Lambda *Calculus*

Definição

Prof. Edson Alves

Faculdade UnB Gama

Sumário

Características do cálculo λ

- O cálculo λ (λ *calculus*) pode ser considerado "*a (segunda) menor linguagem de programação do mundo*"
- Ele consiste apenas em uma regra de transformação e um esquema de definição de funções
- Foi proposto do Alonzo Church na década de 1930, como uma maneira de formalizar a noção de computabilidade
- Qualquer função Turing computável pode ser expressa e avaliada através do cálculo λ, e vice-versa
- Ao contrário das máquinas de Turing, o foco está no uso das regras de transformações, sendo mais próximo do software do que do hardware

Cálculo λ

Termos-λ

O conjunto Λ dos termos-λ (ou expressões-λ, ou simplesmente lambdas) é definido por meio de um conjunto de variáveis V através das regras de aplicação e abstração, dadas a seguir:

1. $x \in V \Rightarrow x \in \Lambda$ (expressão)
2. $M, N \in \Lambda \Rightarrow MN \in \Lambda$ (aplicação)
3. $M \in \Lambda, x \in V \Rightarrow \lambda x.M$ (abstração)

Observação: informalmente, a aplicação equivale a avaliação da função M com argumento N, isto é $M(N)$; a abstração corresponde a definição da função $f(x) = M$.

Exemplos de termos-λ

1. O termo-λ mais simples possível é formado por uma única variável (por exemplo, x)
2. A função identidade $\lambda x.x$ é um exemplo de abstração
3. Parêntesis podem ser utilizados para clarificar uma expressão ou para remover ambiguidades
4. O termo $(\lambda x.x)y$ corresponde a aplicação da função identidade ao termo y
5. A aplicação é associativa à esquerda:
$$M_1 M_2 \ldots M_N = (((M_1 M_2) M_3) \ldots M_N)$$
6. O termo $\lambda y.(\lambda x.M)$ equivale a uma função de duas variáveis
7. Uma notação alternativa para o termo anterior é
$$\lambda yx.M = \lambda y.(\lambda x.M)$$
8. A abstração é associativa à direita:
$$\lambda x_1 x_2 \ldots x_N.M = \lambda x_1.(\lambda x_2.(\ldots(\lambda x_N.M)))$$

Variáveis livres e atadas (*bound*)

- A abstração $\lambda x.M$ une (ata, *to bind*) a variável livre x ao termo-λ M
- Uma variável não precedida por um símbolo λ que a une a uma expressão é denominada variável **livre**
- Na expressão
$$\lambda x.xy$$
a variável x é atada e a variável y é livre
- Uma mesma variável pode ser livre e atada em uma mesma expressão. Por exemplo, na expressão
$$(\lambda x.xy)(\lambda y.y)$$
a variável y é livre no termo entre parêntesis à esquerda, e atada no termo entre parêntesis à direita

Substituições

Substituição

A **substituição** de todas as ocorrências da variável livre x em M por N, cuja notação é $M[x := N]$, é definida pelas quatro regras apresentadas a seguir:

i. $x[x := N] = N$
ii. $y[x := N] = y$
iii. $(M_1 M_2)[x := N] = (M_1[x := N])(M_2[x := N])$
iv. $(\lambda y.M_1)[x := N] = \lambda y.(M_1[x := N])$

Exemplos de substituição

1. Exemplo de substituição pela regra 3:

$$((\lambda x.xyz)(\lambda y.xzy))[z := N] = (\lambda x.xyN)(\lambda y.xNy)$$

2. Exemplo de substituição pela regra 4:

$$(\lambda x.xy)[y := N] = \lambda x.xN$$

3. Exemplo de substituição pelas regras 2 e 4:

$$(\lambda x.xy)[z := N] = \lambda x.xy$$

4. A substituição de x por N não se aplica no termo $\lambda x.xy$, pois as substituições só ocorrem em variáveis livres, e x é atada neste termo

Reduções

Axiomas de Redução

1. A conversão-α permite renomear uma variável atada em um termo, evitando colisões de nomes (y não pode ocorrer em M):

$$\lambda x.M \equiv_\alpha \lambda y.(M[x := y])$$

2. A redução-β estabelece uma relação entre aplicação e substituição:

$$(\lambda x.M)N \equiv_\beta M[x := N]$$

3. A conversão-η elimina de redundâncias em expressões cujo propósito é apenas repassar o argumento para um termo:

$$(\lambda x.Mx) \equiv_\eta M,$$

se x não é variável livre em M.

