Introdução ao $\boldsymbol{\lambda}$-Calculus

José Manuel Valença

Março, 1998

Conteúdo

1 Motivação

O **$\boldsymbol{\lambda}$**-Calculus é uma linguagem para descrever, essencialmente, o conceito de transformação de entidades sintácticas.

Na sua versão original (nos anos 30) destinava-se à representação de *"funções-como-transformações"* em alternativa à noção de *"funções-como-conjuntos"*.

Porém, as modernas Ciências da Computação usam o **$\boldsymbol{\lambda}$**-Calculus para descrever uma gama muito vasta de objectos ditos *"computacionais"*: funções, programas, provas, processos, etc.

Todas as entidades assim descritas satisfazem dois princípios muito importantes:

(i) As entidades objecto da transformação têm uma descrição simbólica (que é descrita por uma sintaxe apropriada) e o processo de transformação assenta na *substituição* de símbolos por entidades sintácticas.

(ii) As transformações são, elas próprias, entidades sintácticas e a sua descrição e transformação obedece aos mesmos princípios que as entidades objecto.

Na linguagem aqui descrita as entidades sintácticas fundamentais vão ser designadas por **expressões-$\boldsymbol{\lambda}$** ou então **$\boldsymbol{\lambda}$-termos**.

1.1 Variáveis e Constantes

De entre as expressões-λ deve-se destacar aquelas que não são decomponíveis em outras expressões com a mesma natureza. Estas expressões-λ dizem-se **atómicas**.

As expressões-λ atómicas são identificadas por **símbolos**. Exemplos de símbolos usados nesta linguagem são

$$0 \quad , \quad x \quad , \quad + \quad , \quad \cdots$$

Além dos símbolos que identificam expressões-λ atómicas, existem outros símbolos na linguagem; tais símbolos são normalmente designados por **meta-símbolos**.

A linguagem do **$\boldsymbol{\lambda}$**-Calculus contém os seguintes meta-símbolos cujo significado iremos sugerir ao longo desta secção:

Parêntises: "(" e ")".

Abstracção: "λ"

Substituição: "$[\,.\,/\,.\,]$"

Dado que a substituição sintáctica pretende ser a noção computacional básica, é necessário introduzir uma distinção entre expressões-λ atómicas:

(a) VARIÁVEIS: expressões-λ que podem ser objecto de substituição.

(b) CONSTANTES: expressões-λ que podem substituir variáveis mas não podem ser objecto de substituições.

EXEMPLO 1.1

x	é uma variável identificada pelo símbolo com este nome
1	é uma constante identificada pelo símbolo numeral 1
$(+1)\,x$	não é nem constante nem variável; é uma expressão-λ composta

1.2 Aplicação e Abstracção

As expressões-λ que não são atómicas são geradas a partir de outras expressões-λ através de duas construções sintácticas: a **aplicação** e a **abstracção**.

Antes de analisarmos a primeira destas construções é necessário referir que no **$\boldsymbol{\lambda}$**-Calculus todas as transformações são consideradas **unárias**: têm apenas um argumento.

Quando se pretende descrever transformações com mais de um argumento usa-se o conhecido processo de *Currying*.

EXEMPLO 1.2

Para representar a função binária adição, no domínio $\mathbb{N}$ dos naturais, usa-se a constante $+$ com o tipo $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$.

Assim, aplicando $+$ a um n em $\mathbb{N}$, dá como resultado uma função unária $\mathbb{N} \to \mathbb{N}$,

$$(+\,n) : \mathbb{N} \to \mathbb{N}$$

Para representar a expressão algébrica $n + m$ usa-se, portanto, a expressão-λ

$$(+\,n)\,m$$

Genericamente, a aplicação de uma expressão M a uma expressão N escreve-se simplesmente

$$M\,N \tag{1}$$

O operador aplicação está implícito na ordem com que são escritas estas entidades. A operação considera-se associativa à esquerda; justifica-se, assim, a seguinte notação.

NOTAÇÃO 1.1 *A expressão-λ*

$$((M\, N_1)\, N_2)\, \cdots\, N_k$$

pode ser denotada, equivalentemente, por

$$M\, N_1\, N_2\, \cdots\, N_k$$

A aplicação é a construção sintáctica que denota as transformações do *argumento* N efectuadas pelo *transformador* M.

A abstracção é, de certa forma, a operação inversa: constrói uma entidade transformadora a partir de resultados genéricos de transformações.

Os resultados genéricos de uma transformação podem ser representados por um λ-termo M onde ocorre uma variável x (ou várias variáveis $x, y, \cdots$).

A ocorrência da variável x em M pode ser tornada explícita se representar-mos a expressão por

$$M[x]$$

Cada substituição de x determina um dos *resultados* particulares desta transformação.

A operação de abstracção pretende construir esta transformação determinada pelos resultados $M[x]$.

Sintácticamente a transformação representa-se por

$$(\lambda\, x.\, M) \tag{2}$$

O significado desta construção deve traduzir a seguinte intuição

> *aplicada a um argumento genérico N, a transformação $(\lambda\, x.\, M)$*
> *produz um resultado que se obtém substituindo x por N em M* (3)

Tal como a notação 1.1 representa uma forma simplificada da sintaxe para a aplicação múltipla, é também possível definir uma sintaxe simplificada para a abstracção múltipla.

NOTAÇÃO 1.2 *A expressão-λ*

$$(\lambda\, x_1.\, (\lambda\, x_2.\, \cdots (\lambda\, x_n.\, M)))$$

é denotada, equivalentemente, por

$$(\lambda\, x_1.\, \lambda\, x_2.\, \cdots\, \lambda\, x_n.\, M)$$

ou então, simplesmente, por

$$(\lambda\, x_1\, x_2\, \cdots\, x_n.\, M)$$

Resumidamente, pode-se afirmar que a linguagem **$\boldsymbol{\lambda}$-Calculus** tem duas construções sintácticas fundamentais

Aplicação $M\,N$

Abstracção $(\lambda\, x.\, M)$

O significado destas duas construções pode ser definido por igualdades que traduzem a relação entre a aplicação e a abstracção traduzida na intuição expressa em (3).

Para tal é necessário representar uma outra noção, a **substituição**, para a qual usaremos a seguinte notação

NOTAÇÃO 1.3 *A expressão-λ que resulta da substituição, em M, da variável x por uma outra expressão-λ, N, usa-se a notação*

$$M[N/x]$$

ou, equivalentemente,

$$M[x := N]$$

Neste curso usaremos, indistintamente, ambas as notações com alguma preferência pela primeira delas.

A relação entre abstracção e aplicação enunciada em (3) pode ser escrita

$$(\lambda\, x.\, M)\, N \;=\; M[N/x] \tag{4}$$

Veremos, ao longo deste curso, que esta relação traduz a noção essencial de **computação** em todo o processo computacional representado pelo **$\boldsymbol{\lambda}$-Calculus**.

A regra (4) é designada por **regra da conversão-β**.

Comparemos agora, por exemplo, as duas abstracções seguintes

$$\lambda\, x.+1\, x \quad , \quad \lambda\, y.+1\, y \tag{5}$$

Intuitivamente estas duas expressões-λ devem representar a <u>mesma</u> transformação: a "função" que

a cada x associa $(+1\, x)$

deve ser a mesma "função" que

a cada y associa $(+1\, y)$

O *nome* da variável de abstracção não deve ser relevante á determinação da expressão-λ que denota esta transformação.

Uma consequência fundamental da operação de abstracção é, portanto, o de "*tornar anónimas*" as variáveis usadas na abstracção. Deve existir uma igualdade que traduza esta equivalência e essa igualdade pode ser escrita da seguinte forma

$$(\lambda\, x.\, M) \;\equiv\; (\lambda\, y.\, M[y/x]) \tag{6}$$

Ou seja a expressão-λ que se obtém abstraindo x em M é a mesma expressão que se obtém se abstrairmos y depois de termos substituído x por y em M.

Uma regra da forma de (6) é designada por **regra de conversão-α**.

É importante salientar uma diferença essencial entre as igualdades (4) e (6).

Na igualdade (4) comparam-se duas expressões-λ distintas mas que têm o mesmo significado. Na igualdade (6) não temos expressões-λ distintas: temos a mesma expressão em ambos os lados da equação.

Para salientar esta diferença usamos dois símbolos diferentes para representar a igualdade.

O símbolo $\equiv$ é usada para comparar **sintácticamente** duas entidades: quando ocorre $M \equiv N$ diremos que temos duas formas **sintácticamente iguais** da mesma expressão-λ.

O símbolo $=$ é usado para comparar **semanticamente** expressões-λ: quando ocorre $M = N$ diremos que temos duas expressões-λ **semanticamente iguais**.

Exemplo 1.3

Alguns exemplos de aplicações da regra de conversão-β e de conversão-α

(i)
$$\begin{aligned}(\lambda\, x.\, +1\, x)\, y &= \\ &= (+1\, x)[y/x] \\ &\equiv +1\, y\end{aligned}$$

(ii)
$$\begin{aligned}(\lambda\, x.\, +1\, x)\,(+1\, y) &= \\ &= (+1\, x)[(+1\, y)/x] \\ &\equiv +1\,(+1\, y)\end{aligned}$$

(iii)
$$\begin{aligned}(\lambda\, x.\, +1\, x) &\equiv (\lambda\, a.\, (+1\, x)[a/x] \\ &\equiv (\lambda\, a.\, +1\, a)\end{aligned}$$

$$
\begin{array}{llcl}
\text{(iv)} & (\lambda\, x.\, (\lambda\, a.\, +\, a\, x))\, y\, b & = & \\
 & & = & (\lambda\, a.\, +\, a\, x)[y/x]\, b \\
 & & \equiv & (\lambda\, a.\, +\, a\, y)\, b \\
 & & = & (+\, a\, y)[b/a] \;\equiv\; (+\, b\, y)
\end{array}
$$

Nestes exemplos é crucial perceber a razão porque, nuns passos de prova, se usa o símbolo de igualdade sintáctica $\equiv$ e nos outros passos de prova se usa o símbolo da igualdade semântica $=$.

1.3 Igualdade extensional

Uma última igualdade definidora do significado conjunto da aplicação e da abstracção é a que traduz a noção de *igualdade extensional* de uma transformação.

Na Teoria dos Conjuntos a igualdade definida entre duas funções $f(\,.\,)$ e $g(\,.\,)$, é extensional.

Isto significa que as funções são consideradas iguais se e só se, quando aplicadas aos mesmos argumentos, produzem os mesmos resultados. Ou seja

$$f = g \quad \text{sse} \quad f(x) = g(x) \quad \forall x \tag{7}$$

Na linguagem do $\boldsymbol{\lambda}$-Calculus a igualdade extensional é expressa por uma regra de inferência da forma

$$\frac{M\,x = N\,x}{M = N} \tag{8}$$

sendo x uma variável arbitrária.

Na premissa o facto de x ser uma variável (e, portanto, sujeita a quaisquer substituições) estabelece que as aplicações $M\,x$ e $N\,x$ são arbitrárias e percorrem todas as possíveis aplicações das transformações M e N.

O facto de estas aplicações arbitrárias serem semanticamente iguais (para o mesmo x) leva-nos a concluir que se verifica a igualdade entre as duas transformações.

Uma regra da forma de (8) designa-se por **regra de conversão-$\boldsymbol{\eta}$**.

1.4 Variáveis livres e ligadas

Um aspecto muito importante da abstracção é o facto de "*tornar anónimas*" as variáveis aí usadas.

No **$\boldsymbol{\lambda}$**-Calculus as variáveis que intervêm numa operação de abstracção dizem-se **ligadas** a essa operação. Inversamente as variáveis que não são usadas em quaisquer abstracções dizem-se **livres**.

Essencialmente o nome de uma variável ligada é irrelevante à definição da expressão-λ onde se insere; este é o sentido da igualdade (6). São excepção a esta regra os casos onde pode existir ambiguidade entre duas variáveis ligadas.

As variáveis ligadas, no contexto da expressão global onde se inserem, não podem ser objecto de substituições. Apenas as variáveis livres podem ser substituídas.

Exemplo 1.4

Considere-se a expressão-λ α definida por

$$\alpha \doteq (\lambda\, x.\, +\, x\, y)$$

Nesta expressão-λ a variável x é ligada e a variável y é livre.

É possível (sem alterar a expressão) mudar o nome de x para qualquer outra variável excepto y.

De facto α é uma expressão-λ diferente de $(\lambda\, y.\, +\, y\, y)$.

É possível substituir, em α , a variável y por qualquer outra expressão-λ.

Porém já não é possível efectuar qualquer substituição sobre x. Note-se que se quiser calcular, por exemplo,

$$\alpha[N/x]$$

o facto de o nome x ser susceptível de ser alterada para outro nome (que não y) tira qualquer sentido a uma eventual substituição.

