash M : (\sigma \to \tau)\end{array} \quad \begin{array}{c}\vdots\\ \Gamma' \vdash N : \sigma\end{array}}{\Gamma' \vdash MN : \tau}\right) = \dfrac{[\mathcal{D}'/x]\left(\begin{array}{c}\vdots\\ \Gamma' \vdash M : (\sigma \to \tau)\end{array}\right) \quad [\mathcal{D}'/x]\left(\begin{array}{c}\vdots\\ \Gamma' \vdash N : \sigma\end{array}\right)}{\Gamma \vdash [N/x](MN) : \tau}$$

Portanto, sejam $\mathcal{D}$ e $\mathcal{D}'$ respectivamente as derivações dos sequentes $\Gamma, x : \sigma \vdash M : \tau$ e $\Gamma \vdash N : \sigma$. O lema 3.2 nos garante que nessas condições existe uma derivação de $\Gamma \vdash [N/x]M : \tau$. Agora, além de termos uma garantia da sua existência, também à conhecemos. Tal derivação é precisamente $[\mathcal{D}'/x]\mathcal{D}$.

Observarndo agora a demonstração do caso indutivo do teorema 3.1 notamos que existe uma outra operação bem definida sobre derivações de tipos. Estamos a transformar derivações da seguinte forma:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\ \mathcal{D}\\ \Gamma, x : \sigma \vdash M : \tau\end{array}}{\Gamma \vdash (\lambda x.M) : (\sigma \to \tau)} \quad \begin{array}{c}\vdots\ \mathcal{D}'\\ \Gamma \vdash N : \sigma\end{array}}{\Gamma \vdash (\lambda x.M)N : \tau}$$

Na derivação $[\mathcal{D}'/x]\mathcal{D}$, ou seja, estamos a copiar e concatenar $\mathcal{D}'$ em todos os axiomas de $\mathcal{D}$ com a forma $\Gamma \vdash x : \sigma$, resultando em algo como:

$$\dfrac{\begin{array}{c}\vdots\ \mathcal{D}'\\ \Gamma \vdash N : \sigma\\ \vdots\ \mathcal{D}\end{array}}{\Gamma \vdash [N/x]M : \tau}$$

É notável que a forma dessa manipulação assemelha-se muito à uma contração de uma fórmula maximal na dedução natural (neste caso $(\sigma \to \tau)$). Essa semelhança não é uma surpresa. No contexto do isomorfismo de Curry-Howard uma β-redução equivale à uma contração de uma fórmula maximal. Ao olharmos para o teorema 3.1 sobre o mesmo isomorfismo, o teorema diz que uma contração não altera a conclusão de uma derivação em dedução natural. Em outras palavras, a derivação de um termo na forma normal corresponde à uma derivação em dedução natural normalizada.

3.3 Cálculo de Sequentes

Ja mostramos a relação entre o lambda-calculus e a dedução natural. De forma semelhante podemos estabelecer a relação entre o lambda-calculus e o calculo de sequentes.

Note que haverá uma pequena mudança de notação neste capítulo, de forma a facilitar a leitura das derivações, os tipos (e consequentemente fórmulas proposicionais) de termos serão notados por $A, B, C, \ldots$ ao invés de $\sigma, \tau, \phi, \ldots$ como nos capítulos anteriores.

Vamos utilizar o sistema **G2i** + *cut* minimal, apresentado com mais detalhes em [TS00], de seguida vamos apresentar uma variação unilateral do cálculo inicial, que será utilizada em capítulos posteriores.

Definição 3.6 (Sequente)**.** Sejam Γ um multiconjunto de fórmulas e A uma fórmula, define-se um sequente do **G2i** + *cut* pela expressão $\Gamma \Rightarrow A$.
A Γ dá-se o nome de contexto.

Dado que estamos a trabalhar apenas com a lógica minimal intuicionista, as regras que utilizaremos são:

$$\frac{}{\Gamma, A \Rightarrow A}\ (Ax) \qquad \frac{\Gamma \Rightarrow A \quad A, \Gamma' \Rightarrow B}{\Gamma, \Gamma' \Rightarrow B}\ (Cut)$$

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

$$\frac{A, A, \Gamma \Rightarrow B}{A, \Gamma \Rightarrow B}\ (LC) \qquad \frac{\Gamma, A \Rightarrow \Delta}{\Gamma \Rightarrow \neg A, \Delta}\ (R\neg)$$

Lema 3.1 (Enfraquecimento)**.** Seja $\mathcal{D}$ uma derivação do sequente $\Gamma \Rightarrow A$, então existe uma derivação $\mathcal{D}'$ do sequente $\Gamma, B \Rightarrow A$. Ou seja, a seguinte regra é admissível ao sistema:

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

Dem. A demonstração segue por indução em $\mathcal{D}$, adicionando B no lado esquerdo dos sequentes, até chegarmos aos axiomas. □

Nota 3.1. Pela definição do sistema, a regra $(L \supset)$ necessita de contextos iguais em ambas premissas, porém com o lema 3.1 podemos passar a utilizar uma verssão com contextos diferentes da regra $(L \supset)$. Isto é, a seguinte regra também é admissível:

$$\frac{\Gamma \Rightarrow A \quad B, \Gamma' \Rightarrow C}{A \supset B, \Gamma, \Gamma' \Rightarrow C}\ (L2 \supset)$$

Note que as duas verssões são equivalentes, isto é, $(L2 \supset)$ simula o comportamento de $(L \supset)$ e vice versa:

$$\dfrac{\dfrac{\Gamma \Rightarrow A \quad B, \Gamma \Rightarrow C}{A \supset B, \Gamma, \Gamma \Rightarrow C}\ (L2 \supset)}{\overline{\overline{A \supset B, \Gamma, \Rightarrow C}}\ (LC)} \qquad \dfrac{\dfrac{\overline{\overline{\Gamma \Rightarrow A}}\ (LW) \quad \overline{\overline{\Gamma', B \Rightarrow C}}\ (LW)}{\Gamma, \Gamma' \Rightarrow A}}{A \supset B, \Gamma, \Gamma' \Rightarrow C}\ (L \supset)$$

A partir de agora, sempre que a regra $(L \supset)$ for utilizada, estaremos na verdade a utilizar a regra $(L2 \supset)$ a não ser que esteja explicitamente comentado.

Ao longo deste capítulo, caso não haja ambiguidade, vamos nos referir à demonstrações pelos seus respectivos lambda-termos.

3.3.1 Interpretação de Gentzen

A tradução de uma derivação em dedução natural para uma derivação em **G2i** + *cut* , numa primeira vista, não é complexa porém apresenta alguns problemas que serão poseriormente discutidos.

Seja $\Gamma = \{x_1 : A_1, \cdots, x_n : A_n\}$ um contexto, vamos notar por $\Gamma^{\mathcal{G}}$ o multi-conjunto que contém apenas os tipos associados às variáveis: $\{\{A_1, \cdots, A_n\}\}$.

Teorema 3.4 (Interpretação de Gentzen)**.** *Sejam M um lambda-termo e $\mathcal{D}$ uma derivação de $\Gamma \vdash M : A$, então, existe uma derivação $\mathcal{D}^{\mathcal{G}}$ de $\Gamma^{\mathcal{G}} \Rightarrow A$ em **G2i** + cut .*

Dem. A demonstração segue por indução em $\mathcal{D}$, e define a função de tradução $(\cdot)^{\mathcal{G}}$:

	Casos possíveis para $\mathcal{D}$	Correspondente $\mathcal{D}^{\mathcal{G}}$
(i)	$\dfrac{}{\Gamma, x : A \vdash x : A}\ (Ax)$	$\dfrac{}{\Gamma^{\mathcal{G}}, A \Rightarrow A}\ (Ax)$
(ii)	$\dfrac{\begin{array}{c}\vdots\ \mathcal{D}_0 \\ \Gamma, x : A \vdash M : B\end{array}}{\Gamma \vdash (\lambda x.M) : (A \to B)}\ (I \to)$	$\dfrac{\begin{array}{c}\mathcal{D}_0^{\mathcal{G}} \\ \Gamma^{\mathcal{G}}, A \Rightarrow B\end{array}}{\Gamma^{\mathcal{G}} \Rightarrow A \supset B}\ (R \supset)$
(iii)	$\dfrac{\begin{array}{cc}\vdots\ \mathcal{D}_0 & \vdots\ \mathcal{D}_1 \\ \Gamma \vdash M : (A \to B) & \Gamma \vdash N : A\end{array}}{\Gamma \vdash MN : B}\ (E \to)$	$\dfrac{\begin{array}{cc}\mathcal{D}_0^{\mathcal{G}} & \dfrac{\begin{array}{cc}\mathcal{D}_1^{\mathcal{G}} & \\ \Gamma^{\mathcal{G}} \Rightarrow A & B \Rightarrow B\end{array}}{A \supset B, \Gamma^{\mathcal{G}} \Rightarrow B}\ (L \supset) \\ \Gamma^{\mathcal{G}} \Rightarrow A \supset B & \end{array}}{\Gamma^{\mathcal{G}} \Rightarrow B}\ (Cut)$

□

Note que a regra correspondente a $(E \to)$ utiliza um corte e a regra de introdução do conectivo em questão à esquerda (Esse comportamento é observado em todas as regras de eliminação da dedução natural). Pela secção 3.2 podemos concluir que a tradução da derivação de $MN : B$ para o calculo de sequente utiliza pelo menos um corte. A forma mais natural de remover tal corte é "empurrá-lo"para cima pelo lado esquerdo da derivação, pois a fórmula do corte é introduzida pelo lado direito.

Se removermos tal corte empurrando-o pelo lado esquerdo, nos três possíveis subcasos: $xN : \sigma$, $(\lambda x.M)N : \sigma$ e $(PQ)N : \sigma$ vamos acabar por encontrar

uma outra traduçao, mais eficiente, da dedução natural no cálculo de sequentes. Tal tradução é conhecida como a tradução de Prawitz[Pra64].

