

Módulo I:

λ -calculus

MCC - 2ºano

José Bernardo Barros José Carlos Bacelar Almeida
(jbb@di.uminho.pt) (bacelar@di.uminho.pt)

Departamento de Informática
Universidade do Minho

Motivação

O λ -calculus é uma *teoria de funções* desenvolvida nos anos 30 por *Alonzo Church*.

Mesmo se este formalismo é muito anterior às linguagens funcionais modernas, é instrutivo fazer uso do conhecimento que dispomos dessas linguagens (e.g. HASKELL) para introduzir alguns dos conceitos que são a base do λ -calculus.

Consideremos a definição de uma função simples em HASKELL:

```
inc = \x -> (+) x 1
```

A linha apresentada pode ser lida como:

inc = ... — o identificador *inc* é definido como...

$\backslash x -> \dots$ — a **abstração** da **variável** *x* na expressão...

$(+)x1$ — que resulta da **aplicação** da função $(+)$ aos argumentos *x* e 1.^a

Nesta breve incursão pelo HASKELL encontramos todos os ingredientes que constituem a base do λ -calculus. No entanto, devemos manter presente que no HASKELL encontramos numerosos conceitos que não estão presentes no λ -calculus. Como características do HASKELL que não encontramos na teoria do λ -calculus, salientamos: ^b

1. O HASKELL dispõe de um mecanismo de tipos que regula “a boa aplicação” das funções aos seus argumentos.
2. Como linguagem de programação moderna, o HASKELL dispõe de múltiplos mecanismos que não encontramos no λ -calculus (e.g. *concordância de padrões; recursividade explícita*; etc).

^aPara sermos mais rigorosos deveríamos dizer *da aplicação do argumento 1 ao resultado da aplicação da função $(+)$ a x* .

^bPelo menos numa primeira fase do estudo onde nos concentramos no λ -calculus sem tipos.

Sintaxe

O conjunto de λ -termos Λ é dado pela seguinte definição indutiva

- Se $x \in Vars$ então $x \in \Lambda$ (**variável**)
- Se $M, N \in \Lambda$ então $(M N) \in \Lambda$ (**aplicação**)
- Se $x \in Vars$ e $M \in \Lambda$ então $(\lambda x.M) \in \Lambda$ (**abstracção**)

onde $Vars$ é um conjunto contável e infinito: *as variáveis*. Denotamos as variáveis por letras minúsculas e os termos por letras maiúsculas.

Convenções sintácticas:

- *precedência*: considera-se a aplicação com uma maior precedência do que a abstracção.
- *associatividade*: a aplicação é associativa à esquerda.
- *abstracções múltiplas*: podemos representar o termo $\lambda x.(\lambda y.M)$ por $\lambda xy.M$.
- *remoção dos parênteses*: sempre que as regras apresentadas acima permitam reconstruir o termo original, podemos omitir os parênteses.

Exemplos:

$$\begin{aligned}\lambda xy.x y z &\rightsquigarrow \lambda x.(\lambda y.((x y) z)) \\ (\lambda x.y x)zy &\rightsquigarrow (((\lambda x.(y x)) z) y)\end{aligned}$$

variáveis livres e ligadas

O papel exercido por uma variável num λ -termo depende crucialmente de esta se encontrar (ou não) vinculada a um λ . No primeiro caso dizemos que se trata de uma ocorrência de uma variável *ligada* e no segundo *livre*.

variáveis livres: As variáveis livres de um λ -termo são definidas recursivamente como:

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

As **variáveis ligadas** como

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

Termos sem variáveis livres dizem-se **termos fechados** ou **combinadores**.

OBS.: uma variável pode ocorrer simultaneamente no conjunto de variáveis livres e ligadas de um termo (e.g. x em $(\lambda x.y \ x) \ x$).