É importante notar-se que o facto de uma variável estar livre ou ligada depende da expressão em que está incluída. Por exemplo, na expressão

$$+\, 1\, x$$

a variável x é livre. Porém vendo-a como parte da expressão

$$(\lambda\, x.\, +\, 1\, x)$$

é já uma variável ligada.

Exemplo 1.5

Alguns exemplos de variáveis livres e ligadas

Expressão	Var. Livres	Var. Ligadas
$+\,x\,y$	$x\,,\,y$	$-$
$(\lambda\, x.\, +\,x\,y)$	y	x
$(\lambda\, y.\, (\lambda\, x.\, +\,x\,y))$	$-$	$x\,,\,y$
$(\lambda\, x.\, x)\, y$	y	x
$(\lambda\, x.\, x)\, x$	x	x **Nota:** O mesmo símbolo denota aqui duas variáveis distintas

---

2 Sistema λ. Sintaxe e semântica

O estudo formal do λ-calculus é feito através da introdução de sistemas cada vez mais complexos quer em relação á sintaxe dos termos quer á dos tipos. Como sempre convém iniciar o nosso estudo pela forma mais simples possível do λ-calculus.

Essa forma será obviamente sem tipos (em inglês é conhecida por "type-free λ-calculus") e onde, portanto, a aplicação é livre.

Também não vamos, para já, estabelecer qualquer distinção entre variáveis e constantes: em princípio todo o símbolo pode ser sujeito a substituições. Deste modo temos apenas um alfabeto gerador (o conjunto das variáveis) e não dois como vimos na secção anterior (variáveis + constantes).

A um tal sistema chamaremos sistema λ. A definição formal da sua sintaxe pode ser expressa de várias formas. A primeira será

DEFINIÇÃO 2.1 *Dado um conjunto V de entidades designadas como* variáveis*, o conjunto das expressões-λ gerado por V, representado por Λ_V, é o menor conjunto que satisfaz as seguintes condições.*

(i) $V \subseteq \Lambda_V$*; ou seja, toda a variável* $x \in V$ *é um elemento de* Λ_V.

(ii) Se $s \in \Lambda_V$ *e* $t \in \Lambda_V$, *então* $(s\,t) \in \Lambda_V$.

(iii) Se $x \in V$ *e* $s \in \Lambda_V$, *então* $(\lambda\, x.\, s) \in \Lambda_V$.

Uma notação compacta para a definição sintáctica desta sintaxe abstracta, recorre ao tipo de representação conhecido como BNF ("Backus-Naur Form") usada extensivamente na representação da sintaxe de linguagens de programação. Neste caso teríamos

$$\Lambda_V \ ::= \ V \ \mid \ \Lambda_V \Lambda_V \ \mid \ \lambda\, V.\, \Lambda_V \tag{9}$$

Uma terceira alternativa para definição desta mesma sintaxe recorre a um sistema dedutivo onde as proposições são asserções da forma $s \in \Lambda_V$. Cada uma das condições da definição 2.1 transforma-se numa regra de inferência.

$$\begin{aligned}
& & x \in \Lambda_V & \qquad \forall x \in V & \text{(10a)}\\
[s \in \Lambda_V\,;\, t \in \Lambda_V] & \implies & (s\,t) \in \Lambda_V & & \text{(10b)}\\
[x \in V\,;\, s \in \Lambda_V] & \implies & (\lambda\, x.\, s) \in \Lambda_V & & \text{(10c)}
\end{aligned}$$

A semântica do sistema λ requer uma definição de substituição onde a noção de variável livre e a noção de variável ligada são frequentemente referidas. Para uma definição formal destes conceitos é conveniente definir duas aplicações[1]

$$\mathfrak{L} : \Lambda_V \to \wp(V) \qquad \mathfrak{B} : \Lambda_V \to \wp(V)$$

A primeira associa a cada λ-termo s o conjunto de todas as suas variáveis livres; análoga mente a segunda aplicação associa a cada λ-termo o conjunto das suas variáveis ligadas ("bounded"em inglês). Estas funções são definidas pelas seguintes regras

Variáveis Livres

$$\begin{aligned}
\mathfrak{L}(x) & \doteq \{x\} & \qquad \forall x \in V & \qquad \text{(11a)}\\
\mathfrak{L}(s\,t) & \doteq \mathfrak{L}(s) \cup \mathfrak{L}(t) & & \qquad \text{(11b)}\\
\mathfrak{L}(\lambda\, x.\, s) & \doteq \mathfrak{L}(s) \setminus \{x\} & & \qquad \text{(11c)}
\end{aligned}$$

Variáveis Ligadas

$$\begin{aligned}
\mathfrak{B}(x) & \doteq \{\,\} & \qquad \forall x \in V & \qquad \text{(12a)}\\
\mathfrak{B}(s\,t) & \doteq \mathfrak{B}(s) \cup \mathfrak{B}(t) & & \qquad \text{(12b)}\\
\mathfrak{B}(\lambda\, x.\, s) & \doteq \mathfrak{B}(s) \cup \{x\} & & \qquad \text{(12c)}
\end{aligned}$$

[1] Neste curso, $\wp(X)$ denota o conjunto *partes de* X; isto é, o conjunto de todos os subconjuntos de X.

Exemplo 2.1

Alguns exemplos do uso das definições (11) e (12).

$$
\begin{array}{llcll}
\text{(i)} & \mathfrak{L}(\lambda\, xy.\, x) & = & \mathfrak{L}(\lambda\, y.\, x) \setminus \{x\} & \\
 & & = & \mathfrak{L}(x) \setminus \{y\} \setminus \{x\} & \\
 & & = & \{x\} \setminus \{y\} \setminus \{x\} & = \{\,\} \\
 & & & & \\
 & \mathfrak{B}(\lambda\, xy.\, x) & = & \mathfrak{B}(\lambda\, y.\, x) \cup \{x\} & \\
 & & = & \mathfrak{B}(x) \cup \{y\} \cup \{x\} & \\
 & & = & \{\,\} \cup \{y\} \cup \{x\} & = \{y, x\} \\
 & & & & \\
\text{(ii)} & \mathfrak{L}((\lambda\, x.\, yx)x) & = & \mathfrak{L}(\lambda\, x.\, yx) \cup \mathfrak{L}(x) & \\
 & & = & \mathfrak{L}(yx) \setminus \{x\} \cup \{x\} & \\
 & & = & \{y, x\} \setminus \{x\} \cup \{x\} & = \{y, x\} \\
 & & & & \\
 & \mathfrak{B}((\lambda\, x.\, yx)x) & = & \mathfrak{B}(\lambda\, x.\, yx) \cup \mathfrak{B}(x) & \\
 & & = & \mathfrak{B}(yx) \cup \{x\} \cup \{x\} & \\
 & & = & \mathfrak{B}(y) \cup \mathfrak{B}(x) \cup \{x\} & \\
 & & = & \{\,\} \cup \{\,\} \cup \{x\} & = \{x\}
\end{array}
$$

Note-se como, no exemplo (ii), o mesmo símbolo x é usado para denotar tanto uma variável livre como uma variável ligada.

No exemplo 1.3 foram apresentados alguns casos de substituições que incidiram, essencialmente, na substituição de uma variável x em expressões-λ geradas só por aplicações.

Quando os λ-termos são gerados também por abstracções a noção de substituição torna-se, necessariamente, mais complexa.

Nomeadamente, a substituição em causa, $s\,[t/x]$ só deve alterar s se a variável x fôr livre em s; portanto a noção de substituição tem de ser tal que se verifique um resultado do tipo

$$s\,[t/x] \;=\; s \qquad\qquad \text{Se } x \notin \mathfrak{L}(s) \tag{13}$$

Por exemplo, tomando s como sendo a expressão no caso (ii) do exemplo 2.1, onde x denota simultaneamente uma variável livre e uma variável ligada, apenas a primeira deve estar sujeita a substituição. Portanto deve-se verificar

$$((\lambda\, x.\, yx)x)[t/x] \;=\; ((\lambda\, x.\, yx)[t/x])\,(x[t/x])$$

Pela propriedade 13, dado que $x \notin \mathfrak{L}(\lambda\, x.\, yx)$, deve-se verificar

$$(\lambda\, x.\, yx)[t/x] \;=\; \lambda\, x.\, yx$$

Portanto

$$((\lambda\, x.\, yx)x)[t/x] \;=\; (\lambda\, x.\, yx)\, t \;=\; y\, t$$

A definição formal de substituição, capaz de lidar com todos os termos, será

DEFINIÇÃO 2.2 *A substituição de uma variável x por um termo t num termo s define-se, para as diferentes formas possíveis de s, segundo os seguintes casos:*

$$\begin{array}{llcll}
(1) & x\,[t/x] & \doteq & t & \\
(2) & y\,[t/x] & \doteq & y & \textit{Se } y \neq x \\
(3) & (s\, s')[t/x] & \doteq & (s\,[t/x])\,(s'\,[t/x]) & \\
(4) & (\lambda\, x.\, s)[t/x] & \doteq & \lambda\, x.\, s & \\
(5) & (\lambda\, y.\, s)[t/x] & \doteq & (\lambda\, y.\, s[t/x]) & \textit{Se } x \notin \mathfrak{L}(s) \;\; \textit{ou} \;\; y \notin \mathfrak{L}(t) \\
(6) & (\lambda\, y.\, s)[t/x] & \doteq & (\lambda\, a.\, s\,[a/y][t/x]) & \textit{Se } x \in \mathfrak{L}(s) \;\; \textit{e} \;\; y \in \mathfrak{L}(t) \\
 & & & \textit{para qualquer } a \notin \mathfrak{L}(s) \cup \mathfrak{L}(t) &
\end{array}$$

A necessidade de um cuidado especial na definição da substituição em λ-termos gerados por abstracção está ilustrada no exemplo seguinte.

EXEMPLO 2.2

Tomemos

$$s \;\equiv\; \lambda\, y.\, xy \qquad\qquad t \;\equiv\; y$$

Uma substituição $s[t/x]$ usando a regra (5) daria

$$(\lambda\, y.\, xy)[y/x] \;\equiv\; \lambda\, y.\, yy$$

que, claramente, não coincide com o resultado previsto.

A definição 2.2 diz-nos que o caso aplicável aqui deveria ser (6), dado verificar-se $x \in \mathfrak{L}(xy)$ e $y \in \mathfrak{L}(y)$; para isso deve-se escolher uma nova variável a que não exista livre em s ou t. Assim, usando (6),

$$(\lambda\, y.\, xy)[y/x] \;\equiv\; \lambda\, a.\, xy[a/y][y/x] \;\equiv\; \lambda\, a.\, ya$$

LEMA 2.1 *Se $x \notin \mathfrak{L}(s)$ então $s\,[t/x] \;\equiv\; s$.*

Prova:

Por indução estrutural nas formas de substituição em s

(1) Se $s \equiv x$ a condição do lema não se aplica já que se verifica $x \in \mathfrak{L}(s)$.

(2) Se $s \equiv y$ e $y \neq x$ então o lema verifica-se trivialmente.

(3) Se $x \notin \mathfrak{L}(s\,s')$ então verifica-se $x \notin \mathfrak{L}(s)$ e $x \notin \mathfrak{L}(s')$. Pela hipótese da indução teremos

$$s\,[t/x] \equiv s \qquad \text{e} \qquad s'\,[t/x] \equiv s'$$

Portanto

$$(s\,s')[t/x] \;\equiv\; (s[t/x])\,(s'[t/x]) \;\equiv\; s\,s'$$

(4) Se $s \equiv (\lambda\, x.\, s')$ então, pela forma (4) da definição, $s[t/x] \equiv s$.

(5) Se $x \notin \mathfrak{L}(\lambda\, y.\, s)$ então $x \notin \mathfrak{L}(s)$. Isto significa que a substituição deve recorrer ao caso (5); por outro lado, pela hipótese da indução verifica-se $s[t/x] \equiv s$. Consequentemente

$$(\lambda\, y.\, s)[t/x] \;\equiv\; \lambda\, y.\, s[t/x] \;\equiv\; \lambda\, y.\, s$$

(6) Este caso não se aplica (como vimos no caso anterior).

$\square$

Lema 2.2 (Lema da Substituição)

$$s\,[u/x][v/y] \;\equiv\; s\,[v/y][u[v/y]/x]$$

Prova:

Por indução estrutural nas formas de substituição em s.

$\square$

2.1 Semântica do $\boldsymbol{\lambda}$-Calculus

O significado do λ-calculus reside essencialmente nas várias interpretações dadas à noção de *igualdade* entre termos.