Vamos ilustrar este comportamento com um exemplo. Vejamos todo o processo de tradução desde a derivação do tipo do combinador B até a sua derivação em cálculo de sequentes sem cortes:

$$B = (\lambda xyz.x(yz)) : (A \to B) \to (C \to A) \to C \to B$$

Derivação do tipo de B:

$$\dfrac{\dfrac{\dfrac{}{\Gamma_1, x : (A \to B) \vdash x : (A \to B)}\ (Ax) \quad \dfrac{\dfrac{}{\Gamma_2, y : (C \to A) \vdash y : (C \to A)}\ (Ax) \quad \dfrac{}{\Gamma_3, z : C \vdash z : C}\ (Ax)}{\Gamma_1, x : (A \to B) \vdash yz : A}\ (E \to)}{\Gamma, x : (A \to B), y : (C \to A), z : C \vdash x(yz) : B}\ (E \to)}{\Gamma \vdash (\lambda xyz.x(yz)) : (A \to B) \to (C \to A) \to C \to B}\ (I \to)$$

Tradução de Gentzen:

$$\dfrac{\dfrac{\dfrac{}{(A \supset B) \Rightarrow (A \supset B)}\ (Ax) \quad \dfrac{\dfrac{}{(C \supset A) \Rightarrow (C \supset A)}\ (Ax) \quad \dfrac{\dfrac{}{C \Rightarrow C}\ (Ax) \quad \dfrac{}{A \Rightarrow A}\ (Ax)}{(C \supset A), C \Rightarrow A}\ (L \supset)}{(C \supset A), C \Rightarrow A}\ (Cut_2) \quad \dfrac{}{B \Rightarrow B}\ (Ax)}{(A \supset B), (C \supset A), C \Rightarrow B}\ (L \supset)}{(A \supset B), (C \supset A), C \Rightarrow B}\ (Cut_1)}{\Rightarrow (A \supset B) \supset (C \supset A) \supset C \supset B}\ (R \supset)$$

Eliminação do corte (Cut_2):

$$\dfrac{\dfrac{\dfrac{}{(A \supset B) \Rightarrow (A \supset B)}\ (Ax) \quad \dfrac{\dfrac{\dfrac{}{C \Rightarrow C}\ (Ax) \quad \dfrac{}{A \Rightarrow A}\ (Ax)}{(C \supset A), C \Rightarrow A}\ (L \supset) \quad \dfrac{}{B \Rightarrow B}\ (Ax)}{(C \supset A), (A \supset B), C \Rightarrow B}\ (L \supset)}{(A \supset B), (C \supset A), C \Rightarrow B}\ (Cut_1)}{\Rightarrow (A \supset B) \supset (C \supset A) \supset C \supset B}\ (R \supset)$$

Eliminação do corte (Cut_1):

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

Chegamos assim na derivação que a tradução de Prawitz escolheria para o tipo do combinador B. É importante notar que esta derivação não é única. Existem outras derivações sem cortes de $(A \supset B) \supset (C \supset A) \supset C \supset B$, por

exemplo:

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

Capítulo 4

Redes de Prova

As redes de prova, ou *proof nets*, são um formalismo geométrico para representar provas (deduções) que elimina alguma *burocracia* dessas provas: (A) remove propriedades estruturais irrelevantes de calculos como a dedução natural ou o cálculo de sequentes, e (B) a ordem das regras aplicadas deixa de fazer sentido. As redes de prova foram introduzidas por Jean-Yves Girard em [Gir87], em conjunto com a lógica linear.

A lógica linear surgiu como um refinamento das lógicas clássica e intuicionista. As contrações e enfraquecimentos não podem ser aplicados arbitrariamente.

Vamos assumir algum conhecimento sobre lógica linear, e para isso refere-se o leitor à [Gir95]. Começaremos por apresentar brevemente os fragmentos multiplicativo e exponencial da lógica linear para podermos codificar o lambda-calculus. Continuaremos por mostrar a sua representação gráfica, as redes de prova, que levaram ao surgimento das redes de interacção, que serão estudadas no próximo capítulo.

4.1 Lógica Linear

As Ciências da Computação focam os seus estudos nos aspectos computacionais, isto é, uma função com tipo $A \to B$ nos fornece um algorítmo, uma *receita*, de como tranformar As em Bs. A Lógica, por outro lado, foca-se nos sistemas formais, com lógicas intuicionistas e constructivas passamos a ler $A \to B$ como: *posso dar-te um B, se me forneceres um A*, isto é, se A for válido, a lógica nos garante a existência de B. Com o Isomorfismo de Curry-Howard, secção 3.2, essas duas ideias uniram-se.

A Lógica Linear trouxe uma nova leitura de $A \to B$, lê-se agora: *posso*

dar-te ***um*** *B, se me forneceres* ***tantos*** *As* ***quantos*** *eu preciso.* A noção de cópia está directamente conectada com a lógica, noção essa que é bastante importante para a ideia de computação.

Vamos começar por apresentar o fragmento multiplicativo da lógica linear, as fórmulas são definidas pela seguinte gramática:

$$A, B ::= p \mid A \parr B \mid A \otimes B \mid ?A \mid !A$$

Onde p representa fórmulas atómicas. O conectivo $\parr$ é chamado de *par*, e, $\otimes$ recebe o nome de *tensor*, esses são os conectivos do fragmento multiplicativo. O fragmento exponencial é composto pelos conectivos unários ?, chamado de *why not* e !, *of course*.

Para cada átomo p existe outro átomo $p^\perp$ tal que $p^{\perp\perp} = p$, extende-se a operação de negação, $(\cdot)^\perp$, para todas as fórmulas recursivamente:

$$\begin{aligned}
(A \parr B)^\perp &= A^\perp \otimes B^\perp \\
(A \otimes B)^\perp &= A^\perp \parr B^\perp \\
(!A)^\perp &= ?A^\perp \\
(?A)^\perp &= !A^\perp
\end{aligned}$$

Definição 4.1 (Tradução de Fórmulas)**.** Seja A uma fórmula sobre a lógica minimal, define-se a correspondente fórmula $A^{\mathcal{L}}$ sobre a lógica linear por:

$$\begin{aligned}
p^{\mathcal{L}} &= p, \text{se } p \text{ for um átomo} \\
(A \supset B)^{\mathcal{L}} &= (A^{\mathcal{L}})^\perp \parr (B^{\mathcal{L}})
\end{aligned}$$

Assim como as outras lógicas, a lógica linear também possui um cálculo. A seguir será apresentado o cálculo de sequentes unilateral para os fragmentos multiplicativo e exponencial.

$$\frac{}{\vdash A, A^\perp}\ (Ax) \qquad \frac{\vdash \Gamma, A \quad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta}\ (Cut)$$

$$\frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\ (\parr) \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\ (\otimes)$$

$$\frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\ (!) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\ (D)$$

$$\frac{\vdash \Gamma}{\vdash \Gamma, ?A}\ (W) \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\ (C)$$

Note que o cálculo de sequentes da lógica linear é um cálculo unilateral, pode-se facilmente transformá-lo num cálculo padrão aplicando as leis de De Morgan, uma vez que tais leis também são válidas para MELL. Vamos, a seguir, apresentar a tradução das derivações em **G2i** + *cut* para MELL:

Teorema 4.1. *Seja $\mathcal{D}$ a derivação de $\Gamma \Rightarrow A$ em **G2i**+cut , uma imagem da tradução $(\cdot)^{\mathcal{G}}$. Então existe uma derivação $\mathcal{D}^{\mathcal{L}}$ de $\vdash ?(\Gamma^{\mathcal{L}})^{\perp}, A^{\mathcal{L}}$ em MELL.*

Dem. A demonstração segue por indução em $\mathcal{D}$:

$$(i) \qquad \frac{}{\Gamma, A \Rightarrow A}\ (Ax) \qquad\qquad \frac{\dfrac{\dfrac{}{\vdash A^{\perp}, A}\ (Ax)}{\vdash ?A^{\perp}, A}\ (D)}{\vdash ?(\Gamma^{\mathcal{L}})^{\perp}, ?A^{\perp}, A}\ (W)$$

$$(ii) \qquad \frac{\overset{\mathcal{D}_0}{\Gamma, A \Rightarrow B}}{\Gamma \Rightarrow A \supset B}\ (R\supset) \qquad\qquad \frac{\overset{\mathcal{D}_0^{\mathcal{L}}}{\vdash ?(\Gamma^{\mathcal{L}})^{\perp}, ?A^{\perp}, B}}{\vdash ?(\Gamma^{\mathcal{L}})^{\perp}, ?A^{\perp} \parr B}\ (\parr)$$

$$(iii) \qquad \frac{\overset{\mathcal{D}_0}{\Gamma \Rightarrow A \supset B} \qquad \dfrac{\overset{\mathcal{D}_1}{\Gamma \Rightarrow A} \qquad \dfrac{}{B \Rightarrow B}\ (Ax)}{A \supset B, \Gamma \Rightarrow B}\ (L\supset)}{\Gamma \Rightarrow B}\ (Cut) \qquad\qquad \frac{\dfrac{\overset{\mathcal{D}_0^{\mathcal{L}}}{\vdash ?(\Gamma^{\mathcal{L}})^{\perp}, ?A^{\perp} \parr B} \qquad \dfrac{\dfrac{\overset{\mathcal{D}_1^{\mathcal{L}}}{\vdash ?(\Gamma^{\mathcal{L}})^{\perp}, A}}{\vdash ?(\Gamma^{\mathcal{L}})^{\perp}, !A}\ (P) \qquad \dfrac{}{\vdash B, B^{\perp}}\ (Ax)}{\vdash !A \otimes B^{\perp}, ?(\Gamma^{\mathcal{L}})^{\perp}, B}\ (\otimes)}{\vdash ?(\Gamma^{\mathcal{L}})^{\perp}, ?(\Gamma^{\mathcal{L}})^{\perp}, B}\ (Cut)}{\vdash ?(\Gamma^{\mathcal{L}})^{\perp}, B}\ (C)$$

□

Nem todas as derivações de **G2i** + *cut* possuem uma derivação correspondente em $MELL$. Apenas um subconjunto de **G2i** + *cut* (fragmento LJT), que contém o contradomínio da função $(\cdot)^{\mathcal{G}}$, possui tradução. Mais detalhes sobre o fragmento LJT em [Her95].

Teorema 4.2 (Interpretação de lambda-termos)**.** *Sejam M um lambda-termo e $\mathcal{D}$ uma derivação de $\Gamma \vdash M : A$, então, existe uma derivação $\mathcal{D}'$ de $\vdash ?\Gamma^{\perp}, A^{\mathcal{L}}$ sobre MELL.*

Dem. A derivação $\mathcal{D}'$ é dada por $(\mathcal{D}^{\mathcal{G}})^{\mathcal{L}}$. □

4.2 Redes de Prova

Para podermos eliminar as burocracias do cálculo de sequentes será preciso sair do domínio dos sequentes e entrar no domínio dos grafos. Cada

ocorrência de uma fórmula será representada por uma aresta, possivelmente anotada com a fórmula correspondente, e cada aplicação de uma regra introduz um vértice, ou *link*, entre as suas fórumlas principais. Note que um *link* pode ter um conjunto vazio de premissas ou concluões, porém tais conjuntos nunca podem ser vazios ao mesmo tempo.

