

Cálculo Lambda

Prof. Marcus Vinícius Midená Ramos

Universidade Federal do Vale do São Francisco

14 de setembro de 2019

`marcus.ramos@univasf.edu.br`
`www.univasf.edu.br/~marcus.ramos`

- 1 *Lambda-Calculus and Combinators - An Introduction*
J. R. Hindley and J. P. Seldin
Cambridge University Press, 2008
Capítulos 1, 3, 4 e 5
- 2 *Teoria da Computação: Máquinas Universais e Computabilidade*
T. A. Divério e P. B. Menezes
Bookman, 2011, 3ª edição
Capítulo 8
- 3 http://en.wikipedia.org/wiki/Lambda_calculus
- 4 http://en.wikipedia.org/wiki/Church_encoding
- 5 <http://www.cburch.com/proj/lambda/index.html>

Roteiro

- 1 Introdução
- 2 Linguagem lambda
- 3 Substituições
- 4 Redução- β
- 5 Exercícios
- 6 Numerais de Church
- 7 Booleanos de Church
- 8 Igualdade- β
- 9 Ponto fixo
- 10 Recursão
- 11 Indecidibilidade

Características

Sistema formal para representação de computações.

- ▶ Baseado na definição e aplicação de funções;
- ▶ Funções são anônimas (não estão associadas à identificadores);
- ▶ Funções são tratadas como objetos de ordem mais elevada, podendo ser passados como argumentos e retornados de outras funções;
- ▶ Simplicidade: possui apenas dois “comandos”;
- ▶ Permite a combinação de operadores e funções básicas na geração de operadores mais complexos;

Características

- ▶ Modelo alternativo para a representação de computações (usando funções no lugar de máquinas);
- ▶ Equivalente à Máquina de Turing;
- ▶ Mesmo na versão *pura* (sem constantes), permite a representação de uma ampla gama de operações e tipos de dados, entre números inteiros e variáveis lógicas;
- ▶ Versões não-tipada e tipada.

História

- ▶ Alonzo Church, 1903-1995, Estados Unidos;
- ▶ Inventou o Cálculo Lambda na década de 1930;
- ▶ Resultado das suas investigações acerca dos fundamentos da matemática;
- ▶ Pretendia formalizar a matemática através da noção de funções ao invés da teoria de conjuntos;
- ▶ Apesar de não conseguir sucesso, seu trabalho teve grande impacto em outras áreas, especialmente na computação.

Aplicações

Modelo matemático para:

- ▶ Teoria, especificação e implementação de linguagens de programação baseadas em funções, especialmente as linguagens funcionais;
- ▶ Verificação de programas;
- ▶ Representação de funções computáveis;
- ▶ Teoria da Computabilidade;
- ▶ Teoria de Tipos;
- ▶ Teoria das Provas;
- ▶ Assistentes interativos de provas (ex: Coq).

Foi usado na demonstração da indecidibilidade de diversos problemas da matemática, antes mesmo dos formalismos baseados em máquinas.

Motivação

Considere a expressão $x - y$. Ela pode ser formalizada, na notação matemática usual, através de funções com um único parâmetro:

- ▶ $f(x) = x - y$, ou
- ▶ $g(y) = x - y$.

ou, ainda:

- ▶ $f : x \mapsto x - y$, ou
- ▶ $g : y \mapsto x - y$.

Por exemplo:

- ▶ $f(0) = 0 - y$, ou
- ▶ $f(1) = 1 - y$.

Motivação

Representação dessas funções na linguagem lambda:

- ▶ $f = \lambda x.x - y$, ou
- ▶ $g = \lambda y.x - y$.

A aplicação da função a um argumento é representada pela justaposição da função ao argumento:

- ▶ $(\lambda x.x - y)(0) = 0 - y$, ou
- ▶ $(\lambda x.x - y)(1) = 1 - y$.

Motivação

Funções com múltiplos parâmetros:

- ▶ $h(x, y) = x - y$, ou
- ▶ $k(y, x) = x - y$.

Podem ser representadas na linguagem lambda como:

- ▶ $h = \lambda xy.x - y$, ou
- ▶ $k = \lambda yx.x - y$.

Motivação

Tais funções, no entanto, podem também ser representadas como funções que retornam outras funções como valores:

$$h^* = \lambda x.(\lambda y.(x - y))$$

De fato, para cada a temos:

$$h^*(a) = (\lambda x.(\lambda y.(x - y))(a) = \lambda y.(a - y)$$

Para cada par a e b temos:

$$(h^*(a))(b) = ((\lambda x.(\lambda y.(x - y))(a))(b) = (\lambda y.(a - y))(b) = a - b = h(a, b)$$

Portanto h^* representa h e, de forma geral, todas as funções com múltiplos parâmetros podem ser representadas através da combinação de funções com um único parâmetro.

Funções computáveis

No Cálculo Lambda, diz-se que uma função $F : \mathbb{N} \rightarrow \mathbb{N}$ é computável se e somente se existir uma expressão-lambda f tal que:

$$\forall x, y \in \mathbb{N}, F(x) = y \Leftrightarrow fx =_{\beta} y$$

($=_{\beta}$ é chamada “igualdade beta”, serve para estabelecer a equivalência entre termos de uma equação envolvendo termos lambda e será introduzida mais adiante)

Trata-se apenas de uma das formas possíveis de se definir computabilidade, como é o caso da Máquina de Turing, de outras máquinas, e das funções recursivas. A equivalência desses formalismos foi demonstrada.