A primeira forma de igualdade é a *igualdade definicional* associada ao símbolo $\doteq$. Essencialmente uma asserção da forma

$$s \;\doteq\; \langle \mathit{exp} \rangle$$

exprime uma *definição* do símbolo s: em todas as ocorrências futuras, o símbolo s é sintácticamente equivalente a $\langle expr \rangle$.

Em seguida podemos referir à *igualdade sintáctica* que está, normalmente, associada ao símbolo $\equiv$.

Em princípio verifica-se

$$M \equiv N$$

se e só se M e N são definidos por, exactamente, os mesmos símbolos.

No entanto a definição 2.2 já foi forçada a introduzir uma extensão deste conceito de igualdade. De facto, no caso (6), apresentou-se uma definição de $(\lambda\, y.\, s)[t/x]$ onde ocorre uma variável a que pode ser qualquer.

Esta definição só pode fazer sentido se estiver estabelecida a equivalência sintáctica entre quaisquer duas expressões definidas por diferentes variáveis a. Isto significa que tem de ser válida sintácticamente a equivalência-α referida em (6); assim é necessário olhar para a equivalência-α como uma extensão natural da igualdade sintáctica.

Pode-se definir formalmente igualdade sintáctica através de um sistema de dedução formado por duas regras apenas. A primeira das regras exprime a igualdade em termos de coincidência sintáctica e a segunda exprime a equivalência-α.

$$\text{(id)} \qquad M \equiv M$$

$$(\alpha) \qquad (\lambda\, x.\, M) \equiv (\lambda\, y.\, M[y/x]) \qquad \text{para todo} \quad y \notin \mathfrak{L}(M)$$

A *igualdade semântica* é denotada pelo símbolo =. Assim a asserção

$$M = N$$

quer afirmar que os termos M e N têm o mesmo significado independentemente do facto de eventualmente não serem representados pelos mesmos símbolos.

A razão porque duas expressões sintácticamente diferentes podem ter o mesmo significado reside essencialmente na operação de substituição que, como vimos atrás, implementa a aplicação de uma função a um argumento. Esse princípio aparece expresso na regra de redução-β que vimos em (4).

Assim, a redução-β vai ser a regra fundamental de um sistema dedutivo que caracterize a igualdade semântica. Não vai, porém, ser única.

Para definir a igualdade $=$ é preciso estabelecer que ela satisfaz as propriedades de uma relação de equivalência (reflexividade, simetria e transitividade).

Adicionalmente deve existir algo que exprima as propriedades ligadas á aplicação e á abstracção.

Para tal recorre-se a duas regras:

- uma regra referente á aplicação, afirmando que a aplicação de duas funções equivalentes a argumentos equivalentes deve produzir resultados equivalentes,
- uma segunda regra referente á abstracção afirmando que a abstracção, pela mesma variável, de duas expressões equivalentes deve produzir funções equivalentes.

A primeira regra reflecte a chamada *propriedade da congruência* enquanto a segunda reflecte a chamada *propriedade da extensão fraca.*

Podemos agora apresentar todas estas propriedades (incluindo as definidoras da igualdade sintáctica) como regras de inferência de um sistema dedutivo definidor do **Sistema** λ (tabela 1).

$$
\begin{array}{rl}
(\mathrm{id}) & M \equiv M \\
(\alpha) & (\lambda\, x.\, M) \equiv (\lambda\, y.\, M[y/x]) \\
& \qquad\qquad \text{para todo} \quad y \notin \mathfrak{L}(M) \\
\\
(\beta) & (\lambda\, x.\, M)\, N = M[N/x] \\
\\
(\mathrm{ref}) & (M \equiv N) \Longrightarrow (M = N) \\
(\mathrm{sim}) & (M = N) \Longrightarrow (N = M) \\
(\mathrm{tra}) & M = N \,\&\, N = P \Longrightarrow (M = P) \\
\\
(\mathrm{con}) & M = N \,\&\, P = Q \Longrightarrow (M\, P = N\, Q) \\
(\xi) & (N = N) \Longrightarrow (\lambda\, x.\, M) = (\lambda\, x.\, N)
\end{array}
$$

Tabela 1: **Sistema** λ

TEOREMA 2.1 (TEOREMA DA SUBSTITUIÇÃO)
Se $s = s'$ *e* $t = t'$ *então* $s[t/x] = s'[t'/x]$.

A prova deste teorema recorre a dois lemas intermédios

LEMA 2.3 *Se* $(s = s')$ *então* $s[t/x] = s'[t/x]$.

Prova:

$$\begin{array}{lll} (s = s') & \Rightarrow & (\lambda\, x.\, s) = (\lambda\, x.\, s') \qquad\qquad (\xi) \\ & \Rightarrow & (\lambda\, x.\, s)t = (\lambda\, x.\, s')t \qquad \text{(con) e (ref)} \\ & \Rightarrow & s[t/x] = s'[t/x] \qquad\qquad\qquad (\beta) \end{array}$$

□

LEMA 2.4 *Se* $(t = t')$ *então* $s[t/x] = s[t'/x]$.

Prova:

Por indução na forma de s.

□

Prova:

Prova do teorema 2.1.

$$\begin{array}{lll} t = t' \;\&\; s = s' & \Rightarrow & s[t/x] = s'[t/x] \qquad \text{(lema 2.3)} \\ & \Rightarrow & s'[t/x] = s'[t'/x] \qquad \text{(lema 2.4)} \\ & \Rightarrow & s[t/x] = s'[t'/x] \qquad\qquad \text{(tra)} \end{array}$$

□

2.2 Redução-η

A semântica das expressões-λ, expressa na relação de igualdade semântica $=$ definida pelas regras atrás indicadas, não consegue capturar algumas "intuições" sobre a noção de igualdade de funções. Por exemplo é costume assumir como válida a seguinte asserção

$$\textit{se } f(x) = g(x)\,,\ \textit{para qualquer } x,\ \textit{então verifica-se } f = g \qquad (14)$$

Esta asserção, representada no sistema λ, tem de representar apropriadamente as "funções" f e g como λ-termos (por exemplo, s e t). Depois tem de ter em atenção o quantificador "qualquer que seja x" expresso ao meta-nível. Para isso basta apenas ter usar uma variável x que não seja livre em nenhum dos λ-termos que representam as funções.

Deste modo, a asserção pode-se representar na seguinte regra chamada **regra da extensão forte**.

$$(\mathbf{ext}) \qquad (M\,x = N\,x) \implies M = N \qquad \forall x \notin \mathfrak{L}(M) \cup \mathfrak{L}(N) \tag{15}$$

É importante mencionar que esta regra não pode ser inferida do conjunto de regras semânticas atrás referidas e que caracterizam o sistema λ. Portanto um sistema do λ-calculus onde seja válida esta regra é já um sistema diferente do sistema λ. é fácil verificar que a regra (**ext**) tem, como consequência imediata, a seguinte regra chamada *regra de redução-η*.

$$(\boldsymbol{\eta}) \qquad (\lambda\, x.\, M\,x) = M \qquad \forall x \notin \mathfrak{L}(M) \tag{16}$$

De facto,

$$\begin{aligned} (\lambda\, a.\, M\,a)\,x &= M[x/a]\,x \qquad \text{por redução } (\beta) \\ &= M\,x \qquad \text{porque } x \notin \mathfrak{L}(M) \end{aligned}$$

Usando (**ext**) conclui-se

$$(\lambda\, a.\, M\,a) = M$$

Inversamente, se admitirmos que é válida a regra ($\boldsymbol{\eta}$), teremos

$$(\lambda\, x.\, M\,x) = M \quad \text{e} \quad (\lambda\, x.\, N\,x) = N$$

Donde

$$\begin{aligned} M\,x = N\,x &\implies (\lambda\, x.\, M\,x) = (\lambda\, x.\, N\,x) \\ &\implies M = N \end{aligned}$$

Como conclusão podemos afirmar que as regras ($\boldsymbol{\eta}$) e (**ext**) equivalem-se: qualquer uma implica a outra.

O sistema onde uma delas (e portanto também a outra) é válida chama-se **sistema** $\lambda\eta$ e está definido na tabela 2

$$
\begin{array}{|lll|}
\hline
 & \{\text{Todas as regras do sistema } \lambda\} \quad + & \\
(\boldsymbol{\eta}) & (\lambda\, x.\, M\, x) = M & \forall x \notin \mathfrak{L}(M) \\
\hline
\end{array}
$$

Tabela 2: **Sistema** $\lambda\eta$

EXEMPLO 2.3

Designam-se por **combinadores** todos os λ-termos sem quaisquer variáveis livres.

Exemplos de combinadores tradicionalmente importantes são

$$
\begin{array}{rclrcl}
\mathbf{S} & \doteq & \lambda\, xyz.\, xz(yz) & \mathbf{K} & \doteq & \lambda\, xy.\, x \\
\mathbf{F} & \doteq & \lambda\, xy.\, y & \mathbf{I} & \doteq & \lambda\, x.\, x
\end{array}
\tag{17}
$$

Alguns exemplos de igualdades semânticas entre expressões-λ construídas com tais combinadores.

$$
\begin{aligned}
\mathbf{SK} &= (\lambda\, xyz.\, xz(yz))\mathbf{K} \\
&= \lambda\, yz.\, \mathbf{K}z(yz) \\
&= \lambda\, yz.\, (\lambda\, ab.\, a)z(yz) \\
&= \lambda\, yz.\, z \; = \; \mathbf{F}
\end{aligned}
$$

Seja X um combinador qualquer

$$
\begin{aligned}
\mathbf{F}X &= (\lambda\, xy.\, y)X \\
&= (\lambda\, y.\, y) \; = \; \mathbf{I}
\end{aligned}
$$

Estas igualdades sugerem que, do quarteto de combinadores atrás apresentados, apenas **S** e **K** são independentes entre si. Os combinadores **I** e **F** podem ser definidos à custa dos dois primeiros.

$$
\mathbf{F} \doteq \mathbf{SK} \tag{18a}
$$

$$
\mathbf{I} \doteq \mathbf{SKK} \tag{18b}
$$

A importância dos combinadores **K** e **F** resultam da sua capacidade de representar valores booleanos verdadeiro e falso, respectivamente, numa implementação do construtor if ... then ... else.

No sistema λ, a expressão funcional

$$\mathsf{if}\ B\ \mathsf{then}\ X\ \mathsf{else}\ Y$$

é implementado por

$$B\,X\,Y$$

desde que **K** implemente verdadeiro e **F** implemente falso. Para se verificar a correcção de tal implementação basta verificar que

$$\begin{array}{lclcl} \mathsf{if\ verdadeiro\ then}\ X\ \mathsf{else}\ Y & \Rrightarrow & \mathbf{K}\,X\,Y & = & X \\ \mathsf{if\ falso\ then}\ X\ \mathsf{else}\ Y & \Rrightarrow & \mathbf{F}\,X\,Y & = & Y \end{array}$$

Outro combinador interessante é o combinador **B** definido por

$$\mathbf{B} \doteq \mathbf{S}(\mathbf{KS})\mathbf{K} \tag{19}$$

Facilmente se verifica (expandindo a definição) que

$$\mathbf{B} = \lambda\, yzx.\, y(zx)$$

O combinador **B** vai representar a composição de funções. De facto, dadas duas quaisquer funções implementadas em λ-termos f e g, definindo

$$h \doteq \mathbf{B}\, f\, g$$

verifica-se

$$h = \lambda\, x.\, f(g\, x)$$

Esta forma de h coincide precisamente com a definição usual de composição de funções: $(f \circ g)(x) \doteq f(g(x))$.

2.3 Coerência

O exemplo anterior justifica que se façam alguns comentários adicionais sobre combinadores e sua importância na noção de *coerência* do sistema λ.

Em toda a teoria em que, tal como acontece no sistema λ, a semântica é determinada por uma noção de igualdade entre termos, põe-se sempre a questão de saber se essa igualdade é (ou não) trivial no sentido em que torna "iguais" entidades que, em princípio, não o deveriam ser.

No caso dos sistema λ e $\lambda\eta$, cuja semântica é determinada pelos sistemas dedutivos representados respectivamente no quadro 1 e no quadro 2, coloca-se a questão de saber se a igualdade definida torna equivalentes entre si *todos* os λ-termos.