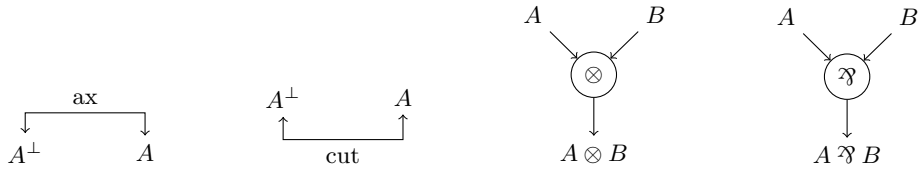


Figura 4.1: *links* do fragmento multiplicativo

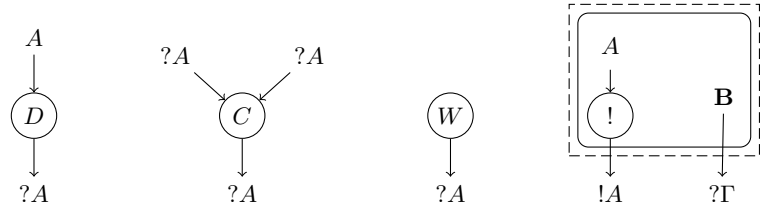


Figura 4.2: *links* do fragmento exponencial

Definição 4.2 (Estrutura de Prova)**.** Uma estrutura de prova é um grafo gerado a partir de axiomas e aplicações dos restantes links. O conjunto de todas as estruturas de prova é denominado por PS.

Definição 4.3 (*Box*)**.** Uma *box* B, ou caixa, é uma estrutura de prova onde todas as conclusões são da forma $?A$, com a excepção de uma, a sua porta principal, que é a conclusão de um !-*link*. As outras conclusões de B são chamadas de portas auxiliares.

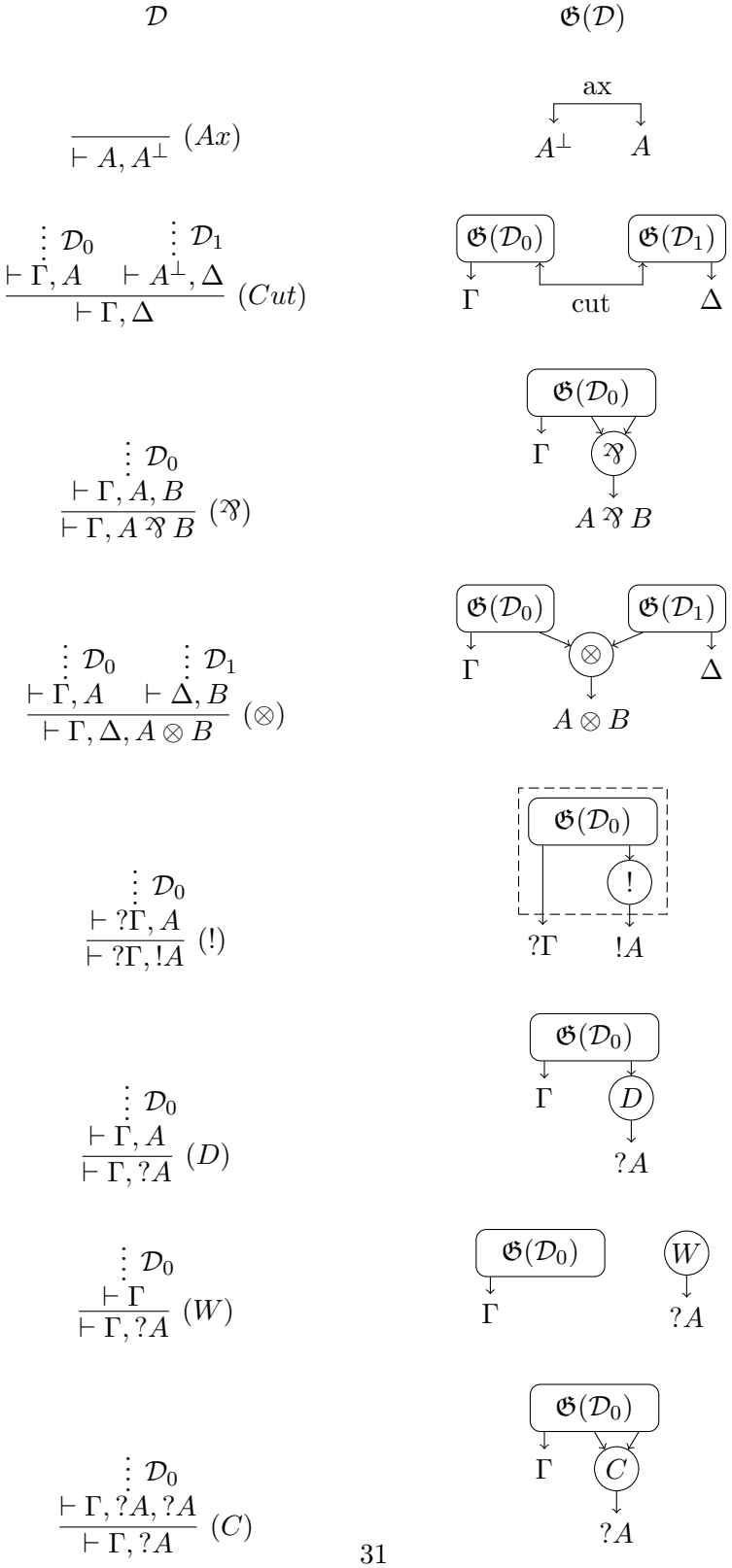
Uma estrutura de prova exponencial consiste numa estrutura de prova e um conjunto $\mathbb{B}$ de *boxes*, tal que para todos os pares $B_1, B_2 \in \mathbb{B}$, então, ou $B_1 \subset B_2 \vee B_2 \subset B_1$, ou $B_1 \cap B_2 = \emptyset$.

A definição de estrutura de prova acima é demasiado vaga. Nem todas as estruturas são interessantes, muitas delas inclusive representam fórmulas mal formadas em MELL.

Teorema 4.3 (Interpretação de MELL em PS)**.** *Seja $\mathcal{D}$ uma derivação de $\vdash \Gamma, A$ em MELL, então, existe uma estrutura de prova $\mathfrak{G}(\mathcal{D})$ com conclusões $\Gamma \cup \{A\}$.*

Dem. A demonstração segue por indução em $\mathcal{D}$. A figura 4.3 mostra a definição da função $\mathfrak{G}$. □

Figura 4.3: Definição de $\mathfrak{G}$



A demonstração do teorema 4.3 fornece uma caracterização simples das estruturas de prova que representam derivações bem formadas. Chamaremos tais estruturas de *Redes de Prova*.

Definição 4.4 (Redes de Prova)**.** Uma rede de prova é uma estrutura de prova que é imagem de $\mathfrak{G}$. O conjunto de todas as redes de prova recebe o nome PN.

4.2.1 Eliminação do Corte

Sejam G e G' duas redes de prova. Escreve-se $G \rightarrow_{cut} G'$ se G' é obtida por um passo de eliminação do corte, $\twoheadrightarrow_{cut}$ denota o fecho transitivo reflexivo.

As figuras 4.4 e 4.5 ilustram as regras de eliminação do corte para os fragmentos multiplicativo e exponencial.

Teorema 4.4 (Eliminação do Corte)**.** *A relação $\rightarrow_{cut}$ é confluente e fortemente normalizável.*