Definição

Um λ -termo (também chamado de expressão lambda) é definido de forma indutiva sobre um conjunto de identificadores $\{x, y, z, u, v, \dots\}$ que representam variáveis:

- ▶ Uma variável (também chamada “átomo”) é um λ -termo;
- ▶ Aplicação: se M e N são λ -termos, então (MN) é um λ -termo; representa a aplicação de M a N ;
- ▶ Abstração: se M é um λ -termo e x é uma variável, então $(\lambda x.M)$ é um λ -termo; representa a função que retorna M com o parâmetro x ;

A linguagem lambda é composta por todos os λ -termos que podem ser construídos sobre um certo conjunto de identificadores; trata-se de uma linguagem com apenas dois operadores ou “comandos”: aplicação de função à argumentos (chamada de função) e abstração (definição de função).

Gramática

$$V \rightarrow u | v | x | y | z | w | \dots$$

$$T \rightarrow V$$

$$T \rightarrow (TT)$$

$$T \rightarrow (\lambda V.T)$$

Exemplos

São exemplos de λ -termos:

- ▶ x
- ▶ (xy)
- ▶ $(\lambda x.(xy))$
- ▶ $((\lambda y.y)(\lambda x.(xy)))$
- ▶ $(x(\lambda x.(\lambda x.x)))$
- ▶ $(\lambda x.(yz))$

Associatividade e precedência

Para reduzir a quantidade de parênteses, modificamos a gramática:

$$V \rightarrow u | v | x | y | z | w | \dots$$

$$T \rightarrow V$$

$$T \rightarrow TT$$

$$T \rightarrow \lambda V.T$$

$$T \rightarrow (T)$$

que assim se torna ambígua. Por exemplo, considere o termo $\lambda x.xx$. Este termo representa $((\lambda x.x)x)$ ou $(\lambda x.(xx))$? A última regra permite o uso opcional de parênteses.

Associatividade e precedência

Para reduzir a quantidade de parênteses e evitar ambigüidades, são usadas as seguintes convenções:

- ▶ Aplicações tem prioridade sobre abstrações;
- ▶ Aplicações são associativas à esquerda;
- ▶ Abstrações são associativas à direita.

Por exemplo:

- ▶ $\lambda x.PQ$ denota $(\lambda x.(PQ))$ — e não $((\lambda x.P)Q)$;
- ▶ $MNPQ$ denota $((((MN)P)Q))$ — e não $(M(N(PQ)))$;
- ▶ $\lambda xyz.M$ denota $(\lambda x.(\lambda y.(\lambda z.M)))$

O símbolo \equiv é usado para denotar a equivalência sintática de λ -termos.

Exemplos

- ▶ $xyz(yx) \equiv (((xy)z)(yx))$
- ▶ $\lambda x.(uxy) \equiv (\lambda x.((ux)y))$
- ▶ $\lambda u.u(\lambda x.y) \equiv (\lambda u.(u(\lambda x.y)))$
- ▶ $(\lambda u.vuu)zy \equiv (((\lambda u.((vu)u))z)y)$
- ▶ $ux(yz)(\lambda v.vy) \equiv (((ux)(yz))(\lambda v.(vy)))$
- ▶ $(\lambda xyz.xz(yz))uvw \equiv (((((\lambda x.(\lambda y.(\lambda z.((xz)(yz))))))u)v)w)$

Comprimento

O “comprimento” de um λ -termo M — $lgh(M)$ — é o número total de ocorrências de átomos em M .

- ▶ Para todo átomo a , $lgh(a) = 1$;
- ▶ $lgh(MN) = lgh(M) + lgh(N)$;
- ▶ $lgh(\lambda x.M) = 1 + lgh(M)$

Se $M \equiv x(\lambda y.yux)$ então $lgh(M) = 5$.

Ocorrência

Sejam P e Q dois λ -termos. A relação P “ocorre” em Q (ou ainda, P está contido em Q , Q contém P ou P é subtermo de Q) é definida de forma indutiva:

- ▶ P ocorre em P ;
- ▶ Se P ocorre em M ou em N , então P ocorre em (MN) ;
- ▶ Se P ocorre em M ou $P \equiv x$ então P ocorre em $(\lambda x.M)$.

No termo $((xy)(\lambda x.(xy)))$ existem duas ocorrências de (xy) e três de x .

Exemplos

- ▶ As ocorrências de xy em $\lambda xy.xy$ são $\lambda xy.xy \equiv (\lambda x.(\lambda y.(\underbrace{xy})))$.
- ▶ As ocorrências de uv em $x(uv)(\lambda u.v(uv))uv$ são $((((x(\underbrace{uv})))(\lambda u.(v(\underbrace{uv}))))u)v$.
- ▶ O termo $\lambda u.u$ não ocorre em $\lambda u.uv$ pois $\lambda u.uv \equiv (\lambda u.(uv))$.

Escopo

Para uma particular ocorrência de $\lambda x.M$ em P , a ocorrência de M é chamada de “escopo” da ocorrência de λx à esquerda.