Este reparo sugere a seguinte definição

DEFINIÇÃO 2.3 *Um sistema do λ-calculus diz-se* INCOERENTE *se, para todo o par de combinadores X e Y, se verifica $X = Y$.*

Suponhamos que, num determinado sistema do λ-calculus com os combinadores definidos no exemplo 2.3, se verificava a igualdade semântica $\mathbf{S} = \mathbf{K}$. Tomemos um combinador arbitrário X e construamos $\mathbf{SI}(\mathbf{K}X)\mathbf{I}$ expandindo a definição dos vários combinadores.

$$\begin{aligned}\mathbf{SI}(\mathbf{K}X)\mathbf{I} &= (\lambda\, xyz.\, xz(yz))\mathbf{I}(\mathbf{K}X)\mathbf{I} \\ &= (\lambda\, z.\, \mathbf{I}z(\mathbf{K}Xz))\mathbf{I} \\ &= (\lambda\, z.\, zX)\mathbf{I} \;=\; \mathbf{I}X \;=\; X\end{aligned}$$

Construamos, agora, o combinador $\mathbf{KI}(\mathbf{K}X)\mathbf{I}$ que terá de coincidir com $\mathbf{SI}(\mathbf{K}X)\mathbf{I}$ na hipótese de a igualdade $\mathbf{S} = \mathbf{K}$ se verificar. Aqui será

$$\mathbf{KI}(\mathbf{K}X)\mathbf{I} \;=\; \mathbf{I}\,\mathbf{I} \;=\; \mathbf{I}$$

Deste modo concluímos que

$$(\mathbf{S} = \mathbf{K}) \implies (X = \mathbf{I}) \qquad \forall X$$

Uma vez que X é arbitrário, é possível inferir de $(\mathbf{S} = \mathbf{K})$ que todos os combinadores são equivalentes entre si já que todos são semanticamente iguais a $\mathbf{I}$.

Portanto, todo o sistema do λ-calculus onde seja válida a igualdade $(\mathbf{S} = \mathbf{K})$ é necessariamente incoerente.

2.4 Pontos Fixos

Numa teoria de funções derivada da Teoria dos Conjuntos não faz sentido, geralmente, uma aplicação da forma $f\,f$.

Note-se que esta expressão sugere que f é uma função que admite, como argumento, ela própria; se f tiver domínio A e contradomínio B, teremos, por uma lado,[2]

$$f \in B^A$$

[2]Em Teoria dos Conjuntos Y^X é o conjunto de todas as funções de domínio X e contradomínio Y.

e, porque f é também argumento da função,

$$f \in A$$

Sendo f arbitrário seria necessário verificar-se

$$A = B^A$$

o que, em Teoria dos Conjuntos, é uma equação de solução impossível.

No entanto, no sistema λ, uma aplicação $x\,x$ é sempre possível. Considere-se, para um combinador arbitrário F, um par de combinadores W e X definidos por

$$W \doteq \lambda x.\, F(xx) \qquad\qquad X \doteq W\,W$$

Expandindo a definição de X verifica-se

$$X = (\lambda x.\, F(xx))\,W = F(W\,W) = F\,X$$

Esta igualdade sugere que o combinador X é um *ponto fixo* de F. Isto é, X é termo que não é alterado pela aplicação de F; deste modo, qualquer que seja o número n de aplicações de F a X, o resultado será sempre X

$$F(F(\ldots(F\,X))) = X$$

Uma vez que a construção de X depende de W que, por seu turno, depende de F, é possível definir um combinador $\mathbf{Y}$ que traduza essa dependência. Ou seja, existe um combinador, a que daremos o nome $\mathbf{Y}$, tal que $\mathbf{Y}\,F$ produz o ponto fixo X.

Atendendo às definições de W e X teremos

$$X = W\,W = (\lambda x.\, F(xx))(\lambda x.\, F(xx))$$

Definindo

$$\mathbf{Y} \doteq \lambda y.\, (\lambda x.\, y(xx))(\lambda x.\, y(xx)) \tag{20}$$

teremos $\mathbf{Y}F = X$ e, consequentemente, verificar-se-á *para todo o combinador* F

$$F\,(\mathbf{Y}F) = \mathbf{Y}F \tag{21}$$

Por motivos óbvios o combinador $\mathbf{Y}$ chama-se *combinador ponto fixo.* Tal como os restantes combinadores com propriedades especiais, é possível defini-lo a partir dos combinadores base **S** e **K**. A definição é bastante complexa e parte de uma definição prévia de um novo combinador **C**.

$$\mathbf{C} \doteq \mathbf{S}(\mathbf{BS}(\mathbf{BKS}))(\mathbf{KK}) \tag{22a}$$
$$\mathbf{Y} \doteq \mathbf{S}(\mathbf{CB}(\mathbf{SII}))(\mathbf{CB}(\mathbf{SII})) \tag{22b}$$

Como exercício pode-se verificar, expandindo a definição de **Y**, que as definições (22b) e (20) produzem o mesmo combinador.

EXERCÍCIOS

2-1 Se c denotar uma constante e x, y, z denotarem variáveis, indique quais, das seguintes substituições, são legítimas e os respectivos valores.

(a) $x\,y\,[y/x][a/y]$

(b) $x\,y\,[a/y][y/x]$

(c) $x\,y\,[a/y][y/x][a/y]$

(d) $c\,x\,[a/c][c/x]$

(e) $c\,x\,[y/c\,x]$

2-2 Sendo s um λ-termo genérico gerado por aplicações, prove por indução estrutural na forma de s que

$$s\,[u/x][v/y] \equiv s\,[v/y][u[v/y]/x]$$

2-3 Apresente expressões-λ que modelem as seguintes transformações da Teoria dos Conjuntos.

(a) $f\,,\,g \mapsto f \circ g$ (composição de funções)

(b) $f \mapsto f(0)$

2-4 Usando directamente a definição das aplicações $\mathfrak{L}$ e $\mathfrak{B}$ determine o conjunto das variáveis livres e o conjunto das variáveis ligadas de cada uma das seguintes expressões-λ.

(a) $(\lambda\, x.\, xy)y$

(b) $(\lambda\, y.\, xy)y$

(c) $(\lambda\, xy.\, ax(yx))$

(d) $(\lambda\, ay.\, ax(yx))yx$

2-5 Determine o resultado das seguintes substituições justificando cada passo com a regra de substituição usada.

(a) $(\lambda\, x.\, ax)[a/x]$

(b) $(\lambda\, a.\, ax)[a/x]$

(c) $(\lambda\, ax.\, ayx)[ax/y]$

(d) $(x\,(\lambda\, x.\, xy))[x/y][y/x]$

(e) $(x\,(\lambda\, x.\, xy))[y/x][x/y]$

(f) $(x\,(\lambda\, x.\, xy))[y/x][x/y][y/x]$

2-6 Verifique quais, dos seguintes λ-termos são sintácticamente equivalentes entre si.

(a) $(\lambda\, x.\, xy)$	(b) $(\lambda\, y.\, xy)$
(c) $(\lambda\, a.\, ay)$	(d) $(\lambda\, x.\, x)(\lambda\, a.\, ay)$

2-7 Prove que a igualdade sintáctica é uma relação de equivalência (é reflexiva, simétrica e transitiva).

2-8 Prove que a igualdade sintáctica é uma congruência e verifica a propriedade da extensão fraca; ou seja, prove que são teoremas no sistema λ os seguintes juízos

(a) $[s \equiv t\, ;\; u \equiv v] \implies (s\, u \;\equiv\; t\, v)$

(b) $(s \;\equiv\; t) \implies (\lambda\, x.\, s) \;\equiv\; (\lambda\, x.\, t)$

2-9 Usando as regras de inferência que determinam a igualdade semântica no sistema-λ, construa provas por propagação em retrocesso para os seguintes teoremas.

(a) $(\lambda\, y.\, y)\, a \;=\; (\lambda\, x.\, x)\, a$

(b) $(\lambda\, a.\, a)\, y \;=\; (\lambda\, x.\, x)(\lambda\, x.\, x)\, y$

(c) $(\lambda\, abx.\, a(bx))(\lambda\, x.\, x) \;=\; (\lambda\, ab.\, ab)$

2-10 Idem para o sistema-$\lambda\eta$ e para os seguintes teoremas.

(a) $(\lambda\, abx.\, a(bx))(\lambda\, a.\, a) \;=\; (\lambda\, x.\, x)$

(b) $(\lambda\, xyz.\, xyz) \;=\; (\lambda\, a.\, a)$

2-11 Atendendo á definição dos combinadores $\mathbf{S}$, $\mathbf{K}$, $\mathbf{B}$, $\mathbf{F}$, e $\mathbf{C}$ e considerando X um combinador arbitrário, simplifique as seguintes expressões.

(a) $\mathbf{KS}$	(b) $\mathbf{SFF}$
(c) $\mathbf{BKF}$	(d) $\mathbf{B}(\mathbf{K}X)X$
(e) $\mathbf{CB}(\mathbf{SII})$	(f) $\mathbf{S}(\mathbf{CB}(\mathbf{SII}))(\mathbf{CB}(\mathbf{SII}))$

2-12 Verifique que as definições (20) e (22) determinam o mesmo combinador $\mathbf{Y}$.

2-13 Determine, por utilização do combinador $\mathbf{Y}$, pontos fixos dos seguintes combinadores.

(a) $\mathbf{K}$	(b) $\mathbf{B}$
(c) $\mathbf{I}$	(d) $\mathbf{B}\,X$

2-14 Para um combinador arbitrário X e para qualquer $n \geq 0$ define-se X^n indutivamente por

$$X^0 \doteq \mathbf{I} \qquad\qquad X^{n+1} \doteq \lambda\, y.\, X\,(X^n y)$$

Com esta definição os *numerais de Church* são a família de combinadores $\{\mathbf{c}_n\}_0^\infty$ definidos por

$$\mathbf{c}_n \doteq (\lambda\, sz.\, s^n z) \qquad \forall n \geq 0$$

Definam-se ainda dois outros combinadores

$$+ \doteq \lambda\, xyab.\, xa(yab) \qquad\qquad * \doteq \lambda\, xya.\, x(ya)$$

Prove que, $+$ e $*$ implementam a soma e a multiplicação de naturais se estes forem, por seu turno, implementados pelo numerais de Church. Para isso prove, por indução em m, que

(a) $+\,\mathbf{c}_n\mathbf{c}_m = \mathbf{c}_{n+m}$

(b) $(\mathbf{c}_n X)^m Y = X^{n*m} Y$

Finalmente, usando (b), prove

(c) $*\,\mathbf{c}_n\mathbf{c}_m = \mathbf{c}_{n*m}$

3 Redução

Para provar a validade de uma igualdade genérica

$$X = Y$$

no sistema λ, será natural pensar-se em utilizar uma das metodologias genéricas construindo resoluções apropriadas com todas as regras de inferência que constituem este sistema dedutivo.

Porém as metodologias genéricas de prova têm a dificuldade intrínseca de não serem decidíveis. Nomeadamente não existe, em cada estado de prova, um critério bem definido que decida qual a estratégia a adoptar na progressão da mesma: qual a regra que deve ser usada, como deve ser usada, etc.

As formas particulares que têm as regras de inferência do sistema λ e os juízos intervenientes neste tipo de prova (que são sempre igualdades semânticas ou igualdades sintácticas) permitem, no entanto, definir uma metodologia alternativa que tem (ou pretende-se que tenha) a propriedade essencial de ser *decidível.*

A estratégia usual para provar este tipo de igualdade começa por "expandir" ambos os termos X e Y efectuando as aplicações que estão definidas nesses termos.

No final deste primeiro passo, tanto X como Y estão reduzidos a termos onde não existem já quaisquer aplicações implícitas; a igualdade demonstra-se verificando se ambos os termos dão origem à mesma expansão.

$$\begin{tikzcd}[column sep=small] X \arrow[dr] & & Y \arrow[dl] \\ & X' \equiv Y' & \end{tikzcd} \tag{23}$$

Procura-se expandir X usando, essencialmente, a regra de redução-β convertendo-o num termo X' que lhe seja equivalente e que não admita expansões subsequentes; procede-se análogamente para o termo Y construindo um termo Y' que lhe é equivalente. Obtém-se, assim, termos X' e Y' que satisfazem

$$X = X' \qquad\qquad Y = Y'$$

Basta que seja $X' = Y'$ para que, por transitividade, se possa concluir a igualdade desejada. Como, por hipótese, X' e Y' não admitem quaisquer expansões o único modo de poderem ser semanticamente iguais é serem sintácticamente iguais.

Esta metodologia tem estado a ser usada, informalmente, em todas as provas de igualdade já apresentadas. O objectivo desta secção é fundamentalmente estudá-la, apresentando as dificuldades a ela inerentes e as soluções respectivas.

- é preciso ver, em primeiro, lugar o que é que se entende realmente por "expansão"ou "redução". Qual é a sua definição, em que condições é possível e que resultados produz. Está-se, assim, a estudar o problema da *existência* de uma prova construída desta forma: só será possível construir, com esta metodologia, uma prova de $X = Y$ se fôr possível definir uma redução única tanto para X como para Y.