Dem. A demonstração encontra-se em [Gir87]. □

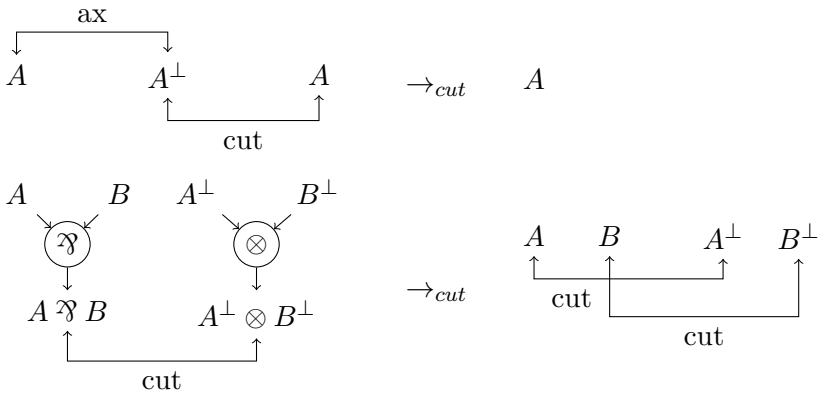


Figura 4.4: Eliminação do corte, MLL

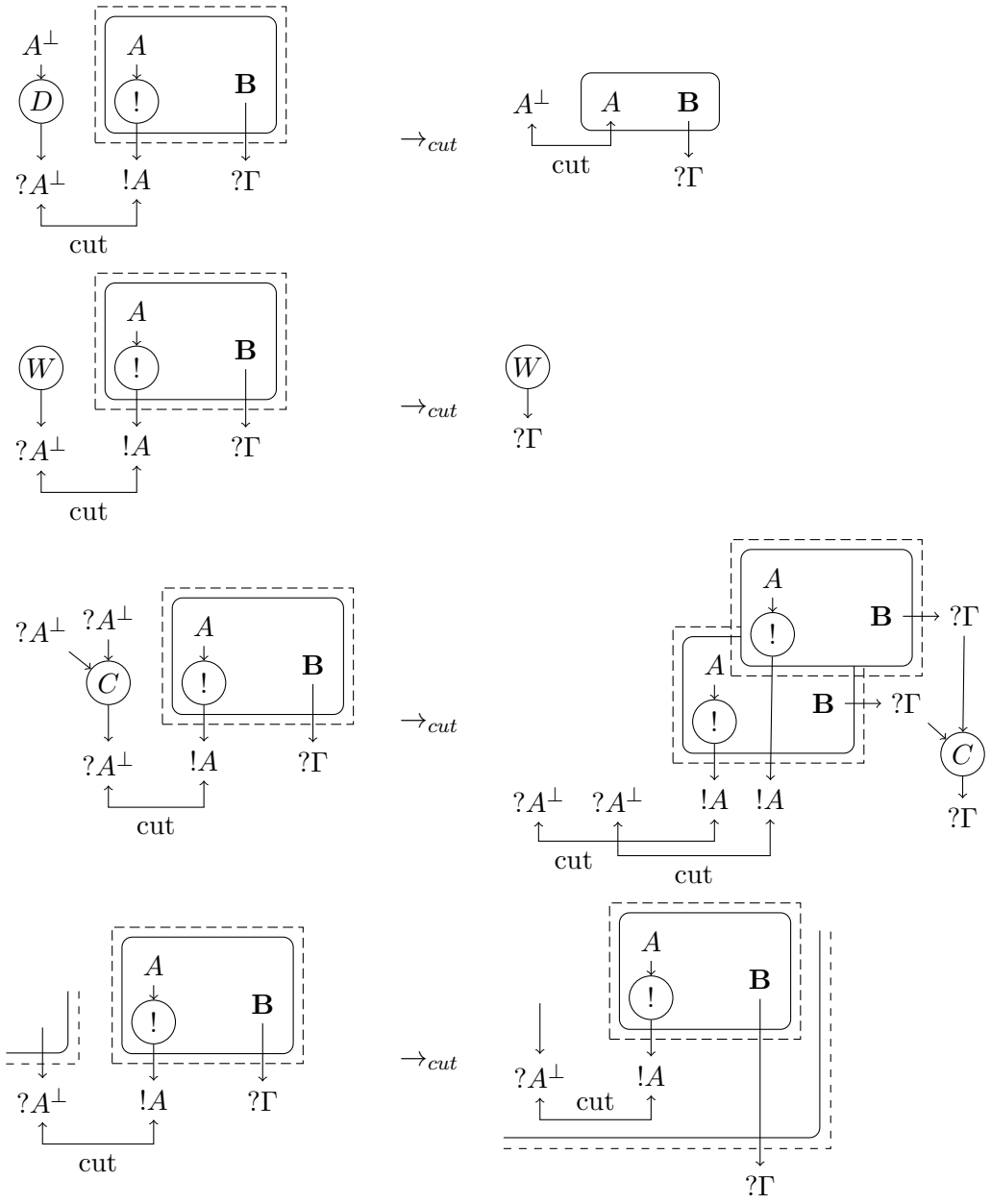


Figura 4.5: Eliminação do corte, MELL

4.2.2 Critério de Correcção

Inicialmente, definimos por Redes de Prova todas as Estruturas de Prova que pertencessem ao contradomínio da função $\mathfrak{G}$. Existe, entretanto, outro caminho para definir as redes de prova correctamente. Pode-se estabelecer um critério geométrico sobre as estruturas de prova, separando assim as bem formadas do resto. Nesta secção vamos apresentar um critério geométrico, para o fragmento multiplicativo.

Definição 4.5 (*Switch*)**.** Um *switch* S de uma estrutura de prova G é uma relação binária, transitiva, simétrica e anti-reflexiva, sobre os vértices de G, definida por $(A, B) \in S$ se uma das seguintes condições se verifica:

i) A, B são as conclusões de um axioma ou premissas de um corte.

ii) A é premissa de um *link* $A \otimes C$ e B é a sua conclusão.

iii) A é premissa de um *link* $A \parr C$ e B a sua conclusão, se $(C, B) \notin S$.

Denota-se por $\mathbb{SW}(G)$ o conjunto de todos os *switches* de uma estrutura de prova G. É relativamente simples vermos que todo $S \in \mathbb{SW}(G)$ define um grafo sobre os vértices de G (figura 4.6).

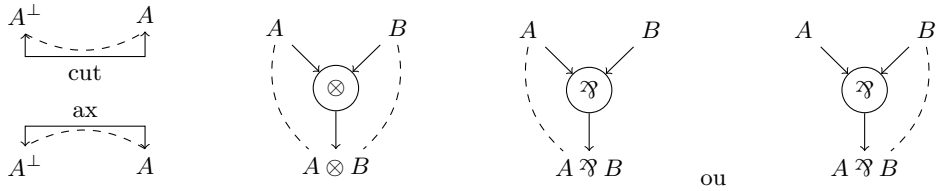


Figura 4.6: Arestas de um *switch* multiplicativo

Definição 4.6 (Critério de Correcção)**.** Uma estrutura de prova G satisfaz o critério de *Danos-Regnier* [DR89] quando todo $S \in \mathbb{SW}(G)$ é uma árvore (grafo simples, acíclico e conexo).

As estruturas de prova exponenciais não possuem nenhum critério de correcção que se assemelhe ao critério de Danos-Regnier, entretanto, o subconjunto de redes que codificam o lambda-calculus possuem tal critério [Gue04].

Não existe apenas um critério de correcção, uma lista de critérios pode ser encontrada em [DC97, Secção 3.2].

Um resultado importane das redes de prova é o teorema da sequencialização. O teorema estabelece uma equivalência entre a definição indutiva (função $\mathfrak{G}$) e a definição por critério de correcção, isto é, $P \in PS$ é uma rede de prova se e só se P satisfaz o critério de *Danos-Regnier*.

4.3 Interpretação do lambda-calculus

A interpretação do lambda-calculus sobre as redes de prova segue directamente por composição das interpretações anteriores.

Teorema 4.5. *Sejam M um lambda-termo e $\mathcal{D}$ uma derivação de $\Gamma \vdash M : A$, então, existe uma rede de prova com conclusões $?(\Gamma^{\mathcal{GL}})^{\perp}, A^{\mathcal{L}}$.*

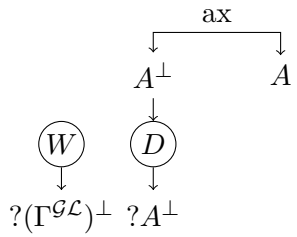
Dem. Basta considerarmos a rede de prova $\mathfrak{G}(\mathcal{D}^{\mathcal{GL}})$. A figura 4.7 mostra o resultado de tal composição. □

Teorema 4.6. *Sejam G, H as redes de prova correspondentes aos termos M, N. Se $M \to_{\beta} N$ então $G \twoheadrightarrow_{cut} H$.*

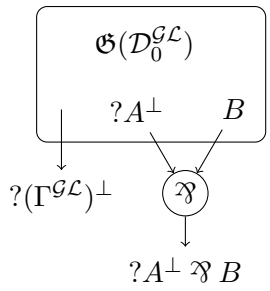
Dem. Foi extendido ao caso de λ-termos puros e redes puras em [Dan90, Reg92] □

Figura 4.7: Interpretação $\mathfrak{G}(\mathcal{D}^{\mathcal{GL}})$

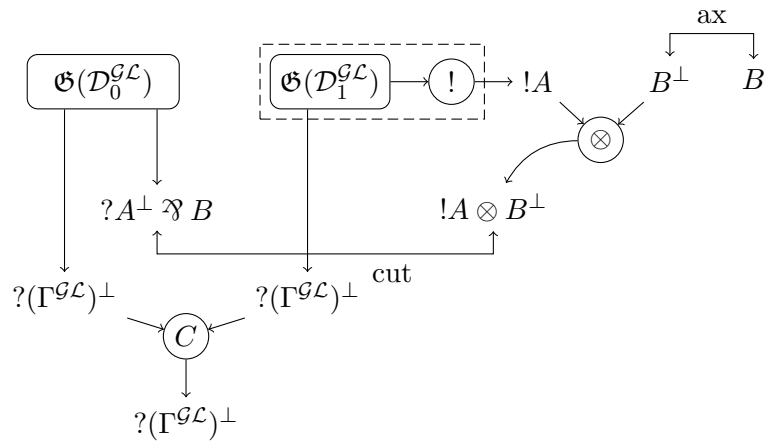
$$\frac{}{\Gamma, x : A \vdash x : A}\ (Ax)$$



$$\frac{\begin{array}{c}\mathcal{D}_0\\ \Gamma, x : A \vdash M : B\end{array}}{\Gamma \vdash (\lambda x.M) : (A \to B)}\ (I \to)$$



$$\frac{\begin{array}{cc}\mathcal{D}_0 & \mathcal{D}_1\\ \Gamma \vdash M : (A \to B) & \Gamma \vdash N : A\end{array}}{\Gamma \vdash MN : B}\ (E \to)$$



Capítulo 5

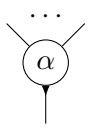
Redes de Interacção

As redes de interacção surgiram em [Laf90], como uma generalização das redes de prova, isto é, a redução numa rede de interacção é semelhante à eliminação de um corte numa rede de prova. Tal formalismo gráfico torna todos os passos em uma computação explícitos e uniformes. A redução sobre as IN's é local e goza de uma confluência forte, isso faz com que as reduções possam ocorrer em qualquer lugar e em paralelo. Por esses motivos as redes de interacção são um paradigma interessante para implementação de linguagens de programação.

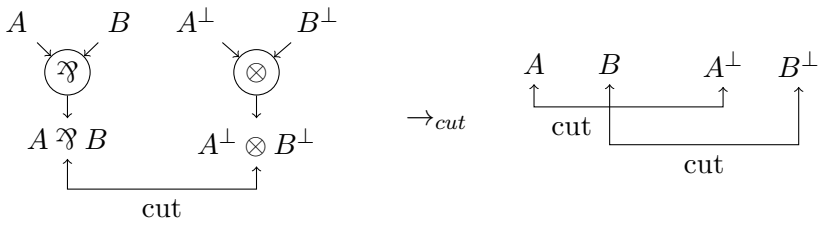
5.1 Redes de Prova versus Redes de Interacção

Nesta secção vamos apresentar as redes de interacção e discutir a sua relação com as redes de prova, para posteriormente podermos aplicá-las ao lambda-calculus.

Vamos considerar um alfabeto Σ de simbolos, ou agentes, com uma aridade ar associada. Cada agente α com $ar(\alpha) = n$ em Σ é representado por um nó com uma porta principal e p portas auxiliares.



Um corte entre dois *links* $\parr$ e $\otimes$ numa rede de prova faz com que tais links sejam reescritos de acordo com as regras da eliminação do corte. O invariante importante nesse processo é que a interface deve ser preservada, se temos quatro conclusões na primeira rede, devemos ter as mesmas quatro conclusões na rede final:



Nas redes de interacção o processo de reescrita funciona de forma semelhante. Se α e β são dois agentes com aridades m e n, respectivamente, uma regra de interacção para $\alpha \bowtie \beta$ é uma redução da seguinte forma:

Onde $\mathcal{T}$ é uma rede de interacção com $m + n$ portas. Ou seja, o processo de redução é iniciado quando existem pelo menos dois agentes conectados pelas suas portas principais. Chamamos de um par activo a dois agentes nessas condições.