Exemplo: seja

$$P \equiv (\lambda y.yx(\lambda x.y(\lambda y.z)x))vw$$

- ▶ O escopo do λy mais à esquerda é $yx(\lambda x.y(\lambda y.z)x)$;
- ▶ O escopo do λx é $y(\lambda y.z)x$;
- ▶ O escopo do λy mais à direita é z .

Variáveis livres e ligadas

A ocorrência de uma variável x em um termo P é dita:

- ▶ “Ligada” se ela está no escopo de um λx em P ;
- ▶ “Ligada e ligadora” se e somente se ela é o x em λx ;
- ▶ “Livre” caso contrário.

Variáveis livres e ligadas

- ▶ Se x tem pelo menos uma ocorrência ligadora em P , x é chamada de “variável ligada” de P ;
- ▶ Se x tem pelo menos uma ocorrência livre em P , x é chamada “variável livre” de P ;
- ▶ O conjunto de todas as variáveis livres de P é denotado $FV(P)$;
- ▶ Um termo que não contém variáveis livres é chamado “fechado”.

Variáveis livres e ligadas

Para determinar $FV(P)$:

- ▶ $FV(\sigma) = \{\sigma\}$ se σ é variável;
- ▶ $FV(\sigma) = \emptyset$ se σ é constante;
- ▶ $FV((MN)) = FV(M) \cup FV(N)$;
- ▶ $FV((\lambda x.M)) = FV(M) - \{x\}$.

Exemplo

Considere o termo $xv(\lambda yz.yv))w) \equiv$

$$(((xv)(\lambda y.(\lambda z.(yv))))w)$$

- ▶ O x mais à esquerda é livre;
- ▶ O v mais à esquerda é livre;
- ▶ O y mais à esquerda é ligado e ligador;
- ▶ O único z é ligado e ligador;
- ▶ O y mais à direita é ligado mas não é ligador;
- ▶ O v mais à direita é livre;
- ▶ O único w é livre.

Exemplo

Considere o termo $P \equiv$

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

- ▶ Todos os quatro y são ligados;
- ▶ Os y mais à esquerda e mais à direita são ligadores;
- ▶ O x mais à esquerda é livre;
- ▶ O x central é ligado e ligador;
- ▶ O x mais à direita é ligado mas não ligador;
- ▶ z, v e w são livres.
- ▶ Logo, $FV(P) = \{x, z, v, w\}$; x , nesse caso, é uma variável ligada e também livre de P .

Definição

Para todo M, N, x , $[N/x]M$ é definido como o resultado da substituição de toda ocorrência livre de x em M por N , juntamente com a mudança de variáveis ligadas caso isso seja necessário para evitar colisões.

- $[N/x]x \equiv N$;
- $[N/x]a \equiv a$, para todo átomo $a \neq x$;
- $[N/x](PQ) \equiv ([N/x]P[N/x]Q)$;
- $[N/x](\lambda x.P) \equiv \lambda x.P$;
- $[N/x](\lambda y.P) \equiv \lambda y.P$, se $x \notin FV(P)$;
- $[N/x](\lambda y.P) \equiv \lambda y.[N/x]P$, se $x \in FV(P)$ e $y \notin FV(N)$;
- $[N/x](\lambda y.P) \equiv \lambda z.[N/x][z/y]P$, se $x \in FV(P)$ e $y \in FV(N)$.

Nos casos (e)-(g), $y \neq x$; no caso (g), z é a primeira variável $\notin FV(NP)$.

Substituição de variável ligada

Considere (i) $\lambda y.x$ e (ii) $\lambda w.x$. Trata-se da mesma função (função constante que retorna x), porém com diferentes parâmetros.

- i. Suponha $[w/x](\lambda y.x)$. Então, $[w/x](\lambda y.x) \equiv \lambda y.w$, pela aplicação da regra (f), pois $x \in FV(x)$ e $y \notin FV(w)$;
- ii. Suponha $[w/x](\lambda w.x)$. Se a substituição fosse feita também pela regra (f), então $[w/x](\lambda w.x) \equiv \lambda w.w$. Mas $\lambda w.w$ é a função identidade, e não a função constante. Para evitar esse problema, a aplicação da regra (g) produz $[w/x](\lambda w.x) \equiv \lambda z.[w/x][z/w]x \equiv \lambda z.[w/x]x \equiv \lambda z.w$, e nesse caso obtemos a mesma função constante. Observe que, nesse caso, $x \in FV(x)$ e $w \in FV(w)$.