As variáveis livres são as que mais se aproximam da utilização corrente em matemática: o seu significado depende de uma *valoração* (substituição) que se defina no “contexto de cálculo”. Já as variáveis ligadas devem ser entendidas como *parâmetros* da expressão lambda. Nessa perspectiva, o identificador concreto utilizado numa variável ligada não é importante, i.e. devemos identificar termos como:

$$\lambda x.x \quad \text{e} \quad \lambda y.y$$

Renomeação de variáveis

Para formalizar o diferente comportamento das variáveis livres e ligadas vamos introduzir o conceito de **renomeação de uma variável num termo** (denotamos a renomeação da variável v pela variável x em M por $M[v/x]$)^a

$$\begin{aligned} z[v/x] &= \begin{cases} x & \text{se } z \equiv v \\ z & \text{se } z \not\equiv v \end{cases} \\ (E \ F)[v/x] &= (E[v/x] \ F[v/x]) \\ (\lambda z.E)[v/x] &= \begin{cases} \lambda z.E & \text{se } z \equiv v \\ \lambda z.(E[v/x]) & \text{se } z \not\equiv v \end{cases} \end{aligned}$$

Algumas Propriedades:^b

- $M[x/x] \equiv M$
- Note-se que a renomeação só afecta as ocorrências livres das variáveis (e.g. $((\lambda x.y \ x) \ x)[x/a] \equiv (\lambda x.y \ x) \ a$). De facto, facilmente se obtém que

$$FV(M[v/x]) \equiv \begin{cases} FV(M) & \text{se } v \notin FV(M) \\ (FV(M) \setminus \{v\}) \cup \{x\} & \text{se } v \in FV(M) \end{cases}$$

- Se $x \notin FV(M)$ então $M[v/x]$ é uma operação reversível, i.e.

$$(M[v/x])[x/v] \equiv M$$

Por outro lado, se $v, x \in FV(M)$, $M[v/x]$ identifica variáveis que antes eram distintas (dai que não possamos encontrar uma renomeação que inverta a efectuada).

^aIremos utilizar o símbolo $\equiv$ para denotar a **igualdade sintáctica** entre dois termos. Assim, $x \equiv y$ diz-nos que x e y são a mesma variável.

^bTodas elas demonstráveis por indução simples na estrutura do termo.

α -equivalência

Estamos agora em condições de definir a equivalência pretendida nos termos: a α -equivalência. Intuitivamente dizemos:

Dois termos dizem-se α -equivalentes ($\equiv^\alpha$) quando diferem apenas nas variáveis ligadas.

A formalização deste conceito pode ser conseguida por uma definição indutiva da relação pretendida:

$$\frac{}{x \equiv^\alpha y} \text{ se } x \equiv y$$

$$\frac{M \equiv^\alpha M' \quad N \equiv^\alpha N'}{(M \ N) \equiv^\alpha (M' \ N')}$$

$$\frac{N \equiv^\alpha M[x/y]}{\lambda x.M \equiv^\alpha \lambda y.N} \quad y \notin FV(M) \text{ ou } x \equiv y$$

A regra de abstracção é a responsável por tornar equivalentes termos com diferentes escolhas de identificadores nas variáveis ligadas. Note-se que a condição lateral imposta nessa regra é tal que impede que a abstracção considerada no termo do lado direito da conclusão (λy) “capture” uma variável livre do termo do lado esquerdo ($\lambda x.M$).

À substituição de um termo M por um termo N α -equivalente é normal chamar-se α -conversão.

α -equivalência (cont.)

exemplos:

$$\begin{aligned}\lambda x.x\ y &\stackrel{\alpha}{=} \lambda z.z\ y \\ \lambda xy.x\ (y\ x) &\stackrel{\alpha}{=} \lambda ab.a\ (b\ a) \\ (\lambda x.x\ x)\ (x\ x) &\stackrel{\alpha}{=} (\lambda y.y\ y)\ (x\ x)\end{aligned}$$

mas não:

$$\begin{aligned}\lambda x.x\ y &\not\stackrel{\alpha}{=} \lambda y.y\ y \\ (\lambda x.x\ x)\ (x\ x) &\not\stackrel{\alpha}{=} (\lambda y.y\ y)\ (y\ y)\end{aligned}$$

o problema nestes últimos casos é que a substituição das variáveis ligadas interfere com as variáveis livres: no primeiro caso y é livre no lado esquerdo e passa a ligada no lado direito; no segundo x é simultaneamente ligada e livre no lado esquerdo pelo que só deve ser substituída enquanto ligada.