Observação: Se a expressão-λ N pode ser obtida através de sucessivas aplicações dos três axiomas acima ao termo M, escrevemos $M \equiv N$.

Exemplos de aplicação dos axiomas de redução

1. Aplicação da função identidade (redução-β)

$$(\lambda x.x)y \equiv x[x := y] \equiv y$$

2. Aplicação em função de duas variáveis (redução-β):

$$\begin{aligned}(\lambda xy.yx)MN &\equiv (\lambda x.(\lambda y.yx))MN\\ &\equiv ((\lambda y.yx)[x := M])N\\ &\equiv (\lambda y.yM)N\\ &\equiv (yM)[y := N]\\ &\equiv NM\end{aligned}$$

3. Eliminação de redundância (conversão-η):

$$(\lambda x.zyx) \equiv zy$$

Exemplos de aplicação dos axiomas de redução

4. Uso da conversão-α para evitar colisão de nomes, pois a variável y, que irá substituir a variável livre x no termo $\lambda y.yx$, tem mesmo nome que a variável atada y:

$$\begin{aligned}(\lambda x.(\lambda y.xy))y &\equiv (\lambda x.(\lambda z.xz))y\\ &\equiv (\lambda z.xz)[x := y]\\ &\equiv \lambda z.yz\end{aligned}$$

Observe que, sem o uso da conversão-α, a aplicação resultaria em $(\lambda y.yy)$, termo que não é equivalente ao termo obtido.

Exemplos de aplicação dos axiomas de redução

5. Outro exemplo de que demanda o uso de conversão-α:

$$\begin{aligned}(\lambda x.(\lambda y.(x\lambda x.xy)))y &\equiv (\lambda x.(\lambda z.(x\lambda x.xz)))y\\ &\equiv (\lambda z.(x\lambda x.xz))[x := y]\\ &\equiv (\lambda z.(y\lambda x.xz))\end{aligned}$$

Aqui novamente a conversão-α foi usada por que y é uma variável atada na expressão $\lambda y.(x\lambda x.xy)$. Além disso, observe que somente a ocorrência livre de x é substituída, conforme dita a regra de substituição apresentada anteriormente.

Combinadores e Igualdade Extensional

Combinadores

(a) O conjunto das variáveis livres $FV(M)$ de M é definido por

i. $FV(x) = \{x\}$
ii. $FV(MN) = FV(M) \cup FV(N)$
iii. $FV(\lambda x.M) = FV(M) - \{x\}$

(b) M é um termo fechado, ou **combinador**, se $FV(M) = \emptyset$

Igualdade Extensional

Os termos-λ E_1, E_2 são **extensionamente iguais** se, $\forall x \in \Lambda$, $E_1 x \equiv E_2 x$.

Combinadores padrão e base S-K

Combinadores padrão

Os combinadores padrão são enumerados a seguir:

1. $\mathbf{I} \equiv \lambda x.x$
2. $\mathbf{K} \equiv \lambda xy.x$
3. $\mathbf{K}_* \equiv \lambda xy.y$
4. $\mathbf{S} \equiv \lambda xyz.xz(yz)$

Completude da base S-K

Dado um combinador E, é possível gerar um novo combinador C a partir dos combinadores $\mathbf{S}$ e $\mathbf{K}$ de tal modo que E e C são extensionalmente iguais.

Exemplo da completude de S-K

A identidade $\mathbf{I}$ é extensionalmente igual ao combinador $\mathbf{SKK}$:

$$
\begin{aligned}
((\mathbf{SKK})x) &\equiv (\mathbf{SKK}x) \\
&\equiv (\mathbf{K}x(\mathbf{K}x)) \\
&\equiv x \\
&\equiv \mathbf{I}x
\end{aligned}
$$

Referências

1. **BARENDREGT**, Henk; **BARENDSEN**, Erik. *Introduction to Lambda Calculus*, March 2000.
2. **LOCZEWSKI**, Georg P. *A++ and the Lambda Calculus: Principles of Functional Programming*, tredition, 2018.
3. **ROJAS**, Raúl. *A Tutorial Introduction to the Lambda Calculus*, FU Berlin, WS-97/98.
4. Wikipédia. Combinatory logic, acesso em 07/01/2020.
5. Wikipédia. Lambda calculus, acesso em 03/01/2020.