Exercícios

Avaliar as seguintes substituições conforme as regras anteriormente apresentadas:

- ▶ $[(uv)/x](\lambda x.zy)$
- ▶ $[(\lambda y.xy)/x](\lambda y.x(\lambda x.x))$
- ▶ $[(uv)/x](\lambda y.x(\lambda w.vwx))$
- ▶ $[(\lambda y.vy)/x](y(\lambda v.xv))$

Soluções dos exercícios

$[(uv)/x](\lambda x.zy)$

- ▶ Aplicação da regra (d): $\lambda x.zy$

Soluções dos exercícios

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

- ▶ Reescrita com todos os parênteses: $[(\lambda y. (xy))/x](\lambda y. (x(\lambda x. x)))$
- ▶ Aplicação da regra (f), pois $x \in FV(x(\lambda x. x))$ e $y \notin FV(\lambda y. (xy))$:
 $\lambda y. ([\lambda y. (xy)/x](x(\lambda x. x)))$
- ▶ Aplicação da regra (c): $\lambda y. ([\lambda y. (xy)/x]x)([\lambda y. (xy)/x](\lambda x. x))$
- ▶ Aplicação da regra (a): $\lambda y. (\lambda y. (xy))([\lambda y. (xy)/x](\lambda x. x))$
- ▶ Aplicação da regra (e), pois $x \notin FV(x)$: $\lambda y. ((\lambda y. (xy))(\lambda x. x))$
- ▶ Remoção dos parênteses desnecessários: $\lambda y. (\lambda y. xy)(\lambda x. x)$

Soluções dos exercícios

$$[uv/x](\lambda y.x(\lambda w.vwx))$$

- ▶ Reescrita com todos os parênteses: $[uv/x](\lambda y.(x(\lambda w.((vw)x))))$
- ▶ Aplicação da regra (f), pois $x \in FV(x(\lambda w.((vw)x)))$ e $y \notin FV(uv)$:
 $\lambda y.([uv/x](x(\lambda w.((vw)x))))$
- ▶ Aplicação da regra (c): $\lambda y.([uv/x]x)([uv/x](\lambda w.((vw)x)))$
- ▶ Aplicação da regra (a): $\lambda y.(uv[uv/x](\lambda w.((vw)x)))$
- ▶ Aplicação da regra (f), pois $x \in FV(\lambda w.((vw)x))$ e $w \notin FV(uv)$:
 $\lambda y.(uv(\lambda w.([uv/x]((vw)x))))$
- ▶ Aplicação da regra (c): $\lambda y.(uv(\lambda w.([uv/x](vw)[uv/x]x)))$
- ▶ Aplicação da regra (b): $\lambda y.(uv(\lambda w.(vw[uv/x]x)))$
- ▶ Aplicação da regra (a): $\lambda y.(uv(\lambda w.(vw)(uv)))$
- ▶ Remoção dos parênteses desnecessários: $\lambda y.uv(\lambda w.vw(uv))$

Soluções dos exercícios

$$[\lambda y.vy/x](y(\lambda v.xv))$$

- ▶ Aplicação da regra (c): $([\lambda y.vy/x]y)([\lambda y.vy/x](\lambda v.xv))$
- ▶ Aplicação da regra (b): $y([\lambda y.vy/x](\lambda v.xv))$
- ▶ Aplicação da regra (g), pois $x \in FV(xv)$ e $v \in FV(\lambda y.vy)$:
 $y(\lambda z.[\lambda y.vy/x][z/v](xv))$
- ▶ Aplicação da regra (c): $y(\lambda z.[\lambda y.vy/x](((z/v)x)((z/v)v)))$
- ▶ Aplicação da regra (b): $y(\lambda z.[\lambda y.vy/x](x((z/v)v)))$
- ▶ Aplicação da regra (a): $y(\lambda z.[\lambda y.vy/x](xz))$
- ▶ Aplicação da regra (c): $y(\lambda z.(((\lambda y.vy/x)x)([\lambda y.vy/x]z)))$
- ▶ Aplicação da regra (a): $y(\lambda z.((\lambda y.vy)([\lambda y.vy/x]z)))$
- ▶ Aplicação da regra (b): $y(\lambda z.((\lambda y.vy)z))$
- ▶ Remoção dos parênteses desnecessários: $y(\lambda z.(\lambda y.vy)z)$

Conversão- α

Seja P um termo que contém uma ocorrência de $\lambda x.M$ e suponha que $y \notin FV(M)$. A substituição de:

$$\lambda x.M \text{ por } \lambda y.[y/x]M$$

é chamada *troca de variável livre* ou ainda *conversão- α* em P . Se P pode ser transformado em Q por meio de uma série finita de conversões- α , diz-se que P e Q são *congruentes* ou então que P é *α -conversível* para Q , denotado:

$$P \equiv_{\alpha} Q.$$

Exemplo de conversão- α

$$\begin{aligned}\lambda xy.x(xy) &\equiv \lambda x.(\lambda y.x(xy)) \\ &\equiv_{\alpha} \lambda x.(\lambda v.x(xv)) \\ &\equiv_{\alpha} \lambda u.(\lambda v.u(uv)) \\ &\equiv \lambda uv.u(uv)\end{aligned}$$

Propriedades da conversão- α

Para todos P, Q e R :

- ▶ (*reflexividade*) $P \equiv_{\alpha} P$;
- ▶ (*transitividade*) $P \equiv_{\alpha} Q, Q \equiv_{\alpha} R \Rightarrow P \equiv_{\alpha} R$;
- ▶ (*simetria*) $P \equiv_{\alpha} Q \Rightarrow Q \equiv_{\alpha} P$.