Algumas Propriedades:^a

- $\stackrel{\alpha}{=}$ é uma relação de equivalência (i.e. reflexiva, transitiva e simétrica)
- Se $M \stackrel{\alpha}{=} N$ então $FV(M) = FV(N)$

OBSERVAÇÃO: PODEMOS ENTENDER A INTRODUÇÃO DA EQUIVALÊNCIA- α COMO UM ARTIFÍCIO TÉCNICO PARA ULTRAPASSAR UMA LIMITAÇÃO DA SINTAXE UTILIZADA (TERMOS DE ATRIBUIR UM IDENTIFICADOR ÀS VARIÁVEIS LIGADAS QUANDO NÃO NOS INTERESSA DISTINGUIR ENTRE DIFERENTES ESCOLHAS DESSE IDENTIFICADOR). ASSIM, TODA A TEORIA SERÁ DESENVOLVIDA “MÓDULO” α -EQUIVALÊNCIA.

^aDemonstráveis por indução nos termos e indução na definição de $\stackrel{\alpha}{=}$. No segundo caso temos oportunidade de fazer uso de propriedades referidas atrás.

β -redex

Se pensarmos numa abstracção como a parametrização da regra de cálculo numa função levanta-se a questão de como *calcular* essa regra quando é instânciado o parâmetro (i.e. a aplicação de uma função a um argumento).

Apelemos mais uma vez ao conhecimento que dispomos da linguagem `HASKELL`: Considere-se a aplicação de uma função a um argumento (e.g. a expressão `(\ x->x+1) 3`). Nós sabemos que o processo de cálculo se processa como:

$$(\lambda x. \rightarrow (+) x 1) 3 \Rightarrow (+) 3 1 \Rightarrow \dots \Rightarrow 4$$

Assim identificamos o resultado da aplicação da função apresentada ao argumento como sendo a *substituição*, na expressão `(+) x 1`, da variável x por 3.

Na notação do λ -calculus, procuramos o significado da *aplicação* de um termo a uma *abstracção* — o que se designa por **β -redex**

$$(\lambda x.M) N \xrightarrow{\beta} M[x \leftarrow N]$$

Dizendo nesse caso que $(\lambda x.M) N$ β -reduz-se em $M[x \leftarrow N]$

$M[x \leftarrow N]$ denota a operação de **substituição**, no termo M , da variável livre x pelo termo N . Esta operação generaliza a operação de *renomeação* definida atrás.

Substituição

A definição da operação de *substituição* encerra algumas subtilezas para garantir que o processo não compromete a natureza das variáveis livres e ligadas.

Sejam M, N termos e x uma variável. A **substituição das ocorrências livres de x em M por N** (que representamos por $M[x \leftarrow N]$ — operação com maior precedência do que a aplicação e também associativa à esquerda) é definida recursivamente por

$$\begin{aligned} z[x \leftarrow N] &= \begin{cases} N & \text{se } z \equiv x \\ z & \text{se } z \not\equiv x \end{cases} \\ (E \ F)[x \leftarrow N] &= (E[x \leftarrow N] \ F[x \leftarrow N]) \\ (\lambda y. E)[x \leftarrow N] &= \lambda z. ((E[y/z])[x \leftarrow N]) \quad \text{, sendo } z \text{ uma } \mathbf{variável nova}^a \end{aligned}$$

*É curioso notar que a definição apresentada não nos fornece “um” resultado concreto — qualquer escolha para a variável nova na clausula da abstracção é possível. A esse respeito note-se que a escolha de diferentes variáveis novas nos conduz a resultados α -equivalentes, pelo que a **definição determina univocamente o resultado módulo α -equivalência**.*

Para enfatizar a necessidade de considerarmos a introdução da variável nova, considere-se o seguinte termo:

$$(\lambda x. y \ x)[y \leftarrow w \ x] \neq \lambda x. (y \ x)[y \leftarrow w \ x] = \dots = \lambda x. w \ x \ x$$

o que faz com que uma ocorrência livre de uma variável (x no termo $(w \ x)$) passe a ligada... (diz-se que é *capturada* pela abstracção). Já se procedermos à correcta manipulação do termo obtemos o resultado esperado:

$$(\lambda x. y \ x)[y \leftarrow w \ x] = (\lambda a. (y \ x)[x/a])[y \leftarrow w \ x] = \lambda a. (y \ a)[y \leftarrow w \ x] = \dots = \lambda a. w \ x \ a$$

Propriedades da Substituição

Facilmente se demonstra que

$$M \stackrel{\alpha}{=} M' \implies M[x \leftarrow N] \stackrel{\alpha}{=} M'[x \leftarrow N]$$

sendo então evidente que a substituição é uma operação definida módulo α -equivalência.