- Em seguida é necessário verificar a *correcção* da metodologia. Para tal é necessário provar que a redução de um termo é semanticamente igual a esse termo. Isso basta para concluir que uma prova por redução, se existir, produz as mesmas conclusões que a prova construída pelo sistema de dedução.

- Finalmente é preciso verificar se a ausência de uma prova por redução implica que também não existe uma prova construída a partir do sistema dedutivo. Está-se a analisar o problema da *completude* da metodologia: a existência de uma prova por redução é necessária para que seja válido $X = Y$.

3.1 A relação $\to_\beta$

Vai-se ilustrar a metodologia de prova por redução num caso concreto: pretende-se provar a validade de

$$\mathbf{SI}(\mathbf{K}X)\mathbf{I} = \mathbf{FI}(\mathbf{K}X)\mathbf{I} \tag{24}$$

Para analisar as possíveis reduções pode-se começar por substituir os primeiros combinadores de cada um dos membros da igualdade

$$\mathbf{SI}(\mathbf{K}X)\mathbf{I} \equiv (\lambda\, xyz.\, xz(yz))\mathbf{I}(\mathbf{K}X)\mathbf{I} \tag{25a}$$
$$\mathbf{FI}(\mathbf{K}X)\mathbf{I} \equiv (\lambda\, xy.\, y)\mathbf{I}(\mathbf{K}X)\mathbf{I} \tag{25b}$$

Em ambos os casos, os termos têm uma forma do tipo

$$(\lambda\, x.\, s_1 \ldots s_n)\, t_1 \ldots t_m$$

que sugere uma expansão baseada na aplicação da regra de *redução-β*.

Para indicar que um termo t provém da expansão de um termo s por *uma* aplicação da regra (β) usaremos a notação

$$s \;\to_\beta\; t$$

e diz-se que s é *β-redutível* a t. Essencialmente, verifica-se a expansão (ou redução) de s em t quando s, ou algum dos seus sub-termos, concorda com o lado esquerdo regra de inferência (β): ou seja, quando algum dos sub-termos de s (ou mesmo o próprio s) é da forma

$$(\lambda\, x.\, u)\, v \tag{26}$$

A regra (β) diz-nos que este termo é semanticamente equivalente a

$$u\,[v/x] \tag{27}$$

Na terminologia do estudo da simplificação de termos, um termo da forma (26) chama-se um *β-redex*. O termo (27) é o respectivo *contractum*.

Assim um termo será β-redutível se e só se contém um *β-redex* como sub-termo. Se não contiver nenhum *β-redex* o termo diz-se uma *forma β-normal* (sumariamente, *fn-β*).

Lema 3.1 *Um λ-termo é uma forma β-normal nas condições seguintes:*

(i) Uma variável x é sempre uma fn-β.

*(ii) Um termo $s\,t$ é uma forma β-normal se e só se ambos os termos s e t são formas β-normais e $s\,t$ não é um β-*redex.

(iii) Um termo $(\lambda\, x.\, s)$ é uma fn-β se e só se s é uma fn-β.

Prova:

Consideremos cada uma das alternativas de definição de λ-termos:

(i) O único sub-termo de uma variável x é a própria variável que, como não coincide com qualquer β-*redex*, é uma fn-β,

(ii) Se algum dos termos s ou t contém um β-*redex* ou $s\,t$ determina um β-*redex* então, necessariamente, $s\,t$ contém um β-*redex* e não é uma forma β-normal.

Inversamente, se s e t não contém qualquer β-*redex* e $s\,t$ não determina um β-*redex*, então $s\,t$ não contém qualquer β-*redex*.

(iii) O conjunto dos sub-termos de $(\lambda\, x.\, s)$ é formado pelo próprio termo $(\lambda\, x.\, s)$ e por todos os sub-termos de s. Como $(\lambda\, x.\, s)$ não é um β-*redex* qualquer eventual β-*redex* de $(\lambda\, x.\, s)$ terá de ser também β-*redex* em s.

$\square$

Para provar a igualdade (24) note-se que os termos em (25) iniciam-se ambos com um β-*redex*. Substituindo esses sub-termos pelos respectivos "contractums" obtém-se

$$\begin{aligned} \mathbf{SI}(\mathbf{K}X)\mathbf{I} &\to_\beta (\lambda\, yz.\, \mathbf{I}z(yz))(\mathbf{K}X)\mathbf{I} \\ \mathbf{FI}(\mathbf{K}X)\mathbf{I} &\to_\beta (\lambda\, y.\, y)(\mathbf{K}X)\mathbf{I} \end{aligned}$$

Mais uma vez, ambos os termos iniciam-se com um β-*redex*; são, portanto, redutíveis e dão origem a

$$\begin{aligned} (\lambda\, yz.\, \mathbf{I}z(yz))(\mathbf{K}X)\mathbf{I} &\to_\beta (\lambda\, z.\, \mathbf{I}z(\mathbf{K}Xz))\mathbf{I} \\ (\lambda\, y.\, y)(\mathbf{K}X)\mathbf{I} &\to_\beta (\mathbf{K}X)\mathbf{I} \end{aligned}$$

Continuando a estratégia de, sistematicamente, substituir o β-*redex* “mais à esquerda” pelo respectivo *contractum*, obtém-se sucessivamente as seguintes reduções

$$\begin{array}{llllll} (\lambda\, z.\, \mathbf{I}z(\mathbf{K}Xz))\mathbf{I} & \to_\beta & \mathbf{II}(\mathbf{K}X\mathbf{I}) & \to_\beta & \mathbf{I}(\mathbf{K}X\mathbf{I}) \\ & \to_\beta & (\mathbf{K}X\mathbf{I}) & \to_\beta & X \\ \\ (\mathbf{K}X)\mathbf{I} & \to_\beta & X \end{array} \qquad (28)$$

Ambos os membros da equação (24) foram reduzidos ao mesmo termo X que não admite qualquer redução adicional porque não contém qualquer β-*redex*: ambos os membros da igualdade foram reduzidos à mesma forma β-normal. Concluímos, assim, que a equação (24) é válida.

Formalmente, a relação $\to_\beta$ define-se do modo seguinte:

DEFINIÇÃO 3.1 *A relação $\to_\beta$ é uma relação binária em Λ definida indutivamente por:*

(β-1) $(\lambda\, x.\, s)\, t \to_\beta s[t/x]$

(β-2) $(s \to_\beta t) \Longrightarrow (s\, u) \to_\beta (t\, u)$

(β-3) $(s \to_\beta t) \Longrightarrow (u\, s) \to_\beta (u\, t)$

(β-4) $(s \to_\beta t) \Longrightarrow (\lambda\, x.\, s) \to_\beta (\lambda\, x.\, t)$

LEMA 3.2 *No sistema λ, $(s \to_\beta t) \Longrightarrow (s = t)$.*

Prova:

Por indução estrutural na forma de termo β-redutível.

(β-1) Se se têm $(\lambda\, x.\, s)\, t \to_\beta s[t/x]$ então a regra (β) no sistema λ garante que se verifica $(\lambda\, x.\, s)\, t = s[t/x]$.

(β-2) A hipótese da indução estabelece, neste caso, que $s = t$. Usando a regra da congruência no sistema λ, concluímos que se verifica $(s\, u) = (t\, u)$.

(β-3) Prova análoga à anterior.

(β-4) Prova análoga à anterior usando, porém, a regra (ξ) do sistema λ.

□

EXEMPLO 3.1

Nem sempre a satisfação da relação $X \to_\beta Y$ força a que X e Y sejam termos distintos. Consideremos, por exemplo, o combinador designado por Ω e definido por

$$\Omega \doteq (\lambda x.\, xx)(\lambda x.\, xx) \tag{29}$$

Note-se que Ω é um β-*redex* e que pode ser reduzido por aplicação da regra (β-1).

$$\begin{array}{rcl} \Omega & \to_\beta & (xx)[(\lambda x.\, xx)/x] \\ & \equiv & (\lambda x.\, xx)(\lambda x.\, xx) \;\equiv\; \Omega \end{array}$$

Portanto verifica-se

$$\Omega \;\to_\beta\; \Omega$$

3.2 Relações $\twoheadrightarrow_\beta$ e $=_\beta$

O exemplo (28) mostra como é possível encadear sucessivamente várias reduções de modo a reduzir um termo a outro. Este encadeamento sucessivo de β-reduções justifica uma notação própria. Escreveremos

$$s \;\twoheadrightarrow_\beta\; t$$

quando s é β-reduzido em t em zero ou mais passos de redução.

Formalmente define-se a relação $\twoheadrightarrow_\beta$ do modo seguinte.

DEFINIÇÃO 3.2 *A relação $\twoheadrightarrow_\beta$ é uma relação binária em Λ definida indutivamente por:*

(β^*-1) $\quad (s \equiv t) \implies (s \twoheadrightarrow_\beta t)$

(β^*-2) $\quad [s \to_\beta u\,;\, u \twoheadrightarrow_\beta t] \implies (s \twoheadrightarrow_\beta t)$

LEMA 3.3 *No sistema λ, $(s \twoheadrightarrow_\beta t) \implies (s = t)$.*

Prova:

Por indução estrutural nas duas formas definidoras da relação $\twoheadrightarrow_\beta$.

(β^*-1) Se se verifica $s \twoheadrightarrow_\beta t$ como consequência de (β^*-1) então será $s \equiv t$ e, por reflexividade, $s = t$.

(β^*-2) A hipótese de indução assegura $u = t$. O lema 3.2 assegura $s = u$. Por transitividade conclui-se $s = t$.

□

LEMA 3.4 *Se X é uma forma β-normal e se se verifica $X \twoheadrightarrow_\beta Y$ então $X \equiv Y$.*

Prova:

Como X não contém nenhum β-*redex* (por ser uma forma β-normal) não se pode verificar $X \rightarrow_\beta X'$ para nenhum X'. Assim só se verificar $X \twoheadrightarrow_\beta Y$ em consequência da regra (β^*-1); donde $X \equiv Y$.

□

A metodologia da prova por redução, usada em (25), pode ser expressa em uma nova relação de equivalência $=_\beta$, designada por *equivalência-β*, e definida formalmente por

DEFINIÇÃO 3.3 *Diz-se que s é β-equivalente a t, e representa-se por*

$$s \ =_\beta \ t$$

quando existe uma par de termos s' e t' que satisfazem as seguintes condições:

(i) $s' \equiv t'$

(ii) $s \twoheadrightarrow_\beta s'$ e $t \twoheadrightarrow_\beta t'$

Resta agora determinar a relação que existe entre a equivalência-β assim definida e a igualdade semântica no sistema λ. Essa relação está expressa no teorema seguinte.

TEOREMA 3.1
No sistema λ, $(s =_\beta t) \Leftrightarrow (s = t)$

A prova deste teorema recorre a dois lemas prévios. Em primeiro lugar o lema 3.5 garante a correcção da prova por redução: se pela metodologia da redução se conseguir provar que dois termos são equivalentes então, no sistema λ eles são realmente equivalentes.

LEMA 3.5 *No sistema λ, $(s =_\beta t) \implies (s = t)$.*

Prova:

Pelo lema 3.3, conclui-se $s = s'$ e $t = t'$. Pela transitividade no sistema λ conclui-se $s = t$.

$\square$

Para provar o resultado inverso (se s e t são semanticamente equivalentes em λ então são β-equivalentes) recorre-se a um resultado intermédio que caracteriza a relação $=_\beta$.

LEMA 3.6 *A relação* $=_\beta$ *verifica*

(i) As propriedades definidoras de uma relação de equivalência (reflexividade, simetria e transitividade).

(ii) As propriedades da congruência: se se verifica $s =_\beta t$ *então verifica-se* $(s\,u) =_\beta (t\,u)$, $(u\,s) =_\beta (u\,t)$ *e* $(\lambda\,x.\,s) =_\beta (\lambda\,x.\,t)$.

Prova:

(i) Em primeiro lugar vamos verificar que $=_\beta$ é uma relação de equivalência.

(a) *Reflexividade:* $(s \equiv t) \implies (s =_\beta t)$
Dado que, por (β^*-1), $s \twoheadrightarrow_\beta s$ e $t \twoheadrightarrow_\beta t$ e, por hipótese, $s \equiv t$, então $s =_\beta t$ pela definição (3.3).

(b) *Simetria:* $(s =_\beta t) \implies (t =_\beta s)$
Pela simetria da própria definição de $=_\beta$.

(c) *Transitividade:* $[(s =_\beta u)\,;\,(u =_\beta t)] \implies (s =_\beta t)$
As hipóteses $(s =_\beta u)$ e $(u =_\beta t)$ implicam a existência de termos s', u', t' tais que $s \twoheadrightarrow_\beta s'$, $t \twoheadrightarrow_\beta t'$, $s' \equiv u' \equiv t'$. Pela transitividade da equivalência sintáctica temos $s' \equiv t'$ e, consequentemente a definição (3.3) garante que se verifica $s =_\beta t$.