Definição

Um termo da forma:

$$(\lambda x.M)N$$

é chamado β -redex, e o termo correspondente:

$$[N/x]M$$

é chamado o seu *contractum*. Se um termo P contém uma ocorrência de $(\lambda x.M)N$ e a mesma é substituída por $[N/x]M$, gerando P' , diz-se que que ocorrência redex em P foi *contraída* e que P β -contraí para P' , denotado:

$$P \triangleright_{1\beta} P'.$$

Definição

Se um termo P pode ser convertido em um termo Q através de um número finito de reduções- β e conversões- α , diz-se que P β -reduz para Q , denotado:

$$P \triangleright_{\beta} Q.$$

Exemplos

$$\begin{aligned}(\lambda x.x(xy))N &\triangleright_{1\beta} N(Ny) \\ (\lambda x.y)N &\triangleright_{1\beta} y\end{aligned}$$

Exemplos

$$\begin{aligned}(\lambda x.(\lambda y.yx)z)v &\triangleright_{1\beta} [v/x]((\lambda y.yx)z) \equiv (\lambda y.yv)z \\ &\triangleright_{1\beta} [z/y](yv) \equiv zv\end{aligned}$$

Exemplos

$$\begin{aligned}
 (\lambda x.xx)(\lambda x.xx) &\triangleright_{1\beta} [(\lambda x.xx)/x](xx) \equiv (\lambda x.xx)(\lambda x.xx) \\
 &\textit{etc.}
 \end{aligned}$$

Exemplos

$$\begin{aligned}
 (\lambda x. xxy)(\lambda x. xxy) &\triangleright_{1\beta} (\lambda x. xxy)(\lambda x. xxy)y \\
 &\triangleright_{1\beta} (\lambda x. xxy)(\lambda x. xxy)yy \\
 &\triangleright_{1\beta} (\lambda x. xxy)(\lambda x. xxy)yyy \\
 &\triangleright_{1\beta} (\lambda x. xxy)(\lambda x. xxy)yyyy \\
 &\textit{etc.}
 \end{aligned}$$

Forma normal

- ▶ Um termo Q que não possui nenhuma redução- β é chamado de *forma normal- β* ;
- ▶ Se um termo P reduz- β para um termo Q na forma normal- β , então diz-se que Q é uma *forma normal- β* de P .

Exemplos

- ▶ $(\lambda x.(\lambda y.yx)z)v$ tem como forma normal- β zv ;
- ▶ $L \equiv (\lambda x.xxy)(\lambda x.xxy) \triangleright_{1\beta} Ly \triangleright_{1\beta} Lyy \triangleright_{1\beta} \dots$ não tem forma normal- β pois trata-se de uma seqüência infinita e não existe outra forma de reduzir- β a expressão;
- ▶ $P \equiv (\lambda u.v)L$ possui as seguintes reduções:
 - ▶ $P \equiv (\lambda u.v)L \triangleright_{1\beta} [L/u]v \equiv v$;
 - ▶ $P \triangleright_{1\beta} (\lambda u.v)(Ly) \triangleright_{1\beta} (\lambda u.v)(Lyy) \triangleright_{1\beta} \dots$

Portanto, P tem forma normal- β e também uma seqüência infinita de reduções.

- ▶ $(\lambda x.xx)(\lambda x.xx)$, também conhecido como Ω , não possui forma normal- β , pois ele reduz sempre para si mesmo e não há outra redução possível.

Interpretação

Expressão lambda:

- ▶ Representa um programa, um algoritmo, um procedimento para produzir um resultado;

Redução- β :

- ▶ Representa uma computação, a passagem de um estado de um programa para o estado seguinte, dentro do processo de geração de um resultado.

Forma normal:

- ▶ Representa um resultado de uma computação, um valor que não é passível de novas simplificações ou elaborações.

Exercícios

Reduzir os seguintes termos para formas normais- β :

- ▶ $(\lambda x.xy)(\lambda u.vuu)$
- ▶ $(\lambda xy.yx)uv$
- ▶ $(\lambda x.x(x(yz))x)(\lambda u.uv)$
- ▶ $(\lambda x.xxy)(\lambda y.yz)$
- ▶ $(\lambda xy.xyy)(\lambda u.uyx)$
- ▶ $(\lambda xyz.xz(yz))((\lambda xy.yx)u)((\lambda xy.yx)v)w$

Soluções dos exercícios

$$\underbrace{(\lambda x.xy)(\lambda u.vuu)} \triangleright_{1\beta} [\lambda u.vuu/x](xy) \equiv \underbrace{(\lambda u.vuu)y}$$

$$\triangleright_{1\beta} [y/u](vuu) \equiv vyy$$

Soluções dos exercícios

$$(\lambda xy.yx)uv \equiv ((\lambda x.(\lambda y.yx))u)v$$

$$\underbrace{((\lambda x.(\lambda y.yx))u)}v \triangleright_{1\beta} ([u/x](\lambda y.yx))v \equiv \underbrace{(\lambda y.yu)}v$$

$$\triangleright_{1\beta} [v/y](yu) \equiv vu$$

Soluções dos exercícios

$$\begin{aligned}
 \underbrace{(\lambda x. x(x(yz)))x}_{\text{}} (\lambda u. uv) &\triangleright_{1\beta} [(\lambda u. uv)/x](x(x(yz)))x \\
 &\equiv (\lambda u. uv)(\underbrace{(\lambda u. uv)(yz)}) (\lambda u. uv) \\
 &\triangleright_{1\beta} (\lambda u. uv)([(yz)/u](uv)) (\lambda u. uv) \\
 &\equiv \underbrace{(\lambda u. uv)((yz)v)}_{\text{}} (\lambda u. uv) \\
 &\triangleright_{1\beta} [(((yz)v)/u](uv)) (\lambda u. uv) \\
 &\equiv ((yz)v)v (\lambda u. uv) \\
 &\equiv yzvv (\lambda u. uv)
 \end{aligned}$$