Outras propriedades que resultam directamente da definição e que são sempre utilizadas para simplificar casos particulares na aplicação da substituição:

- Se $x \notin FV(M)$ então $M[x \leftarrow N] \stackrel{\alpha}{=} M$
- $(\lambda x.M)[x \leftarrow N] \stackrel{\alpha}{=} \lambda x.M$
- Se $x \notin FV(N)$ então $(\lambda x.M)[y \leftarrow N] \stackrel{\alpha}{=} \lambda x.(M[y \leftarrow N])$

Propriedades adicionais:

- **Lema da Substituição^a** — Se $x \neq y$ e $x \notin FV(L)$, então

$$M[x \leftarrow N][y \leftarrow L] \stackrel{\alpha}{=} M[y \leftarrow L][x \leftarrow N[y \leftarrow L]]$$

^aEste resultado pode ser demonstrado por indução em M . Dado que trabalhamos módulo α -**equivalência**, existe um argumento que diminui muito o número de casos a considerar: a **convenção das variáveis** — nesta convenção adoptam-se representativos para as classes de equivalência- α em que as variáveis livres e ligadas são sempre distintas.

Redução β (um passo)

Um β -redex é um λ -termo com uma estrutura muito rígida. Considere-se o seguinte termo:

$$x ((\lambda y.y) z)$$

Não se tratando de um β -redex, este termo “contém” um β -redex logo é natural que possamos reduzir esse sub-termo resultando em $(x z)$.

A definição da relação de **β -redução** incorpora esta possibilidade de reduzirmos um qualquer sub-termo do considerado: ^a

$$\frac{}{(\lambda x.M) N \xrightarrow{\beta} M[x \leftarrow N]}$$

$$\frac{M \xrightarrow{\beta} N}{(X M) \xrightarrow{\beta} (X N)} \quad \forall X$$

$$\frac{M \xrightarrow{\beta} N}{(M X) \xrightarrow{\beta} (N X)} \quad \forall X$$

$$\frac{M \xrightarrow{\beta} N}{\lambda x.M \xrightarrow{\beta} \lambda x.N} \quad \forall x, M$$

^aUma relação sobre o conjunto de λ -termos $R \subseteq \Lambda \times \Lambda$ diz-se **compatível com a estrutura dos termos** se

$$(M, N) \in R \quad \Rightarrow \quad \begin{cases} ((\lambda x.M), (\lambda x.N)) \in R \\ ((PM), (PN)) \in R \\ ((MP), (NP)) \in R \end{cases} \quad , \forall P \in \Lambda$$

Assim, a relação de **β -redução** pode ser definida de forma compacta como a menor relação R compatível com a estrutura dos termos em que $((\lambda x.M) N, M[x \leftarrow N]) \in R$

β -redução (cont.)

Quando obtemos N a partir de M por um número finito de reduções β dizemos que $M \xrightarrow{\beta} N$. Formalmente, $\xrightarrow{\beta}$ é o fecho reflexivo e transitivo da relação de redução- β . Ao fecho reflexivo, transitivo e simétrico da redução- β chamamos **equivalência- β** e denominamos por $\stackrel{\beta}{=}$ (ou simplesmente por $=$)^a.

Quando num termo existem vários β -redex, um passo de redução- β conduz-nos a diferentes termos (conforme escolha do redex).

Exemplos:

$$\begin{aligned} (\lambda x. (\lambda y. x \ y) \ x) (\lambda x. x \ x) &\xrightarrow{\beta} (\lambda x. x \ x) (\lambda x. x \ x) \\ (\lambda x. (\lambda y. x \ y) \ x) (\lambda x. x \ x) &\xrightarrow{\beta} (\lambda y. (\lambda x. x \ x) y) (\lambda x. x \ x) \end{aligned}$$

Uma **estratégia de redução** determina qual o redex a ser reduzido quando dispomos de várias alternativas. As mais simples são:

- *leftmost-innermost*: opta pelo primeiro redex (mais à esquerda) mais interior (i.e. que não contém qualquer outro redex como sub-termo) — primeiro exemplo.
- *outermost-leftmost*: opta pelo redex mais exterior que se encontrar mais à esquerda — segundo exemplo.