(ii) Em seguida verificam-se as propriedades da congruência.

(a) $(s =_\beta t) \implies (s\,u) =_\beta (t\,u)$
Se $s \twoheadrightarrow_\beta s'$ então (por (β^*-2) e (β-2)) verifica-se $(s\,u) \twoheadrightarrow_\beta (s'\,u)$; pelos mesmos argumentos, $t \twoheadrightarrow_\beta t'$ implica $(t\,u) \twoheadrightarrow_\beta (t'\,u)$. Dado que $s' \equiv t'$ implica $(s'\,u) \equiv (t'\,u)$, conclui-se que existem termos $(s'\,u)$ e $(t'\,u)$ que estão nas condições impostas pela definição (3.3) e que garantem $(s\,u) =_\beta (t\,u)$.

(b) $(s =_\beta t) \implies (u\,s) =_\beta (u\,t)$ e $(s =_\beta t) \implies (\lambda\,x.\,s) =_\beta (\lambda\,x.\,t)$
Provas análogas à anterior.

□

Prova:

Do teorema 3.1.

O resultado $(s =_\beta t) \Rightarrow (s = t)$ deriva do lema 3.5. O resultado inverso $(s = t) \Rightarrow (s =_\beta t)$ prova-se por indução estrutural nas regras que, no sistema λ, definem a igualdade semântica (ver quadro 1 na página 15).

(β) A igualdade $(\lambda\, x.s)t = s\,[t/x]$ entre um β-*redex* e o seu *contractum* origina sempre $(\lambda\, x.\, s)t =_\beta s\,[t/x]$ dado que, pela regra (β-1), se verifica $(\lambda\, x.\, s)t \to_\beta s\,[t/x]$ e portanto $(\lambda\, x.\, s)t \twoheadrightarrow_\beta s\,[t/x]$.

(ref) Se $s = t$ se verificar porque $s \equiv t$ então, porque a relação $=_\beta$ é uma relação de equivalência (lema 3.6), também se verifica $s =_\beta t$.

(sim) Se $s = t$ se verifica porque $t = s$ se verifica então, admitindo a hipótese de indução $t =_\beta s$, concluímos $s =_\beta t$ pela simetria da relação $=_\beta$.

- Para igualdades $s = t$ geradas por qualquer das restantes regras – (tra), (con) e (ξ) – a prova é semelhante: cada uma das hipóteses $s = t$ dá origem a uma hipótese de indução $s =_\beta t$ e as propriedades de $=_\beta$ determinadas no lema 3.6 determinam o resultado pretendido.

□

3.3 Confluência

Recordemos que o objectivo da prova por redução é a substituição da prova de igualdade no sistema λ (que é não decidível) por uma prova de igualdade decidível. O teorema 3.1 permite substituir a prova da igualdade $X = Y$ pela prova da igualdade $X =_\beta Y$; no entanto não existe garantia que esta última igualdade seja mais decidível do que a primeira.

De facto a definição (3.3) aponta claramente para a não-decidibilidade da relação $=_\beta$. Note-se que esta definição exige, para garantir a validade de $X =_\beta Y$, que sejam encontrados duas reduções X' e Y', de X e de Y, sintácticamente equivalentes; porém nada é dito sob a forma como esses termos podem ou devem ser construídos.

Em princípio todo o termo pode ter muitas reduções; a definição parece sugerir que deveriam ser construídas todas as reduções de X e todas as reduções de Y e verificar, para cada combinação possível de elementos de um e do outro conjunto, se existe um par de termos sintácticamente equivalentes.

Computacionalmente este mecanismo é, genericamente, indecidível e para esclarecer a razão desta afirmação é conveniente introduzir alguns conceitos e resultados adicionais ligados à noção de redução-β.

DEFINIÇÃO 3.4 *Uma* CADEIA DE REDUÇÃO-β *é uma sequência (finita ou infinita) de λ-termos* $\omega \doteq \{X_n\}$ *tais que*

$$X_0 \to_\beta X_1 \to_\beta X_2 \to_\beta \ldots \to_\beta X_n \to_\beta \ldots$$

Formalmente: $X_{n-1} \to_\beta X_n \quad n = 1, 2, \ldots.$

(i) Se se verificar $X \equiv X_0$ *então diz-se que* X *é* RAIZ *de* ω. *e escreve-se*

$$X \twoheadrightarrow \omega$$

(ii) Numa cadeia finita o último termo X_n *designa-se por* OBJECTIVO *da cadeia e a sua ordem* n *é o* COMPRIMENTO *da cadeia.*

(iii) Uma cadeia é MAXIMAL *se é finita e o seu objectivo é uma forma* β*-normal.*

Representa-se por $[\,.\,]$ a aplicação que associa uma cadeia finita ao seu objectivo, e uma cadeia infinita ao "pseudo-termo" $\perp$.

$$[\omega] \doteq \begin{cases} \perp & \text{se } \omega \text{ é infinita} \\ X_n & \text{se } \omega \text{ é finita e } X_n \text{ é o seu objectivo} \end{cases} \tag{30}$$

Note-se que $\perp$ não é um λ-termo gerado pela sintaxe do λ-calculus; é simplesmente um símbolo que permite usar a mesma notação para conceitos semelhantes em cadeias infinitas e em cadeias finitas. Assim $\perp$ pode ser interpretado informalmente como um "pseudo-objectivo no infinito" comum a todas as cadeias infinitas.

A introdução de $\perp$ permite estender a definição da relação $\twoheadrightarrow_\beta$ (definição (3.2)) de modo a incluir reduções infinitas.

DEFINIÇÃO 3.5 *Dado um λ-termo X, é válido*

$$X \twoheadrightarrow_\beta \perp$$

quando existe ω *tal que* $X \twoheadrightarrow \omega$ *e* $[\omega] \equiv \perp$.

LEMA 3.7 *Dada uma cadeia de redução* $\omega \doteq \{X_n\}$ *e um termo* X *raiz de* ω *verifica-se:*

(i) $X \twoheadrightarrow_\beta X_m$ *para todo* $m = 0, 1, \ldots.$

(ii) $X \twoheadrightarrow_\beta [\omega]$

(iii) Se X *é uma forma* β*-normal então o comprimento da cadeia é* 0.

Prova:

(i) Vamos provar, por indução linear em n que

$$X_{m-n} \twoheadrightarrow_\beta X_m \qquad \forall m \forall n \leq m$$

Para $n = 0$ temos $X_m \twoheadrightarrow_\beta X_m$ que é válido pela regra (β^*-1). Partindo da hipótese de indução $X_{m-n} \twoheadrightarrow_\beta X_m$, e dado que a definição de cadeia de redução garante que se verifica $X_{m-n-1} \rightarrow_\beta X_{m-n}$, concluímos pela regra (β^*-2) que se verifica $X_{m-n-1} \twoheadrightarrow_\beta X_m$. Por indução verifica-se então $X_{m-n} \twoheadrightarrow_\beta X_m$ para todo $n \leq m$.

Dado que a hipótese $X \twoheadrightarrow \omega$ implica $X \equiv X_0 \equiv X_{m-m}$ concluímos que também se verifica $X \twoheadrightarrow_\beta X_m$.

(ii) Se a cadeia for infinita então, por definição, temos $X \twoheadrightarrow_\beta \perp$ e $\perp \equiv [\omega]$. Se a cadeia for finita então, pelo resultado anterior, $X \twoheadrightarrow_\beta X_m$ para qualquer elemento X_m da cadeia e, em particular, para o último elemento.

(iii) Por (i) $X \twoheadrightarrow_\beta X_m$ verifica-se para todo o elemento da cadeia. Por outro lado, pelo lema 3.1, dado que X é uma forma β-normal temos $X \equiv X_m$ para todo m e, portanto, todos os elementos da cadeia são formas β-normais. Nessas circunstâncias não é possível verificar a condição $X_{n-1} \rightarrow_\beta X_n$ $\forall n = 1, 2, \ldots$ a menos que a cadeia tenha apenas um elemento: X_0.

$\square$

é possível construir cadeias de redução combinando, por concatenação, outras cadeias. De facto

DEFINIÇÃO 3.6 *Sejam* ω *e* ω' *duas cadeias de redução em que a primeira é necessariamente finita e a segunda verifica*

$$[\omega] \twoheadrightarrow \omega'$$

Teremos, assim,

$$\omega \doteq \{X_i\}_{i=0}^{n} \qquad \omega' \doteq \{X'_j\}_{j=0}^{m} \quad (m \text{ pode ser } \infty) \qquad X_n \equiv X'_0$$

Define-se $\omega\,\omega'$ *como a sequência de* λ*-termos de elementos* $\{Y_i\}_{i=0}^{\infty}$ *definidos por*

$$Y_i \doteq \begin{cases} X_i & se \quad i \leq n \\ X'_{i-n} & se \quad i \geq n \end{cases}$$

LEMA 3.8 *Se* ω *e* ω' *são cadeias de redução, se* ω *é finita e se se verifica* $[\omega] \twoheadrightarrow \omega'$ *então*

1. $\omega\,\omega'$ *é uma cadeia de redução.*
2. *Se* $X \twoheadrightarrow \omega$ *então* $X \twoheadrightarrow \omega\,\omega'$.
3. $[\omega'] \equiv [\omega\,\omega']$
4. *Se* ω' *for finito então* $\omega\omega'$ *é finito e o seu comprimento é igual à soma dos comprimentos de* ω *e* ω'.

Prova:

(i) Para $i \leq n$ temos $Y_{i-1} \rightarrow_\beta Y_i$ já que $Y_i \equiv X_i$ e a sequência $\{X_i\}$ forma uma cadeia de redução. Para $i > n$ temos também $Y_{i-1} \rightarrow_\beta Y_i$ porque $Y_i \equiv X'_{n-i}$ e os $\{X'_j\}$ formam uma cadeia de redução.

(ii) $X \twoheadrightarrow \omega$ implica $X \equiv X_0$; como, por definição, se verifica $X_0 \equiv Y_0$ conclui-se $X \twoheadrightarrow \omega\omega'$.

(iii) Se ω' é finita e m é o seu comprimento então, pela definição, o último termo de $\omega\omega'$ é o termo $Y_{n+m} \equiv X'_m$. Se ω' não é finita também $\omega\omega'$ não será finita. Em qualquer dos casos verifica-se $[\omega\,\omega'] \equiv [\omega']$.

(iv) Ver (iii).

$\square$

PROPOSIÇÃO 3.1 *Se* ω *é uma cadeia maximal e* $\omega\omega'$ *existe então*

$$\omega\omega' \equiv \omega$$

Prova:

Se ω é maximal então, pela definição, $[\omega]$ é uma forma β-normal. Se $\omega\omega'$ existe então verifica-se $[\omega] \twoheadrightarrow \omega'$. Usando o lema 3.7 conclui-se que ω' tem comprimento 0. Portanto $\omega\omega'$ tem de coincidir com ω.

$\square$

DEFINIÇÃO 3.7 *Dado uma cadeia de redução finita ω diz-se que ela admite uma* EXTENSÃO MAXIMAL *quando existe uma outra cadeia ω' tal que $\omega\omega'$ é uma cadeia maximal. A cadeia $\omega\omega'$ é uma extensão maximal de ω.*

EXEMPLO 3.2

A construção de cadeias de redução denota o processo computacional essencial quando se procura aplicar a definição da equivalência $=_\beta$ para provar a equivalência semântica entre dois termos.

Tomemos, por exemplo, o termo $\mathbf{SI}(\mathbf{K}X)\mathbf{I}$ e as reduções apresentadas na página 27. Essas reduções sugerem uma cadeia de redução ω_1 dada por

$$\begin{array}{rl} \mathbf{SI}(\mathbf{K}X)\mathbf{I} & \to_\beta \;(\lambda\, yz.\, \mathbf{I}z(yz))(\mathbf{K}X)\mathbf{I} \;\to_\beta\; (\lambda\, z.\, \mathbf{I}z(\mathbf{K}Xz))\mathbf{I} \;\to_\beta \\ & \to_\beta \;\mathbf{II}(\mathbf{K}X\mathbf{I}) \;\to_\beta\; \mathbf{I}(\mathbf{K}X\mathbf{I}) \;\to_\beta\; \mathbf{K}X\mathbf{I} \;\to_\beta\; X \end{array}$$

Esta cadeia ω_1 é finita, tem comprimento 6 (note-se que, pela definição, o primeiro elemento da sequência tem o índice 0 e o último tem um índice igual ao comprimento) e tem o termo X como objectivo. Temos assim

$$\mathbf{SI}(\mathbf{K}X)\mathbf{I} \twoheadrightarrow \omega_1 \qquad [\omega_1] \equiv X \qquad \mathbf{SI}(\mathbf{K}X)\mathbf{I} \twoheadrightarrow_\beta X$$

O termo X é, aqui, uma variável. Portanto, não contém qualquer β-*redex*; sendo $X \equiv [\omega_1]$ uma forma β-normal podemos afirmar que ω_1 é uma cadeia de redução maximal.