Sejam I e J, se I reduz para J pela regra de interacção $\alpha \bowtie \beta$, notamos por $I \Rightarrow_{\alpha,\beta} J$ e $\Rightarrow^*$ representa o fecho reflexivo transitivo da relação de redução.

Definição 5.1 (Sistema de Interacção)**.** Define-se um sistema de interacção por um conjunto de agentes Σ e um conjunto de regras de interacção não ambíguas.

Teorema 5.1 (Confluência forte)**.** *Sejam I, I' e I'' redes de interacção tais que $I \Rightarrow I'$ e $I \Rightarrow I''$, então existe J tal que $I' \Rightarrow J$ e $I'' \Rightarrow J$.*

Dem. Uma vez que as reduções feitas sobre pares activos são locais, a demonstração torna-se trivial. Ora, se $I \Rightarrow_{\alpha,\beta} I'$ e $I \Rightarrow_{\sigma,\delta} I''$, então basta-nos reduzir os pares activos $\sigma \bowtie \delta$ em I' e $\alpha \bowtie \beta$ em I'' para chegarmos à uma mesma rede J. □

5.2 Interpretações do lambda-calculus

Como as redes de interacção são uma generalização das redes de prova, é evidente que podemos traduzir um lambda termo para a sua rede de prova, e depois traduzir a rede resultante numa rede de interacção, Mackie fornece uma codificação da lógica linear sobre as redes de interacção em [Mac00]. As interpretações que seguem tal estratégia são chamadas de interpretações

lógicas. Dentro dessas interpretações temos duas classes: óptimas e não-óptimas.

No ramo das traduções óptimas, isto é, que efectuam o número mínimo de β-reduções necessárias para reduzir um termo, temos, por Gonthier, Abadi e Lèvy (GAL) em [GAL92], uma tradução baseada no algorítmo de Lamping, [Lam90], e na Geometria de Interacção de Girard. Tal tradução foca-se primariamente em implementar a redução óptima, possuindo assim agentes específicos de *sharing* (fan-in e fan-out) e utiliza como ferramenta de implementação um sistema de reescrita e não redes de interacção, uma vez que possui um número infinito de agentes. Note que essa abordagem foi melhorada na BOHM, [AGN95], que extende a redução óptima de Lamping ao *core* de uma linguagem funcional, entretanto, também sai do paradigma das redes de interacção devido à diverssas optimizações implementadas.

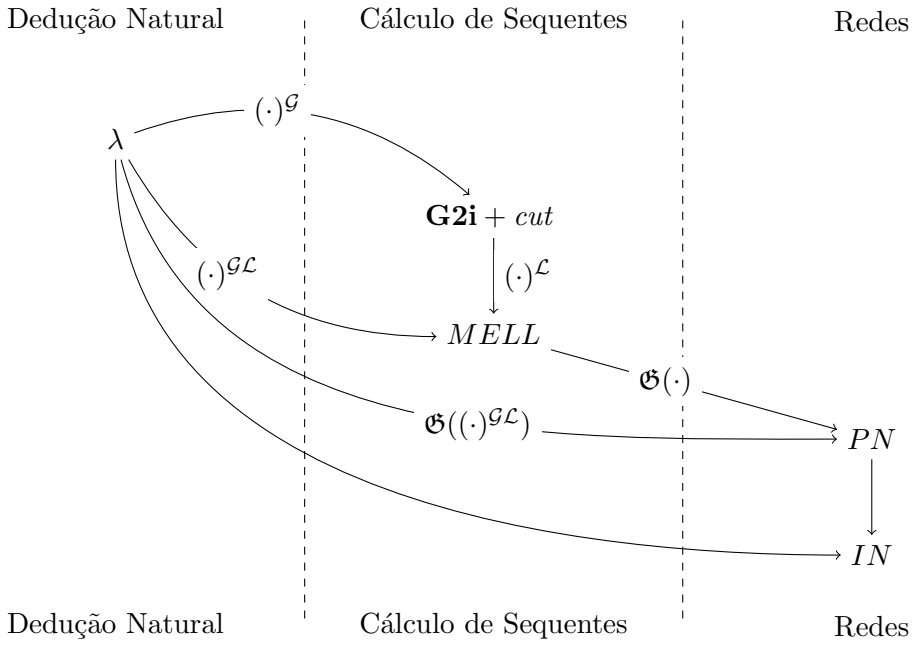
Uma outra implementação óptima, sobre o lambda-calculus sem tipos, é o lambdascope [vOvdLZ04], que utiliza as redes de interacção como mecanismo de implementação. A tradução foca-se na factorização da β-redução em três agentes: (i) Delimitador (apenas um agente delimitador, em contraste com o GAL que utiliza dois), (ii) Duplicador, e, (iii) Apagador. É mais eficiente do que as implementações padrão da BOHM. O preço por essa melhora de desempenho é a dificuldade do *read-back*.

Saindo do escopo das traduções óptimas, uma outra abordagem foi utilizada pelo Ian Mackie, em [Mac94, Mac98, Mac04], utilizando o agente λ ternário, primeiramente utilizado por Abramsky, que encapsula o corpo das abstrações numa estrutura de *box*, com os agentes auxiliares b e v utilizados para representar a lista de variáveis livres (portas auxiliares da caixa). A primeira tradução [Mac94], utiliza os agentes b com as suas portas principais viradas para o agente λ correspondente, deixando a *box* fechada até ocorrer o passo de β-redução que abrirá a *box*. O YALE [Mac98] visa corrigir tal problema, orientando as portas principais dos agentes b para baixo, utilizando uma estratégia de redução que segue quase directamente de eliminações eficientes do corte sobre MELL. O KCLE [Mac04], por sua vez, visa melhorar o YALE ajustando as regras de interacção, e consegue reduzir termos até a sua forma β-normal num número de passos de interacção menor que a BOHM para termos relativamente grandes. A redução até a forma β-normal fornece uma noção de *read-back* simples.

Algumas traduções focam-se no comportamento óptimo sobre a β redução, outras procuram reduzir o número de passos de interacção globais na rede, porém ainda existe outra tradução que visa minimizar o número de agentes utilizados na codificação. A tradução do Mackie e S. Pinto [MP02], utiliza apenas os três combinadores de Lafont [Laf95], δ, ϵ e γ, para codificar a lógica linear e apresenta um sistema não óptimo, porém eficiente no número de passos de interacção.

Abordando o tema de uma perspectiva completamente diferente, podemos também traduzir um lambda-termo para uma rede de interacção directamente, por meio de uma interpretação directa. A secção 5.3 apresenta um estudo detalhado de uma tradução directa.

Mapa de Interpretações



5.3 Tradução de Sinot

Nesta secção vamos apresentar um sistema de interacção bastante semelhante ao sistema em [Sin06a], o sistema consiste em representar graficamente as regras da definição 5.2, para isso vamos utilizar os agentes introduzidos na figura 5.1. Como é evidente, teremos agentes específicos do lambda-calculus e agentes estruturais, como era o caso das redes de prova.

O sistema discutido nesta secção aborda apenas a estratégia *call-by-name*, entretanto, Sinot possui outras duas traduções que simulam as estratégias *call-by-value* e *call-by-need* em [Sin06a, Sin06b].

5.3.1 Call-by-Name, com tokens

Nesta secção vamos estudar, com mais detalhes, a estratégia de redução apresentada na secção 2.3, desta vez com o intúito de apresentar uma codificação em redes de interacção capaz de reduzir um lambda-termo seguindo a estratégia *call-by-name*.

Para tornar mais clara a passagem do lambda-calculus para as redes de interacção vamos precisar de um novo cálculo, apresentado em [Sin06a]. O cálculo novo consiste no cálculo tradicional acrescido dos *tokens* $\Downarrow$ e $\Uparrow$, denominados *eval* e *return* respectivamente. O conjunto $\mathcal{S}$ de termos é definido pela seguinte gramática:

$$\mathcal{S} ::= \Downarrow M \mid \Uparrow W \mid \mathcal{S}@N$$

Onde M, N são lambda-termos e W é um termo na whnf.

Vamos também considerar uma função $|\cdot| : \mathcal{S} \to \Lambda$ que apaga os *tokens* $\Downarrow$ e $\Uparrow$ dos termos de $\mathcal{S}$. Dizemos que um termo $T \in \mathcal{S}$ está na whnf se $|T|$ estiver na whnf.

Vamos denotar os termos de $\mathcal{S}$ por $T, U, V, ...$ e vamos manter $M, N, ...$ para denotar os termos de Λ.

Definição 5.2 ($\to_s$)**.** Sejam M, N, W lambda-termos, com W uma whnf, e x uma variável então define-se a seguinte relação binária sobre $\mathcal{S}$:

$\Downarrow M$	$\to_s$	$\Uparrow M$, se M for uma abstração ou uma variável.
$\Downarrow (MN)$	$\to_s$	$(\Downarrow M)@N$
$(\Uparrow x)@N$	$\to_s$	$\Uparrow (xN)$
$(\Uparrow (WM))@N$	$\to_s$	$\Uparrow ((WM)N)$
$(\Uparrow (\lambda x.M))@N$	$\to_s$	$\Downarrow [N/x]M$

Se $T \to_s T'$ então $T@N \to_s T'@N$.

Se observarmos atentamente a definição 2.9 ($\Downarrow$, *call-by-name*), podemos ver que a definição de $\to_s$ seque das regras de $\Downarrow$, e de facto possuímos equivalência entre as duas relações, como vamos demonstrar a seguir.

Lema 5.1. Sejam $T, U \in \mathcal{S}$ então seque que $T \twoheadrightarrow_s U$ implica que $|T| \twoheadrightarrow_{wh\beta} |U|$.

Dem. Indução no comprimento de $T \twoheadrightarrow_s U$.

□

Lema 5.2. Sejam $M, N \in \Lambda$. Se M não possui whnf então MN também não possui whnf.

Dem. A demonstração segue por contradição. Assumindo que MN possui whnf W, então $MN \twoheadrightarrow_{wh\beta} W$ e pelo lema 2.9 segue que $M \twoheadrightarrow_{wh\beta} W'$, mas por hipótese M não possui whnf.

□

Lema 5.3. Se um termo M não possui whnf, então não existe W whnf tal que $\Downarrow (MN) \twoheadrightarrow_s \Uparrow W$, com $N \in \Lambda$.

Dem. Assumindo que M não possui whnf, a demonstração segue por contradição. Pelo lema 5.1 e por hipótese temos que $MN \twoheadrightarrow_{wh\beta} W$, mas se M não possui whnf, pelo lema 5.2, MN não reduz para whfn por $\twoheadrightarrow_{wh\beta}$.