Soluções dos exercícios

$$\begin{aligned}
 \underbrace{(\lambda x. xxy)(\lambda y. yz)} &\triangleright_{1\beta} [(\lambda y. yz)/x](xxy) \equiv \underbrace{(\lambda y. yz)(\lambda y. yz)} y \\
 &\triangleright_{1\beta} [(\lambda y. yz)/y](yz)y \equiv \underbrace{(\lambda y. yz)} z y \\
 &\triangleright_{1\beta} ([z/y](yz))y \equiv zzy
 \end{aligned}$$

Soluções dos exercícios

$$(\lambda xy.xyy)(\lambda u.uyx) \equiv (\lambda x.(\lambda y.xyy))(\lambda u.uyx)$$

$$\begin{aligned} \underbrace{(\lambda x.(\lambda y.xyy))(\lambda u.uyx)} &\triangleright_{1\beta} [(\lambda u.uyx)/x](\lambda y.xyy) \\ &\equiv [(\lambda u.uyx)/x](\lambda z.xzz) \\ &\equiv \lambda z. \underbrace{(\lambda u.uyx)z} z \\ &\triangleright_{1\beta} \lambda z.([z/u](uyx))z \\ &\equiv \lambda z.zyxz \end{aligned}$$

ou ainda:

$$\begin{aligned} \underbrace{(\lambda x.(\lambda y.xyy))(\lambda u.uyx)} &\equiv_{\alpha} (\lambda v.(\lambda w.vww))(\lambda u.uyx) \\ &\triangleright_{\beta} \lambda w.wyxw \end{aligned}$$

Soluções dos exercícios

No caso:

$$(\lambda xyz.xz(yz))((\lambda xy.yx)u)((\lambda xy.yx)v)w$$

observar que:

- ▶ $(\lambda xy.yx)u \equiv (\lambda x.(\lambda y.yx))u \triangleright_{1\beta} [u/x](\lambda y.yx) \equiv \lambda y.yu$
- ▶ $(\lambda xy.yx)v \triangleright_{1\beta} \lambda y.yv$

Soluções dos exercícios

$$\begin{aligned}
& (\lambda x y z. x z (y z)) \underbrace{((\lambda x y. y x) u)} \underbrace{((\lambda x y. y x) v)} w \triangleright_{\beta} \\
& \quad (\lambda x y z. x z (y z)) (\lambda y. y u) (\lambda y. y v) w \equiv \\
& \quad \underbrace{(\lambda x. \lambda y. \lambda z. x z (y z))} (\lambda y. y u) (\lambda y. y v) w \triangleright_{1\beta} \\
& \quad [(\lambda y. y u) / x] (\lambda y. \lambda z. x z (y z)) (\lambda y. y v) w \equiv \\
& \quad \underbrace{(\lambda y. \lambda z. (\lambda y. y u) z (y z))} (\lambda y. y v) w \triangleright_{1\beta} \\
& \quad [(\lambda y. y v) / y] (\lambda z. (\lambda y. y u) z (y z)) w \equiv \\
& \quad \underbrace{(\lambda z. (\lambda y. y u) z ((\lambda y. y v) z))} w \triangleright_{1\beta}
\end{aligned}$$

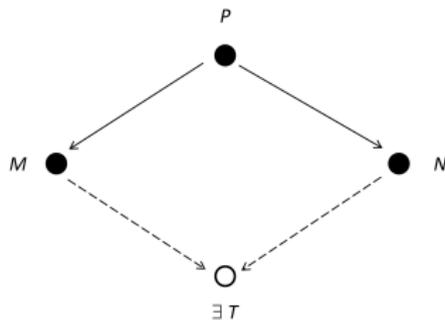
Soluções dos exercícios

$$\begin{aligned}
[w/z]((\lambda y.yu)z((\lambda y.yv)z)) &\equiv \\
\underbrace{(\lambda y.yu)w}((\lambda y.yv)w) &\triangleright_{1\beta} \\
[w/y](yu)((\lambda y.yv)w) &\equiv \\
wu(\underbrace{(\lambda y.yv)w}) &\triangleright_{1\beta} \\
wu([w/y](yv)) &\equiv \\
wu(wv) &
\end{aligned}$$

Teorema de Church-Rosser para \triangleright_β

Se $P \triangleright_\beta M$ e $P \triangleright_\beta N$, então existe um termo T tal que:

$$M \triangleright_\beta T \quad \text{e} \quad N \triangleright_\beta T.$$



Teorema de Church-Rosser para \triangleright_{β}

A redução- β é *confluente*.