Quando um λ -termo não contém nenhum β -redex dizemos que se encontra na **forma normal**. Se um termo $M \xrightarrow{\beta} N$ e N está na forma normal diz-se que N é **(um)a forma normal** de M e que o termo M é **normalizável**. Um termo para o qual *toda a estratégia* de redução nos permita obter (um)a forma normal diz-se **fortemente normalizável**.

^aÉ importante que estamos a desenvolver toda a teoria *módulo* α -equivalência. Assim, temos trivialmente que $\lambda x. x = \lambda a. a$.

Propriedades I

Questões:

1. Será que todo o termo dispõe de uma forma normal?
2. Será que um termo pode ter duas formas normais distintas?
3. Quando um termo dispõe de forma normal, será que existe *uma estratégia* que nos permita encontrá-la?

Computação infinita

Existem termos para os quais a sequência de reduções é infinita.

$$\Omega \doteq (\lambda x.x x) \lambda x.x x \rightarrow (\lambda x.x x) \lambda x.x x \rightarrow (\lambda x.x x) \lambda x.x x$$

Logo, Ω não pode dispor de forma normal pelo que se responde negativamente à primeira questão. Por outro lado, facilmente construímos um termo que dispõe de uma forma normal e também de uma sequência infinita de reduções (o que reforça a pertinência da terceira questão)

$$(\lambda xy.y) \Omega (\lambda x.x)$$

(se optarmos sempre por reduzir o redex de Ω somos levados a uma computação infinita mas se reduzirmos o primeiro redex obtemos diretamente $\lambda x.x$ que está na forma normal). Neste caso verificamos que a estratégia *outermost-leftmost* nos permite obter a forma normal, ao contrário da *leftmost-innermost* que nos conduz a sequência infinita de reduções. Este é um resultado que se pode demonstrar:

Se um λ -termo é normalizável então a estratégia *outermost-leftmost* permite obter a forma normal.

pode um termo ter diferentes formas normais?

Uma *visão computacional* do λ -calculus leva-nos a identificar:

redução- β	$\Leftrightarrow$	computação
λ -termo	$\Leftrightarrow$	programa
forma normal	$\Leftrightarrow$	resultado
seq. infinita de reduções	$\Leftrightarrow$	não terminação

Assim, um termo dispor de diferentes formas normais corresponde a um programa poder dar diferentes resultados (ou seja, uma computação *não determinística*).

Propriedade Church-Rosser

Se $M \xrightarrow{\beta} X_1$ e $M \xrightarrow{\beta} X_2$ então existe um termo N tal que $X_1 \xrightarrow{\beta} N$ e $X_2 \xrightarrow{\beta} N$.

corolário: Se N_1 e N_2 são formas normais de M então $N_1 = N_2$.

Extensionalidade

O **princípio da extensionalidade** diz-nos que duas funções são iguais se tiverem o mesmo domínio e produzirem o mesmo resultado para todos os valores desse domínio, i.e.

$$(\forall x \quad . \quad fx = gx) \quad \Rightarrow \quad f = g$$

Sendo o λ -calculus uma *teoria de funções*, é razoável questionarmo-nos se, nessa teoria, também se verifica a *extensionalidade* — ou seja, se para termos F, G

$$(\forall M \quad . \quad FM = GM) \quad \Rightarrow \quad F = G$$

A redução- β , por si só, não satisfaz a extensionalidade, como se verifica analisando a lei anterior para os termos $\lambda x.Ex$ e E

$$(\forall M \quad . \quad (\lambda x.Ex)M = EM) \quad \not\Rightarrow \quad (\lambda x.Ex) = E$$

(o lado esquerdo verifica-se por um passo de redução- β enquanto, no lado direito, nada nos permite derivar a equivalência).

Se, ao λ -calculus, adicionarmos uma nova regra de redução

$$\lambda x.Mx \quad \xrightarrow{\eta} \quad M$$

ficamos com uma teoria que satisfaz o *princípio da extensionalidade*. De facto, considerando termos F, G , se $Fx = Gx$ temos, porque $=$ é compatível com estrutura dos termos, $\lambda x.Fx = \lambda x.Gx$ que η -reduz em $F = G$. É habitual designarmos por λ_η -calculus (ou $\lambda_{\beta\eta}$ -calculus) o λ -calculus com redução- η . Todas principais propriedades do λ -calculus são preservadas com a introdução desta redução.