□

Lema 5.4. Sejam $M, N \in \Lambda$. Se $M \Downarrow N$ então $\Downarrow M \twoheadrightarrow_s \Uparrow N$.

Dem. A demonstração segue por indução em $\Downarrow$. □

Se combinarmos alguns resultados clássicos com o lema 5.4 podemos ver que a relação $\rightarrow_s$ também encontra a whnf de um termo, caso ela exista.

Lema 5.5. Sejam $M, W, W' \in \Lambda$. Se M possui whnf W', isto é, $M \twoheadrightarrow_\beta W'$ então $\Downarrow M \twoheadrightarrow_s \Uparrow W$, e W é um termo na whnf.

Dem. Temos que:

$m \twoheadrightarrow_\beta w' \Rightarrow m \twoheadrightarrow_{wh\beta} w$	Resultados clássicos
$\Rightarrow m \Downarrow w$	Teorema 2.4
$\Rightarrow \Downarrow m \twoheadrightarrow_s \Uparrow w$	Lema 5.4

□

Teorema 5.2 ([Sin06a], Prop. 2)**.** *Sejam $M, N \in \Lambda$ então $\Downarrow M \twoheadrightarrow_s \Uparrow N$ se e só se $M \Downarrow N$.*

Dem.

$\Leftarrow$) Lema 5.4.

$\Rightarrow$) Queremos provar que, se $\Downarrow M \twoheadrightarrow_s \Uparrow N$ então $M \Downarrow N$, a demonstração segue por indução no comprimento k de $\twoheadrightarrow_s$. Assumindo que $\Downarrow M \rightarrow_s^k \Uparrow N$, então:

Caso $k = 1$, é imediato que $M \equiv \lambda x.M' \equiv N$ ou $M \equiv x \equiv N$.
É evidente que $(\lambda x.M') \Downarrow (\lambda x.M')$.

Caso $k > 1$, é evidente que $M \equiv M'N'$, a demonstração segue por casos.

Se M possuir whnf $\lambda x.M''$, então pelo lema 5.5 temos que $\Downarrow M' \twoheadrightarrow_s \Uparrow (\lambda x.M'')$. Podemos agora analisar o início da cadeia de reduções:

$$\begin{array}{lll} \Downarrow (M'N') & \rightarrow_s & (\Downarrow M')@N' \\ & \twoheadrightarrow_s & (\Uparrow (\lambda x.M''))@N' \\ & \rightarrow_s & \Downarrow [N'/x]M'' \\ & \twoheadrightarrow_s & \Uparrow N \end{array}$$

Por hipótese de indução temos que $[N'/x]M'' \Downarrow N$, logo:

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

Se M' possuir whnf $x\,\vec{v}$ a demonstração é análoga ao caso anterior.

Se M' não possuir whnf, utilizando o lema 5.3, $\Downarrow (M'N') \twoheadrightarrow_s \Uparrow N$ não acontece, assim como $M \Downarrow N$ também não está definido.

□

Corolário 5.1. *Sejam $M, N \in \Lambda$, com N uma whnf. Temos que $M \twoheadrightarrow_{wh\beta} N$ se e só se $\Downarrow M \twoheadrightarrow_s \Uparrow N$.*

Dem. A demonstração segue a partir do teorema 5.2 e da equivalência entre $\twoheadrightarrow_{wh\beta}$ e $\Downarrow$. □

5.3.2 Sistema de Interacção

Vamos aqui apresentar o sistema de interacção em detalhe. A figura 5.1 mostra os agentes utilizados. As regras de interacção são apresentadas em (A) e (B) e sumariadas pela figura 5.2, com α um agente qualquer.

A seguir vamos apresentar a função $\mathcal{T} : \Lambda \rightarrow IN$, que traduz lambda-termos para redes de interacção. Uma vez que passaremos para um domínio gráfico, perderemos a noção de nome de uma variável. As variáveis livres de um termo M serão apenas arestas abertas na rede $\mathcal{T}(M)$, podemos então ordenar essas arestas de forma à, possuindo a rede $\mathcal{T}(M)$ e o termo M, sabemos quais arestas correspondem a quais variáveis livres. O invariante mais simples é ordenar as arestas de acordo com a ocorrência textual das variáveis.

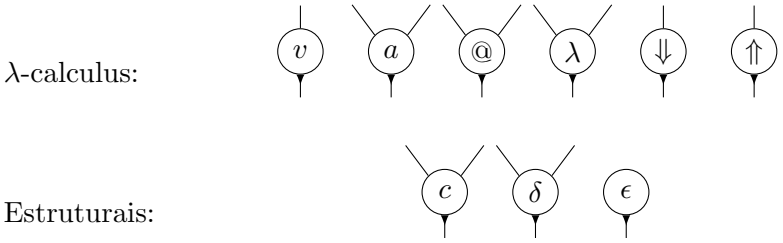


Figura 5.1: Agentes do sistema de interacção

λ-calculus: $v \bowtie \Downarrow$, $\lambda \bowtie \Downarrow$, $a \bowtie \Downarrow$, $\Uparrow \bowtie @$, $\lambda \bowtie @$, $a \bowtie @$, $v \bowtie @$.
Estruturais: $\epsilon \bowtie \alpha$, $c \bowtie \lambda$, $c \bowtie \alpha$, $\delta \bowtie \delta$, $\delta \bowtie v$, $\delta \bowtie \alpha$.

Figura 5.2: Regras de interacção

Definição 5.3 (*Wiring*)**.** Seja ω uma rede de interacção, dizemos que ω é um *wiring*, ou, religação, se ω é apenas uma reordenação de arestas.

Variáveis. Á tradução de variáveis, tanto livres como ligadas, será dada pelo agente v, figura 5.3. Note que o invariante é trivialmente é trivialmente garantido.

Figura 5.3: $\mathcal{T}(x)$

Aplicações. A tradução $\mathcal{T}(MN)$ de uma aplicação é dada pela figura 5.4. As ocorrências livres da mesma variável devem ser agrupadas com contracções. O *wiring* ω reordena as variáveis, mantendo o invariante acerca da ordem das variáveis.

Abstrações. A tradução $\mathcal{T}(\lambda x.M)$ é o caso mais semelhante às redes de prova. Se x ocorre em M, então tem-se a rede da esquerda (com o x representado apenas por uma aresta). Se x não ocorre em M, temos o agente ϵ, que desempenha um papel semelhante ao enfraquecimento da lógica linear.

A função $\mathcal{K}_x : IN \times \Lambda \to IN$ é a função que apaga as ocorrências do agente v relativos às ocorrências de uma variável na rede que traduz um lambda-termo. A definição de $\mathcal{K}_x(\mathcal{T}(M), M)$ é dada pela figura 5.6 Note que a rede $\mathcal{K}_x(\mathcal{T}(M), M)$ quebra o invariante das redes, mas não é problema uma vez que a aresta x será ligada à um agente λ.

Os três casos acima traduzem termos em Λ, vamos agora extender a tradução

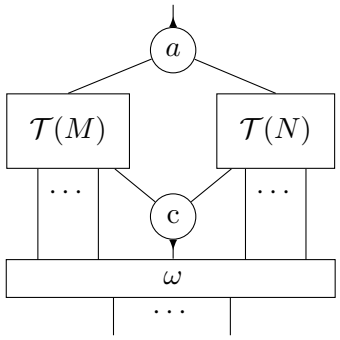


Figura 5.4: $\mathcal{T}(MN)$

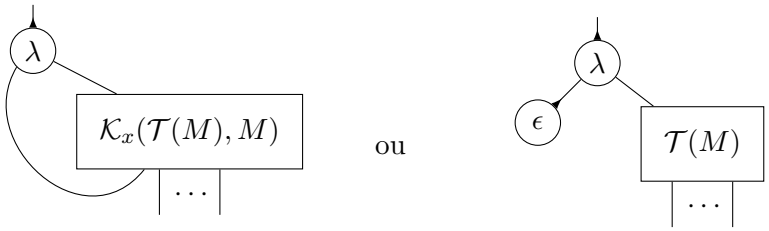


Figura 5.5: $\mathcal{T}(\lambda x.M)$

$\mathcal{T}$ para $\mathcal{S}$. A aplicação entre um termo $T \in \mathcal{S}$ e um termo $N \in \Lambda$ é muito parecida com a tradução de uma aplicação normal e é dada por pela figura 5.7.

Um aspecto importante a salientar é que a rede $\mathcal{T}(T)$, resultante de um termo fechado T, não tem nenhum par activo, ou seja, está na forma normal, ainda mais importante, possui apenas uma porta principal, na sua raiz. Por essa razão que utiliza-se mais dois agentes unários, $\Downarrow$ e $\Uparrow$, para controlarmos a redução num termo.

Lema 5.1. Seja $T \in \Sigma$ um termo fechado sem nenhum token, então $\mathcal{T}(T)$ possui apenas uma porta principal.

Dem. A demonstração segue por indução em T. □

Para reduzir uma rede $\mathcal{T}(M)$, basta-nos criar um par activo com a porta principal da rede e o token $\Downarrow$, o melhor é, de facto, extender a tradução para abrangir também os termos em Σ. Define-se $\mathcal{T}(\Downarrow M)$ e $\mathcal{T}(\Uparrow M)$ respectivamente por:

Note que as traduções acima só são possíveis por causa do teorema 5.1, que

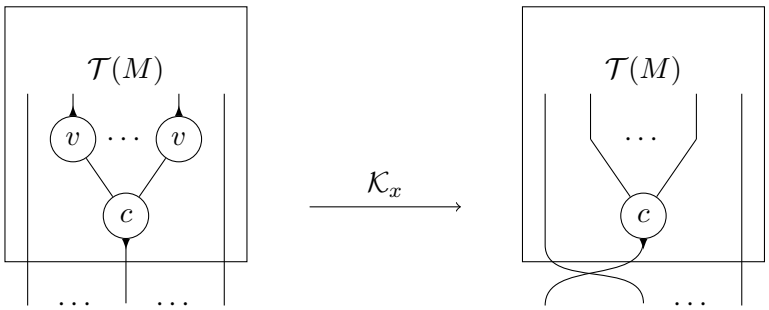


Figura 5.6: Definição de $\mathcal{K}_x$

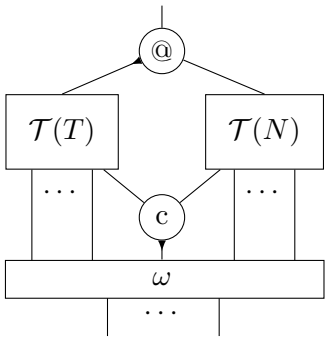


Figura 5.7: $\mathcal{T}(T@N)$

nos fornece um invariante acerca da forma da rede resultante de um termo, seguindo a tradução $\mathcal{T}$.