Conseqüências:

- ▶ Uma computação no Cálculo Lambda não pode produzir dois ou mais resultados diferentes;
- ▶ Duas ou mais reduções de um mesmo termo produzem a mesma forma normal (o resultado da computação independe do caminho escolhido).
- ▶ Exemplo:

$P \equiv (\lambda u.v)L$ reduz para $M \equiv v$ e também para $N \equiv (\lambda u.v)(Ly)$.

No entanto, tanto M quanto N reduzem para $T \equiv v$.

Teorema de Church-Rosser para \triangleright_β

Corolário: Se P tem uma forma normal- β , ela é única módulo \equiv_α , ou seja, se P possui formas normais- β M e N , então $M \equiv_\alpha N$.

Prova: Suponha que $P \triangleright_\beta M$ e $P \triangleright_\beta N$, e que ambos M, N reduzem para T . Como M e N não possuem redexes e, pelo Teorema de Church-Rosser, deve existir um termo T tal que $M \triangleright_\beta T$ e $N \triangleright_\beta T$, segue que $M \equiv_\alpha T$ e $N \equiv_\alpha T$. Como a conversão- α é transitiva, segue que $M \equiv_\alpha N$.

Exercício 1

Inserir os parênteses e λ s faltantes nos seguintes termos- λ abreviados:

- 1 $xx(xxx)x$;
- 2 $vw(\lambda xy.vx)$;
- 3 $(\lambda xy.x)uv$;
- 4 $w(\lambda xyz.xz(yz))uv$.

Solução do exercício 1

- 1 $xx(xxx)x \equiv (((xx)((xx)x))x);$
- 2 $vw(\lambda xy.vx) \equiv ((vw)(\lambda x.(\lambda y.(vx))));$
- 3 $(\lambda xy.x)uv \equiv (((\lambda x.(\lambda y.x))u)v);$
- 4 $w(\lambda xyz.xz(yz))uv \equiv (((w(\lambda x.(\lambda y.(\lambda z.((xz)(yz))))))u)v).$

Exercício 2

Marcar todas as ocorrências de xy nos seguintes termos:

- 1 $(\lambda xy.xy)xy$;
- 2 $(\lambda xy.xy)(xy)$;
- 3 $\lambda xy.xy(xy)$.

Solução do exercício 2

- 1 $(\lambda xy.xy)xy \equiv (((\lambda x.(\lambda y.(\underline{xy})))x)y);$
- 2 $(\lambda xy.xy)(xy) \equiv ((\lambda x.(\lambda y.(\underline{xy})))\underline{xy});$
- 3 $\lambda xy.xy(xy) \equiv (\lambda x.(\lambda y.((\underline{xy})(\underline{xy}))))).$

Exercício 3

Indicar os termos dos exercícios 1 e 2 que contêm algum dos seguintes subtermos:

- 1 $\lambda y.xy;$
- 2 $y(xy);$
- 3 $\lambda xy.x.$

Solução do exercício 3

- 1 $\lambda y.xy$:
 - (2.1): $(((\lambda x.((\lambda y.(xy))))x)y)$;
 - (2.2): $((\lambda x.((\underline{\lambda y.(xy)})))(xy))$.
- 2 $y(xy)$:
Nenhuma ocorrência.
- 3 $\lambda xy.x \equiv \lambda x.(\lambda y.x)$:
(1.3): $((\underline{(\lambda x.(\lambda y.x))})u)v$.

Exercício 4

Avaliar as seguintes substituições:

1 $[vw/x](x(\lambda y.yx));$

2 $[vw/x](x(\lambda x.yx));$

3 $[ux/x](x(\lambda y.yx));$

4 $[uy/x](x(\lambda y.yx)).$

Solução do exercício 4

- 1 $[vw/x](x(\lambda y.yx)) \equiv ([vw/x]x[vw/x](\lambda y.yx)) \equiv$
 $((vw)[vw/x](\lambda y.yx)) \equiv ((vw)(\lambda y.[vw/x](yx))) \equiv$
 $((vw)(\lambda y.([vw/x]y[vw/x]x))) \equiv ((vw)(\lambda y.(y(vw))));$
- 2 $[vw/x](x(\lambda x.yx)) \equiv ([vw/x]x[vw/x](\lambda x.yx)) \equiv$
 $(vw[vw/x](\lambda x.yx)) \equiv (vw(\lambda x.yx));$
- 3 $[ux/x](x(\lambda y.yx)) \equiv ([ux/x]x[ux/x](\lambda y.yx)) \equiv$
 $((ux)[ux/x](\lambda y.yx)) \equiv ((ux)(\lambda y.[ux/x](yx))) \equiv$
 $((ux)(\lambda y.([ux/x]y[ux/x]x))) \equiv ((ux)(\lambda y.(y[ux/x]x))) \equiv$
 $((ux)(\lambda y.(y(ux))));$
- 4 $[uy/x](x(\lambda y.yx)) \equiv ([uy/x]x[uy/x](\lambda y.yx)) \equiv$
 $((uy)[uy/x](\lambda y.yx)) \equiv ((uy)(\lambda z.[uy/x][z/y](yx))) \equiv$
 $((uy)(\lambda z.[uy/x]([z/y]y[z/y]x))) \equiv ((uy)(\lambda z.[uy/x](z[z/y]x))) \equiv$
 $((uy)(\lambda z.[uy/x](zx))) \equiv ((uy)(\lambda z.([uy/x]z[uy/x]x))) \equiv$
 $((uy)(\lambda z.(z[uy/x]x))) \equiv ((uy)(\lambda z.(z(uy)))).$

Exercício 5

Reduzir os seguintes termos para formas normais- β :

- 1 $(\lambda xy.xyy)uv$;
- 2 $(\lambda xy.yx)(uv)zw$;
- 3 $(\lambda xy.x)(\lambda u.u)$;
- 4 $(\lambda xyz.xz(yz))(\lambda uv.u)$.