A cadeia ω_1 é extensão maximal de todas as suas sub-cadeias.

A cadeia ω_1 do exemplo anterior não é a única cadeia de redução que é possível construir com a mesma raiz $\mathbf{SI}(\mathbf{K}X)\mathbf{I}$. Ela foi construída usando a estratégia que consiste em substituir, sistematicamente, o β-*redex* "mais à esquerda" pelo seu *contractum*. Essencialmente aplicou-se sempre a redução-β mais externa.

Esta estratégia corresponde ao que, na terminologia da programação funcional, se chama "lazy evaluation".

Alternativamente poder-se-ia usar uma outra estratégia que consistia em sistematicamente substituir o β-*redex* "mais interior". Isso corresponde a simplificar primeiro os argumentos de uma função antes de aplicar a função a esses argumentos.

Na terminologia da programação funcional essa estratégia chama-se "eager evaluation". Usando a mesma raiz pode-se construir uma nova cadeia como se ilustra no seguinte exemplo.

Exemplo 3.3

Vai-se construir uma nova cadeia ω_2 com a mesma raiz $\mathbf{SI}(\mathbf{K}X)\mathbf{I}$ identificando (sublinhando) o $\boldsymbol{\beta}$-*redex* substituído em cada redução.

A estratégia de redução aqui usada consiste em escolher, em cada passo, o $\boldsymbol{\beta}$-*redex* "mais interior".

$$\begin{aligned} \underline{\mathbf{SI}}(\mathbf{K}X)\mathbf{I} \;&\to_\beta\; (\lambda\, yz.\, \underline{\mathbf{I}z}(yz))(\mathbf{K}X)\mathbf{I} \;\to_\beta\; (\lambda\, yz.\, z(yz))\underline{(\mathbf{K}X)}\mathbf{I} \\ &\to_\beta\; (\underline{(\lambda\, yz.\, z(yz))(\lambda\, y.\, X)})\mathbf{I} \;\to_\beta\; (\lambda\, z.\, z(\underline{(\lambda\, y.\, X)z}))\mathbf{I} \\ &\to_\beta\; \underline{(\lambda\, z.\, zX)\mathbf{I}} \;\to_\beta\; \underline{\mathbf{I}X} \;\to_\beta\; X \end{aligned}$$

Note-se que o objectivo da cadeia ω_2 assim construída é precisamente o mesmo que o da cadeia ω_1 definida no exemplo (3.2). Não se trata de uma coincidência: resulta, como veremos adiante, do facto de serem ambas cadeias maximais com a mesma raiz.

Interessa precisar o significado das designações "mais exterior" e "mais interior" aplicados ao $\boldsymbol{\beta}$-*redex* escolhido em cada passo da construção das cadeias ω_1 e ω_2 determinadas nos dois últimos exemplos. Para identificar os pontos comuns e as diferenças nas estratégias que conduziram a estas duas cadeias de redução é necessário um resultado intermédio e alguns conceitos adicionais.

Lema 3.9 *Todo o termo $s \in \Lambda$ pode ser escrito na forma*

$$\lambda\, x_1 \,\ldots\, x_n.\, u\, t_1 \,\ldots\, t_m \qquad n, m \geq 0$$

com $t_1\,, t_2\,, \ldots, t_m \in \Lambda$ e em que u assume uma de duas formas alternativas:

(i) Uma variável y que pode ou não coincidir com alguma das variáveis $x_1\,, x_2\,, \ldots, x_n$.

*(ii) Um $\boldsymbol{\beta}$-*redex $(\lambda\, y.\, s_0)\, t_0$.

Prova:

Em Barendregt (1992, lema 2.3.21, pag. 143).

□

DEFINIÇÃO 3.8 *O sub-termo u no lema 3.9 assume, em relação ao termo global, o nome de* FACTOR PRINCIPAL. *Os termos t_1, t_2, ..., t_m chamam-se* FACTORES NÃO-PRINCIPAIS. *Ao conjunto de todos os factores chamamos* FACTORIZAÇÃO.
No caso de o factor principal ser uma variável ela designa-se por VARIÁVEL PRINCIPAL. *As variáveis x_1, x_2, ..., x_n são* VARIÁVEIS DE ABSTRACÇÃO PRINCIPAIS.
No caso de o factor principal ser um $\boldsymbol{\beta}$-redex *será designado por* $\boldsymbol{\beta}$*-redex* PRINCIPAL.

Note-se que a factorização do termo s pode não conter quaisquer variáveis de abstracção principais (n pode ser 0). Nesse caso o termo escreve-se, simplesmente,

$$u\, t_1 \,\ldots\, t_m$$

Pode também acontecer que não existam factores não principais (m pode ser 0). Nesse caso o termo será

$$\lambda\, x_1, \ldots, x_n.\, u$$

Finalmente temos o caso mais simples de todos em que não existem variáveis de abstracção principais nem outros factores que não o principal. O termo reduz-se então ao factor principal

$$u$$

Note-se que a ausência de variáveis de abstracção principais não implica que não existam outras variáveis de abstracção. Por exemplo, o termo

$$(\lambda\, x.\, x)(\lambda\, x.\, x)$$

reduz-se a um factor principal que tem a forma de um $\boldsymbol{\beta}$*-redex*. Neste termo existem duas variáveis de abstracção mas nenhuma é principal.

A diferença entre as estratégias que conduziram às cadeias ω_1 e ω_2 exprime-se no critério que, em cada passo de redução, selecciona o $\boldsymbol{\beta}$*-redex* que deve ser substituído pelo respectivo *contractum*.

Dado um termo s factorizado de acordo com a forma geral referida no lema 3.9,

$$\lambda\, x_1 \,\ldots\, x_n.\, u\, t_1 \,\ldots\, t_m \tag{31}$$

a escolha do $\boldsymbol{\beta}$-*redex* "mais externo" (característica da *lazy evaluation*) selecciona u no caso de ser um $\boldsymbol{\beta}$-*redex*.

Se o factor principal for uma variável então é seleccionado, usando a mesma metodologia, o $\boldsymbol{\beta}$-*redex* "mais externo" do primeiro factor não principal t_i que não é uma forma β-normal.

A escolha do $\boldsymbol{\beta}$-*redex* "mais interno" (característica do *eager evaluation*) só selecciona u, quando u for $\boldsymbol{\beta}$-*redex* $(\lambda\, y.\, s_0)t_0$ e tanto s_0 como t_0 forem formas β-normais. Se em $u \equiv (\lambda\, y.\, s_0)t_0$ os termos s_0 ou t_0 não forem formas normais, é seleccionado o $\boldsymbol{\beta}$-*redex* "mais interno" (usando a mesma metodologia) de s_0 ou (caso s_0 seja uma forma β-normal) de t_0.

Se o factor principal u for uma variável então, tal como na estratégia anterior, será seleccionado com a mesma metodologia o $\boldsymbol{\beta}$-*redex* mais interno do primeiro factor não principal t_i que não seja uma forma β-normal.

Para definir com precisão estas duas estratégias vamos introduzir uma função parcial

$$\underline{\;.\;} : \Lambda \to \Lambda$$

que associa a cada termo $s \in \Lambda$ o $\boldsymbol{\beta}$-*redex* escolhido de acordo com a estratégia que se pretende definir. A função é parcial porque existem termos (as formas β-normais) que não contém qualquer $\boldsymbol{\beta}$-*redex*.

Lazy Evaluation

$$\begin{array}{rcll}
\underline{x} & = & \bot & \text{para toda a variável } x \\
\underline{(\lambda\, x.\, s)} & = & \underline{s} & \\
\underline{s\, t} & = & (s\, t) & \text{se } (s\, t) \text{ for um } \boldsymbol{\beta}\text{-}\textit{redex} \\
\underline{s\, t} & = & \underline{s} \oplus \underline{t} & \text{se } (s\, t) \text{ não for um } \boldsymbol{\beta}\text{-}\textit{redex}
\end{array}$$

Em que o operador escolha $\oplus$ é definido pelas seguintes regras

$$\bot \oplus \bot = \bot, \qquad \bot \oplus s = s, \qquad s \oplus s' = s \qquad\qquad s, s' \in \Lambda \tag{32}$$

Eager Evaluation

$$
\begin{array}{rcll}
\underline{x} & = & \perp & \text{para toda a variável } x \\
\underline{(\lambda\, x.\, s)} & = & \underline{s} & \\
\underline{s\, t} & = & (s\, t) & \text{se } (s\, t) \text{ for um } \boldsymbol{\beta}\textit{-redex} \text{ e tanto } s \text{ como } t \text{ forem fn-}\beta \\
\underline{s\, t} & = & \underline{s} \oplus \underline{t} & \text{se } (s\, t) \text{ não for um } \boldsymbol{\beta}\textit{-redex} \text{ ou } \underline{s} \neq \perp \text{ ou } \underline{t} \neq \perp
\end{array}
$$

Estas duas estratégias básicas podem ser combinadas produzindo estratégias mais complexas; na construção de uma cadeia de redução pode ser usada a "lazy evaluation" em alguns dos passos e a "eager evaluation" noutros. No entanto existe um resultado que garante uma certa uniformidade nos objectivos das várias cadeias que assim podem ser construídas.

TEOREMA 3.2 (CHURCH-ROSSER)
Para todo $X \in \Lambda$, se existem $Y_1, Y_2 \in \Lambda$ tais que se verifica $X \twoheadrightarrow_\beta Y_1$ e $X \twoheadrightarrow_\beta Y_2$ então existe Y tal que se verifica

$$Y_1 \twoheadrightarrow_\beta Y \qquad e \qquad Y_2 \twoheadrightarrow_\beta Y$$

Prova:

Ver Barendregt (1992, Teorema 2.3.7, pag. 136).

□

A propriedade do termo X traduzida por este teorema pode ser graficamente representada no seguinte diagrama. A forma particular do mesmo sugere a designação de *propriedade do diamante* ou *propriedade do losângulo.*

$$
\begin{tikzcd}
 & X \arrow[dl, "\beta"'] \arrow[dr, "\beta"] & \\
Y_1 \arrow[dr, dotted, "\beta"'] & & Y_2 \arrow[dl, dotted, "\beta"] \\
 & Y &
\end{tikzcd}
\tag{33}
$$

DEFINIÇÃO 3.9 *Um termo X que verifique a propriedade do losângulo para uma relação genérica $\mathcal{R}$ (em vez da relação $\twoheadrightarrow_\beta$) diz-se* CONFLUENTE *para $\mathcal{R}$.*

O teorema de Church-Rosser tem, assim, uma forma alternativa:

TEOREMA 3.3 (CHURCH-ROSSER (ALT.))
No sistema λ todo o termo X é confluente para a relação $\twoheadrightarrow_\beta$.

Corolário 3.1 *Se $X \in \Lambda$ é raiz de cadeias de redução maximais ω_1 e ω_2 então*

$$[\omega_1] \equiv [\omega_2]$$

Prova:

Pelo lema 3.7 verifica-se $X \twoheadrightarrow_\beta [\omega_1]$ e $X \twoheadrightarrow_\beta [\omega_2]$. Pelo teorema 3.2, existe um termo Y tal que

$$[\omega_1] \twoheadrightarrow_\beta Y \qquad \text{e} \qquad [\omega_2] \twoheadrightarrow_\beta Y$$

No entanto, como as cadeias são maximais, tanto $[\omega_1]$ como $[\omega_2]$ são formas β-normais. Donde, pelo lema 3.4, $[\omega_1] \twoheadrightarrow_\beta Y$ implica $\omega_1 \equiv Y$ e $[\omega_2] \twoheadrightarrow_\beta Y$ implica $\omega_2 \equiv Y$. Consequentemente verifica-se $[\omega_1] \equiv Y \equiv [\omega_2]$.

$\square$

Essencialmente este corolário afirma que duas cadeias maximais que tenham a mesma raiz têm necessariamente o mesmo objectivo. Uma outra consequência muito importante do Teorema de Church-Rosser é o facto de um termo arbitrário só poder ser semanticamente equivalente a uma única forma β-normal (a menos da equivalência sintáctica). Este resultado pode ser expresso do modo seguinte.