A figura 5.9 exemplifica a tradução $\mathcal{T}$ de dois lambda-termos.

A função $\mathcal{T}$ apresentada em [Sin06a] apenas traduz os termos em Λ, criando o par activo com o token $\Downarrow$ antes de iniciar a avaliação da rede. Entretanto, a tradução $\mathcal{T}$ apresentada neste documento aborda os termos em $\mathcal{S}$, facilitando a resolução de problemas relacionados com o *read-back*, além de se integrar melhor com o sistema $\rightarrow_s$ para a redução. O tratamento das variáveis feito pelo Sinot também é bastante diferente. Sinot não utiliza agentes v para todas as variáveis e não ordena as arestas livres.

5.3.3 Regras de Interacção

As regras de interacção são divididas em duas categorias: (A) regras de redução e (B) regras estruturais. As regras de redução são aquelas que simulam a redução $\rightarrow_s$ e as regras estruturais são responsáveis pelas contracções e enfraquecimentos de termos.

Figura 5.8: $\mathcal{T}(\Downarrow M)$ e $\mathcal{T}(\Uparrow M)$

$\mathcal{T}((\lambda xy.xyx))$ $\mathcal{T}((\Downarrow((\lambda x.x)y))@y)$

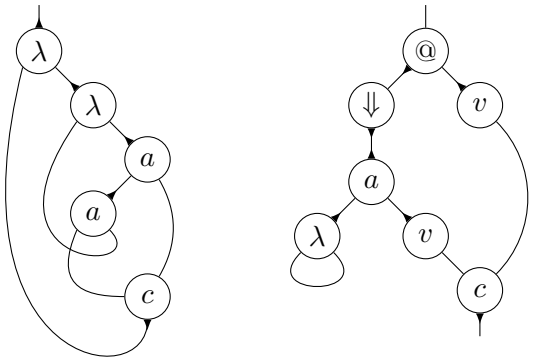
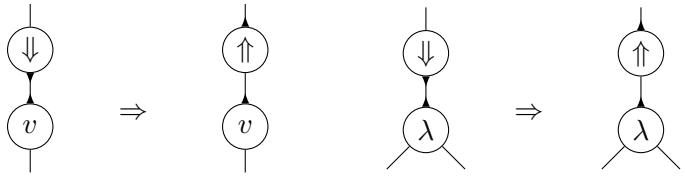


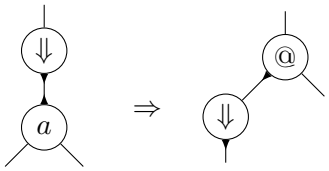
Figura 5.9: Exemplos da traduçao $\mathcal{T}$

(A) Regras de Redução.

De acorco com a definição 5.2, primeira regra, a situação mais simples é quando se encontra uma abstração ou uma variável, apenas retorna-se tal abstração ou variável.

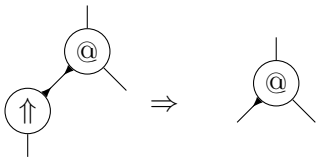


Se olharmos para a segunda regra da definição 5.2, é simples ver que precisamos avaliar o lado esquerdo da aplicação. Um novo agente binário, @, será preciso.

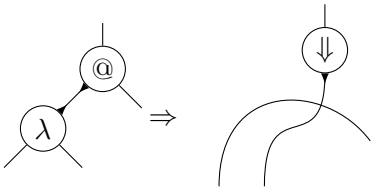


Quando a avaliação chega a um agente a, muda-se o agente para @, ainda representando uma aplicação, mas com a sua porta principal para a esquerda, de forma a esperar que o agente $\Uparrow$ retorne.

Quando o agente $\Uparrow$ retorna, ou traz uma abstração ou uma aplicação da forma $xN_1 \cdots N_n$, o mais simples a ser feito é apagar o agente $\Uparrow$ e criar uma regra para cada caso, com o compromisso de que as duas regras devem criar ou um agente $\Uparrow$ ou $\Downarrow$.

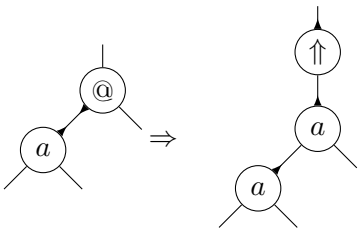


Caso encontre-se uma abstração, estamos no caso $(\Uparrow\ (\lambda x.M))@N \rightarrow_s \Downarrow [N/x]M$.

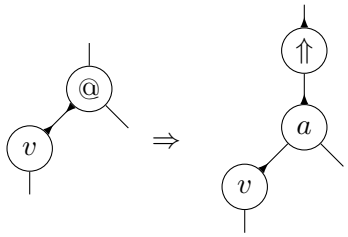


A regra $\lambda \bowtie @$ ilustra uma das grandes mais valias das redes de interacção, nomeadamente a simplicidade da operação de substituição, apenas junta-se a aresta no argumento do redex para a variável da abstração, de seguida avalia-se o corpo da abstração.

Caso encontre-se uma aplicação, estamos no caso $(\Uparrow (WM))@N \rightarrow_s \Uparrow (WMN)$, devemos apenas retornar tudo.



Caso encontre-se uma variável, estamos no caso $(\Uparrow x)@N \rightarrow_s \Uparrow (xN)$.



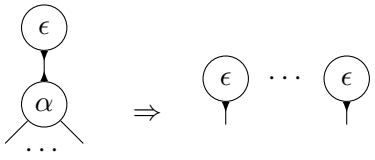
Note que as três regras anteriores ($\lambda \bowtie @$, $a \bowtie @$ e $v \bowtie @$) criam um e um só dos agentes $\Downarrow$ e $\Uparrow$, como era suposto.

O sistema introduzido em [Sin06a] utiliza um agente $\Uparrow_o$ para retornar lambda-termos da forma $xN_1 \cdots N_n$, em contraste com o sistema aqui apresentado, que retorna todas as whnf's com o agente $\Uparrow$.

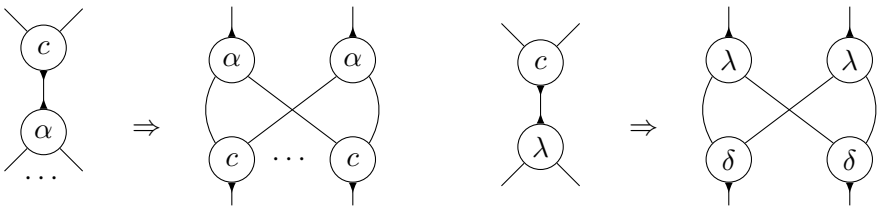
(B) Regras Estruturais

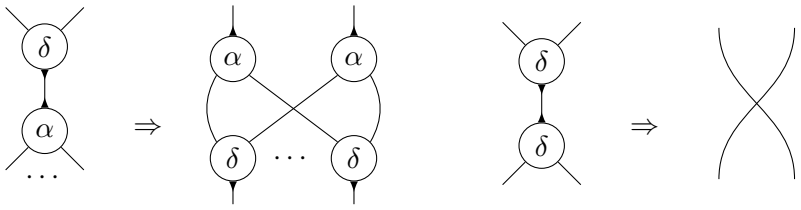
Para além dos agentes já introduzidos até agora, serão precisos mais alguns para controlarmos os recursos disponíveis. Teremos dois agentes binários de cópia, c e δ, e um agente unário de enfraquecimento, ϵ.

Seja α um agente qualquer, então:

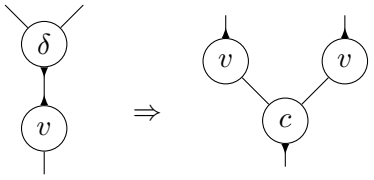


O agente c duplicará qualquer agente que encontre, excepto abstrações. Para prevenir que a rede duplique-se indefinidamente, vamos utilizar um agente auxiliar δ, que tem o mesmo papel que c, porém é utilizado apenas para abstrações, a única diferença é que quando dois agentes δ formam um par activo, anulam-se.



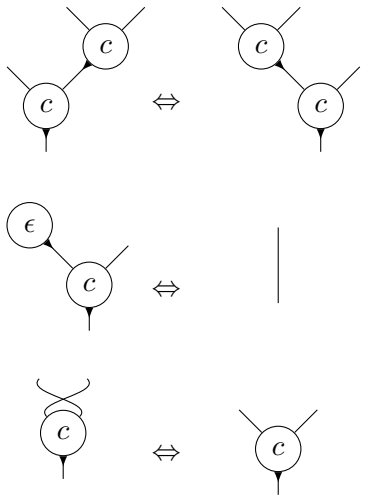


Uma vez que podemos ter variáveis livres dentro de uma abstração, é preciso algum cuidado com a situação $\delta \bowtie v$, dado que tais variáveis podem vir a serem ligadas mais tardes, ao juntar-se redes, é mais simples transformar o agente δ no agente c outra vez.



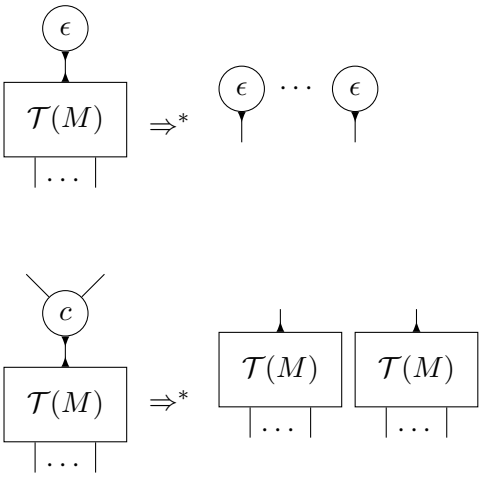
De forma a facilitar o tratamento de redes, vamos introduzir duas congruências estruturais:

Definição 5.4 (Congruências Estruturais)**.**



A partir de agora, trabalharemos com a redução de redes módulo congruências estruturais, ou seja, sejam R e R' redes de interacção, $R \Rightarrow^* R'$ se R reduz por passos de interacção e congruências estruturais para R'.

Teorema 5.3 (Duplicação e Apagamento)**.** *Seja M um lambda-termo, então:*



Dem. Basta decompor $\mathcal{T}(M)$ em uma *tree* e um *wiring*, como definido em [Laf95], em seguida aplicar os lemas 2.2 e 2.3 do mesmo artigo. □

Lema 5.2 (Lema da Substituição)**.** Seja $[\mathcal{T}(N)/x]\mathcal{T}(T)$ a rede que resulta da substituição de x por $\mathcal{T}(N)$ em $\mathcal{T}(T)$, figura 5.10, onde $T \in \mathcal{S}$ é um termo com ocorrências livres da variável x, N um lambda-termo e $\omega + c$ é uma rede composta de contrações e um *wiring*, por forma a preservar o invariante da ordem das arestas. Então $[\mathcal{T}(N)/x]\mathcal{T}(T) \Rightarrow^* \mathcal{T}([N/x]T)$.

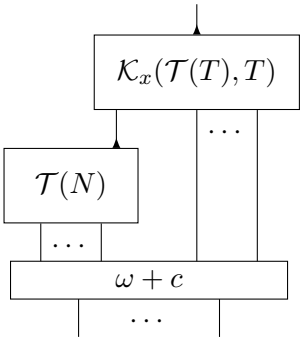


Figura 5.10: Definição da rede $[\mathcal{T}(N)/x]\mathcal{T}(T)$

Dem. Indução em t. □

Teorema 5.4. *Sejam $T, U \in \mathcal{S}$. Se $T \rightarrow_s U$ então $\mathcal{T}(T) \Rightarrow^* \mu$, onde μ é uma rede de interacção composta por $\mathcal{T}(U)$ e um dado número de agentes ϵ não conectados.*

Dem. A demonstração segue por indução em $\rightarrow_s$. Vamos apresentar a sequência de regras de interacção para cada caso de base. Caso $\mathcal{T}(T)$ seja:

$\mathcal{T}(\Downarrow x) \overset{\Downarrow\bowtie v}{\Longrightarrow} \mathcal{T}(\Uparrow x)$

$\mathcal{T}(\Downarrow (\lambda x.M)) \overset{\Downarrow\bowtie\lambda}{\Longrightarrow} \mathcal{T}(\Uparrow (\lambda x.M))$

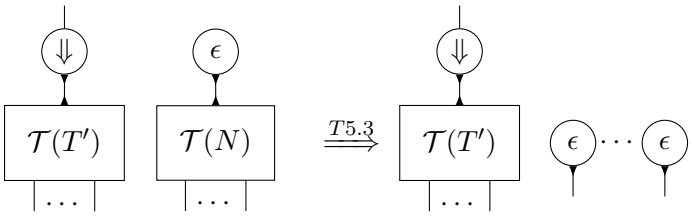
$\mathcal{T}(\Downarrow (MN)) \overset{\Downarrow\bowtie a}{\Longrightarrow} \mathcal{T}((\Downarrow M)@N)$

$\mathcal{T}((\Uparrow x)@N) \overset{\Uparrow\bowtie@}{\Longrightarrow} \cdot \overset{v\bowtie@}{\Longrightarrow} \mathcal{T}(\Uparrow (xN))$

$\mathcal{T}((\Uparrow (WM)@N) \overset{\Uparrow\bowtie@}{\Longrightarrow} \cdot \overset{a\bowtie@}{\Longrightarrow} \mathcal{T}(\Uparrow (WMN))$

$\mathcal{T}((\Uparrow (\lambda x.T'))@N) \overset{\Uparrow\bowtie@}{\Longrightarrow} \cdot \overset{@\bowtie\lambda}{\Longrightarrow} \cdot \overset{L.5.2}{\Longrightarrow} \mathcal{T}(\Downarrow [N/x]T')$, se x ocorre em T'.

O caso de uma abstração vacuosa é o menos intuitivo:
$\mathcal{T}((\Uparrow (\lambda x.T'))@N) \overset{\Uparrow\bowtie@}{\Longrightarrow} \cdot \overset{@\bowtie\lambda}{\Longrightarrow}$



O caso indutivo vem directamente da localidade e da preservação da interface de um passo de interacção. □

Aplicando o teorema anterior repetidamente temos o seguinte diagrama:

$$\begin{array}{ccccccccc} \Downarrow T & \xrightarrow[s]{} & U_1 & \xrightarrow[s]{} & U_2 & \xrightarrow[s]{} & \cdots & \xrightarrow[s]{} & \Uparrow V \\ | & & | & & | & & & & | \\ \mathcal{T} & & \mathcal{T} & & \mathcal{T} & & & & \mathcal{T} \\ \downarrow & & \downarrow & & \downarrow & & & & \downarrow \\ \mathcal{T}(\Downarrow T) & \overset{*}{\Longrightarrow} & \mathcal{T}(U_1) & \overset{*}{\Longrightarrow} & \mathcal{T}(U_2) & \overset{*}{\Longrightarrow} & \cdots & \overset{*}{\Longrightarrow} & \mathcal{T}(\Uparrow V) \end{array}$$

Portanto, pelo diagrama acima e pelo teorema 5.2 temos o sequinte corolário:

Corolário 5.2. *Sejam $M, N \in \Lambda$ tais que $M \Downarrow N$ então*
$\mathcal{T}(\Downarrow M) \Rightarrow^* \mathcal{T}(\Uparrow N)$.

Capítulo 6

Conclusão

Neste documento apresentou-se tanto o processo de tradução completo dos lambda-termos às redes de interacção como um exemplo detalhado de uma tradução directa. O trabalho consistiu numa análise e classificação das traduções lógicas, em óptimas e não-óptimas e no estudo pormenorizado da tradução de Sinot, incluindo a formalização do lambda-calculus de Sinot.

O objectivo principal, que consistia no estudo e na apresentação do processo de tradução do lambda-calculus para as redes de interacção foi completo, entretanto não foi possível estudar com o mesmo nível de detalhamento todas as traduções existentes, como ficou claro neste documento, a tradução em [Sin06a] foi estudada mais profundamente.

O estudo das redes de prova, por outro lado, foi levado muito mais a fundo do que inicialmente suposto. Todas as traduções utilizam redes de prova, uma vez que as redes de interacção são uma generalização delas. Entretanto, as traduções lógicas passam explicitamente pelas redes de prova, enquando as directas utilizam-nas apenas implicitamente. As redes de prova levantaram algumas questões interessantes, que ficaram como trabalho futuro: (A) efectuar o *read-back* de redes de prova baseado num critério geométrico, e, (B) estudar traduções de outras variantes do lambda-calculus, nomeadamente o cálculo com substituições explíticas, λ_x, uma vez que tal cálculo parece facilitar o *read-back* das redes de prova.

O ramo das interpretações directas também deixou algumas portas abertas para um refinamento da tradução: (C) Em que medida o sistema apresentado deve ser alterado para reduzir os termos até a sua forma β-normal, e, (D) poderia um agente v com duas portas principais facilitar a abordagem para termos abertos.

Bibliografia

[AGN95] A. Asperti, C. Giovannetti, and A. Naletto. The bologna optimal higher-order machine. Technical report, 1995.

[Bar84] H. Barendregt. *The lambda calculus: its syntax and semantics*. North Holland, 2nd. revised edition, 1984.

[Bar93] H. Barendregt. *Handbook of Logic in Computer Science: Lambda Calculi with Types*, volume 2. Oxford University Press, 1993.

[Chu32] A. Church. A set of postulates for the foundation of logic. *Annals of Mathematics*, 33(2), 1932.

[Chu36a] A. Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1, 1936.

[Chu36b] A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2), 1936.

[Chu40] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(2), 1940.

[Dan90] Vincent Danos. *La Logique Linéaire appliquée à l'étude de divers processus de normalisation (principalement du λ-calcul)*. PhD thesis, Université de Paris 7, 1990.

[DC97] V. Danos and R. Di Cosmo. The linear logic primer. http://www.dicosmo.org/CourseNotes/LinLog/, 1997.

[DR89] V. Danos and L. Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

[GAL92] G. Gonthier, M. Abadi, and J.-J. Lévy. The geometry of optimal lambda reduction, 1992.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[Gir95] Jean-Yves Girard. Linear logic: its syntax and semantics. pages 1–42, 1995.

[Gue04] Stefano Guerrini. *Proof nets and the lambda-calculus*, pages 65–118. Cambridge University Press, 2004.

[HCS72] J. Hindley H. Curry and P. Seldin. Combinatory logic. *North-Holland Elsevier*, 2, 1972.

[Her95] Hugo Herbelin. A lambda-calculus structure isomorphic to Gentzen-style sequent calculus structure. In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers*, volume 933 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 1995.

[HS86] J. Hindley and P. Seldin. *Introduction to combinators and [lambda]-calculus*. London Mathematical Society student texts. Cambridge University Press, 1986.

[Laf90] Yves Lafont. Interaction nets. In *POPL*, pages 95–108. ACM Press, 1990.

[Laf95] Yves Lafont. Interaction combinators. *Information and Computation*, 137:69–101, 1995.

[Lam90] John Lamping. An algorithm for optimal lambda calculus reduction. pages 16–30. ACM Press, 1990.

[Mac94] Ian Mackie. A lambda-evaluator based on interaction nets. In *Theory and Formal Methods*, pages 41–60, 1994.

[Mac98] Ian Mackie. Yale: yet another lambda evaluator based on interaction nets. In *Proceedings of the third ACM SIGPLAN international conference on Functional programming*, 1998.

[Mac00] Ian Mackie. Interaction nets for linear logic. *Theoretical Computer Science*, 247:83–140, 2000.

[Mac04] Ian Mackie. Efficient λ-evaluation with interaction nets. In Vincent van Oostrom, editor, *Rewriting Techniques and Applications*, volume 3091 of *Lecture Notes in Computer Science*, pages 155–169. Springer Berlin / Heidelberg, 2004.

[MP02] Ian Mackie and Jorge Sousa Pinto. Encoding linear logic with interaction combinators. *Inf. Comput.*, 176(2):153–186, 2002.

[Pra64] D. Prawitz. *Natural Deduction: A Proof-Theoretical Study*. Dover, 1964.

[Reg92] L. Regnier. *Lambda-calcul et résaux*. PhD thesis, Unisertié de Paris 7, 1992.

[Sin06a] F. R. Sinot. Call-by-name and call-by-value as token-passing interaction nets. *Mathematical Structures in Computer Science*, 16(4), 2006.

[Sin06b] F. R. Sinot. Call-by-need in token-passing nets. *Mathematical. Structures in Comp. Sci.*, 16(4):639–666, 2006.

[TS00] A. Troelstra and H. Schichtenberg. *Basic Proof Theory*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2nd edition edition, 2000.

[vOvdLZ04] V. v. Oostrom, K.-J. v. de Looij, and M. Zwitserlood. Lambdascope: another optimal implementation of the lambda-calculus. April 2004.