Corolário 3.2 *Se se verifica $X = Y_1$ e $X = Y_2$ e se, adicionalmente, Y_1 e Y_2 são formas β-normais então verifica-se $Y_1 \equiv Y_2$.*

Prova:

Pelo lema 3.5, verifica-se $X =_\beta Y_1$ e $X =_\beta Y_2$. Pela definição de $=_\beta$ (definição 3.3) devem existir termos X_1', X_2', Y_1', Y_2' tais que

* $X \twoheadrightarrow_\beta X_1', \qquad Y_1 \twoheadrightarrow_\beta Y_1', \qquad X_1' \equiv Y_1'.$
* $X \twoheadrightarrow_\beta X_2', \qquad Y_2 \twoheadrightarrow_\beta Y_2', \qquad X_2' \equiv Y_2'.$

Dado que Y_1 e Y_2 são formas β-normais terá de se verificar

$$Y_1 \equiv Y_1' \equiv X_1' \qquad \text{e} \qquad Y_2 \equiv Y_2' \equiv X_2'$$

Consequentemente verifica-se

$$X \twoheadrightarrow_\beta Y_1 \qquad \text{e} \qquad X \twoheadrightarrow_\beta Y_2$$

Usando mais uma vez o Teorema de Church-Rosser e o facto de que Y_1 e Y_2 são formas β-normais conclui-se (usando os argumentos apresentados no corolário anterior) que será

$$Y_1 \equiv Y_2$$

$\square$

O corolário assegura que cada termo é semanticamente equivalente, quanto muito, a uma única forma β-normal. Podem no entanto existir termos que não são equivalentes a nenhuma dessas formas: é o caso do combinador $\Omega \doteq (\lambda\, x.xx)(\lambda\, x.xx)$.

Pode também acontecer que o mesmo termo seja raiz de uma cadeia maximal, o que significa que é semanticamente equivalente ao objectivo da cadeia, e simultaneamente ser também raiz de uma cadeia de redução infinita.

é o caso do combinador $\mathbf{K}\,\mathbf{I}\,\Omega$. Usando uma estratégia de redução "lazy" e sublinhando o ***β**-redex* escolhido em cada passo constrói-se a seguinte cadeia

$$\mathbf{K}\,\mathbf{I}\,\Omega \;\equiv\; \underline{(\lambda\, xy.\, x)\mathbf{I}}\Omega \;\to_\beta\; \underline{(\lambda\, y.\, \mathbf{I})\Omega} \;\to_\beta\; \mathbf{I}$$

Deste modo concluímos que se verifica $\mathbf{K}\,\mathbf{I}\,\Omega \twoheadrightarrow_\beta \mathbf{I}$ e, consequentemente, $\mathbf{KI}\Omega = \mathbf{I}$.

Por outro lado, usando a estratégia "eager", construi-se uma redução infinita

$$\mathbf{K}\,\mathbf{I}\,\Omega \;\equiv\; (\lambda\, xy.\, x)\mathbf{I}\Omega \;\to_\beta\; (\lambda\, y.\, \mathbf{I})\Omega \;\to_\beta\; (\lambda\, y.\, \mathbf{I})\Omega \;\to_\beta\; \ldots$$

é importante caracterizar os termos que não podem originar reduções infinitas. Para isso define-se

DEFINIÇÃO 3.10 *Um λ-termo diz-se* NORMALIZÁVEL *quando todas as cadeias que têm o termo como raiz admitem uma extensão maximal.*

Note-se que, como caso particular de cadeia de raiz X existe a cadeia vazia formada apenas pelo elemento X. Se X for normalizável esta cadeia admite uma extensão maximal cujo destino é uma forma β-normal semanticamente equivalente a X. Portanto uma forma normalizável é, pelo menos, equivalente semanticamente a uma forma β-normal.

O corolário 3.2 garante que essa fn-β é a única (a menos de uma equivalência sintáctica) que é equivalente a X. A definição seguinte faz, portanto, sentido

DEFINIÇÃO 3.11 *Dado um λ-termo normalizável X, representa-se por $\overline{X}$ a forma β-normal que é objectivo comum a todas as cadeias maximais de raiz X. Ao termo $\overline{X}$ chama-se* NORMALIZAÇÃO *de X.*

O combinador Ω não é normalizável mas já o combinador $\mathbf{K\,I}\,\Omega$ é normalizável. De facto qualquer que seja a cadeia de raiz nesse termo é sempre possível continuá-la de modo a se construir uma cadeia que termine; por exemplo a cadeia

$$\mathbf{K\,I}\,\Omega \;\to_\beta\; (\lambda\, y.\, \mathbf{I})\Omega \;\to_\beta\; (\lambda\, y.\, \mathbf{I})\Omega$$

pode ser continuada usando a estratégia "lazy" no $\boldsymbol{\beta}$-*redex* final.

$$\mathbf{K\,I}\,\Omega \;\to_\beta\; (\lambda\, y.\, \mathbf{I})\Omega \;\to_\beta\; (\lambda\, y.\, \mathbf{I})\Omega \;\to_\beta\; \mathbf{I}$$

De facto pode-se representar as várias extensões maximais deste exemplo no diagrama seguinte

$$\begin{tikzcd}
\mathbf{K\,I}\,\Omega \arrow[r, two heads, "\beta"] & (\lambda\, y.\, \mathbf{I})\,\Omega \arrow[r, two heads, "\beta"] \arrow[drr, two heads, "\beta"'] & (\lambda\, y.\, \mathbf{I})\,\Omega \arrow[r, two heads, "\beta"] \arrow[dr, two heads, "\beta"] & (\lambda\, y.\, \mathbf{I})\,\Omega \arrow[r, two heads, "\beta"] \arrow[d, two heads, "\beta"] & \cdots \\
& & & \mathbf{I} &
\end{tikzcd} \tag{34}$$

Portanto $\mathbf{I}$ é a normalização de $\mathbf{K\,I}\,\Omega$.

Definição 3.12 *Um termo é* fortemente normalizável *quando é normalizável e, adicionalmente, todas as cadeias de redução de raiz X terminam.*

Face a esta definição conclui-se que $\mathbf{K\,I}\,\Omega$, apesar de normalizável, não é fortemente normalizável já que é raiz de uma cadeia de redução que não termina.

O objectivo final de todo este estudo da redução está no seguinte resultado.

Teorema 3.4
Sendo $X, Y \in \Lambda$ termos normalizáveis então verifica-se $X = Y$ se e só se $\overline{X} \equiv \overline{Y}$.

Prova:

Se for $X = Y$ então, porque $X = \overline{X}$ e $Y = \overline{Y}$ se verificam sempre, teremos, por transitividade, $X = \overline{Y}$. Pelo corolário 3.2, dado que $\overline{X}$ e $\overline{Y}$ são ambas formas β-normais, terá de ser $\overline{X} \equiv \overline{Y}$.

Inversamente, se se verificar $\overline{X} \equiv \overline{Y}$ então será $\overline{X} = \overline{Y}$ e, por transitividade, será $X = Y$.

$\square$

A vantagem deste teorema sobre a utilização directa da definição de $=_\beta$ e do teorema 3.1 para teste de uma igualdade semântica $X = Y$ reside no facto de ele indicar uma metodologia aparentemente, pelo menos, decidível: basta calcular a normalização de cada um dos termos X e Y (que são únicas) e verificar se esses termos são sintácticamente equivalentes.

No entanto o processo de procura do termo $\overline{X}$ que é normalização de X só é decidível se X for fortemente normalizável. Só neste caso há garantia que, qualquer que seja a estratégia de redução utilizada, se consegue alcançar uma forma β-normal num número finito de passos de redução; uma vez obtido o objectivo da redução, qualquer que tenha sido a cadeia construída, esse objectivo é sempre a normalização pretendida.

Se o termo X for apenas normalizável mas não fortemente normalizável não há garantia que se chegue a alcançar uma forma β-normal porque pé sempre possível "escolher mal"a estratégia e enveredar por uma cadeia infinita.

Mesmo no caso em que o termo X é apenas normalizável existe uma estratégia que garante sempre a possibilidade de se alcançar a normalização.

Teorema 3.5
Se X é normalizável então a estratégia de redução definida pela "lazy evaluation" (ver pag. 39) gera sempre uma cadeia de redução maximal com raiz X.

Prova:

Ver Barendregt (1992, Teorema 2.3.20, pag. 143).

□

EXERCÍCIOS

Nos exercícios seguintes $\mathbf{K}, \mathbf{S}, \mathbf{I}, \mathbf{B}, \ldots$ denotam os combinadores definidos na secção anterior (ver exemplo 2.3).

3-1 Indique quais dos seguintes termos são formas β-normais e, para os que não o forem, enumere todos os ***β**-redex*'s neles contidos.

(a) $(\lambda\, x.\, x)$

(b) $(\lambda\, x.\, x)(\lambda\, x.\, x)$

(c) $(\lambda\, xy.\, x(\lambda\, a.\, a)y)$

(d) $(\lambda\, xyz.\, xy(\mathbf{I}xz))\mathbf{I}$

(e) $\mathbf{SKI}$

(f) $\mathbf{SI}(\mathbf{KI})\mathbf{I}$

3-2 Enumere os termos a que são β-redutíveis cada uma das seguintes formas.

(a) $(\lambda x.\, xx)(\lambda x.\, x)$

(b) $(\lambda xy.\, x)((\lambda ab.\, b)\, x)$

(c) $\mathbf{S}\,(\mathbf{KS})\,\mathbf{K}$

(d) $\mathbf{B}\,\mathbf{S}(\mathbf{BKS})$

3-3 Por indução estrutural na forma β-redutível s prove:

$$s \to_\beta t \implies s[u/x] \to_\beta t[u/x]$$

3-4 Instanciando s com cada um dos seguintes termos enumere todos os termos t que verificam $s \twoheadrightarrow_\beta t$.

(a) $(\lambda x.\, xx)(\lambda x.\, x)$

(b) $(\lambda x.\, xx)(\lambda x.\, xx)$

(c) $\mathbf{BS}(\mathbf{BKS})$

(d) $\mathbf{B}(\mathbf{KI})\mathbf{I}$

3-5 Prove o seguinte resultado:

Se $X =_\beta Y$ se verifica e Y é uma forma β-normal então verifica-se $X \twoheadrightarrow_\beta Y$.

3-6 Usando directamente a definição de $=_\beta$ (definição 3.3) verifique se são válidas as seguintes asserções.

(a) $(\lambda x.\, xx)\mathbf{I} =_\beta \mathbf{I}(\lambda x.\, xx)$

(b) $\mathbf{B}(\mathbf{SK}) =_\beta \mathbf{SK}$

3-7 Definindo $\Omega \doteq (\lambda x.\, xx)(\lambda x.\, xx)$ mostre que:

(a) $\Omega \twoheadrightarrow_\beta X \Leftrightarrow \Omega \equiv X$

(b) $\Omega =_\beta X \Leftrightarrow \Omega \equiv X$

3-8 Determine as cadeias de redução que têm como raiz cada um dos seguintes termos. Verifique quais destas cadeias são maximais. Para as cadeias finitas apresente o comprimento e o objectivo.

(a) $\mathbf{B}(\mathbf{KS})\mathbf{II}$

(b) Ω

(c) $\Omega\Omega$

(d) $\mathbf{B}(\mathbf{KS})\Omega\Omega$

(e) $\mathbf{B}(\mathbf{KS})\mathbf{III}$

3-9 Para cada uma das cadeias finitas determinadas no problema anterior determine as suas possíveis extensões maximais. Verifique se todas as extensões maximais de uma mesmas cadeia têm o mesmo objectivo.

3-10 Para cada um dos termos apresentados no exercício (3-8) apresente (se existirem) as cadeias de redução construídas segundo a estratégia "lazy evaluation" e segundo a estratégia "eager evaluation".

3-11 Determine o factor principal em cada um dos seguintes termos:

(a) Ω

(b) $\Omega\Omega$

(c) $\mathbf{B}(\mathbf{KS})\mathbf{II}$

(d) $(\lambda\, x.\, x\mathbf{S}(yz))$

(e) $(\lambda\, xyz.\, x\mathbf{S}(yz))$

(f) $(\lambda\, xyz.\, x\mathbf{S}(yz))\Omega$

3-12 Usando directamente as definições da página 39 determine o ***β**-redex* escolhido por "lazy evaluation" e por "eager evaluation" em cada um dos termos apresentados no exercício anterior.

3-13 Identifique, de entre os termos referidos no problema anterior, quais os que são normalizáveis e, para esses, apresente a respectiva normalização.

3-14 Quais destes termos são fortemente normalizáveis?

3-15 Use o teorema 3.4 para provar a validade das asserções apresentadas no exercício (3-6).

Referências

Barendregt, H. P. (1992). Lambda calculi with types, *in* S. Abramsky, D. Gabbay & T. S. E. Maibaum (eds), *Handbook of Logic in Computer Science*, Oxford University Press.