Solução do exercício 5

$$\begin{aligned}
 1. \quad & (\lambda xy. xyy)uv \equiv \\
 & (((\lambda x. (\lambda y. ((xy)y)))\underline{u})v) \triangleright_{1\beta} \\
 & (([u/x](\lambda y. ((xy)y)))v) \equiv \\
 & (((\lambda y. ((uy)y))\underline{v}) \triangleright_{1\beta} \\
 & ([v/y]((uy)y)) \equiv \\
 & ((uv)v)
 \end{aligned}$$

Solução do exercício 5

$$\begin{aligned}
 2. \quad & (\lambda xy.yx)(uv)zw \equiv \\
 & (((\lambda x.(\lambda y.(yx)))(\underline{uv}))z)w \triangleright_{1\beta} \\
 & ((([uv/x](\lambda y.(yx)))z)w) \equiv \\
 & (((\lambda y.[uv/x](yx)))z)w \equiv \\
 & (((\lambda y.([uv/x]y[uv/x]x)))z)w) \equiv \\
 & (((\lambda y.(y[uv/x]x)))z)w \equiv \\
 & (((\lambda y.(y(uv))))z)w \triangleright_{1\beta} \\
 & (([z/y](y(uv)))w) \equiv \\
 & ((([z/y]y[z/y](uv)))w) \equiv \\
 & (((z[z/y](uv)))w) \equiv \\
 & (((z([z/y]u[z/y]v)))w) \equiv \\
 & (((z(u[z/y]v)))w) \equiv \\
 & (((z(uv)))w)
 \end{aligned}$$

Solução do exercício 5

$$\begin{aligned}
 3. \quad & (\lambda x y . x)(\lambda u . u) \equiv \\
 & (\lambda x . (\lambda y . x))(\lambda u . u) \triangleright_{1\beta} \\
 & [((\lambda u . u))/x](\lambda y . x) \equiv \\
 & (\lambda y . [((\lambda u . u))/x]x) \equiv \\
 & (\lambda y . (\lambda u . u))
 \end{aligned}$$

Solução do exercício 5

$$\begin{aligned}
 4. \quad & (\lambda x y z. x z (y z)) (\lambda u v. u) \equiv \\
 & (\lambda x. (\lambda y. (\lambda z. ((x z) (y z)))) (\lambda u. (\lambda v. u))) \triangleright_{1\beta} \\
 & [((\lambda u. (\lambda v. u)) / x) (\lambda y. (\lambda z. ((x z) (y z))))] \equiv \\
 & \lambda y. [(\lambda u. (\lambda v. u)) / x] (\lambda z. ((x z) (y z))) \equiv \\
 & \lambda y. (\lambda z. [(\lambda u. (\lambda v. u)) / x] ((x z) (y z))) \equiv \\
 & \lambda y. (\lambda z. ([(\lambda u. (\lambda v. u)) / x] (x z) [(\lambda u. (\lambda v. u)) / x] (y z))) \equiv \\
 & \lambda y. (\lambda z. (((\lambda u. (\lambda v. u)) z) [(\lambda u. (\lambda v. u)) / x] (y z))) \equiv \\
 & \lambda y. (\lambda z. (((\lambda u. (\lambda v. u)) \underline{z}) (y z))) \triangleright_{1\beta} \\
 & \lambda y. (\lambda z. ([z / u] (\lambda v. u)) (y z)) \equiv \\
 & \lambda y. (\lambda z. (((\lambda v. [z / u] u)) (y z))) \equiv \\
 & \lambda y. (\lambda z. (((\lambda v. z)) (\underline{y z}))) \triangleright_{1\beta} \\
 & \lambda y. (\lambda z. ([y z / v] z)) \equiv \\
 & \lambda y. (\lambda z. (z))
 \end{aligned}$$

Sistemas puro e aplicado

- ▶ No Cálculo Lambda “*puro*” não existe representação para números inteiros, operações aritméticas, valores ou operadores lógicos, entre outros tipos de dados e operações usualmente encontradas em linguagens de programação de alto-nível;
- ▶ No Cálculo Lambda “*aplicado*” admite-se o uso explícito dos mesmos:

$$\lambda x.x + 1$$

$$(\lambda x.x + 1)(3) \triangleright_{1\beta} [3/x](x + 1) \equiv 3 + 1 \equiv 4$$

- ▶ É possível, no entanto, representar tipos de dados e operadores quaisquer usando o sistema puro, como demonstram os casos apresentados a seguir.

Numerais de Church

Definição

Para todo $n \in \mathbb{N}$, o “Numeral de Church” de n , denotado \bar{n} , é um termo- λ que representa n :

$$\bar{n} := \lambda xy. x^n y$$

onde:

$$x^n y$$

é definido como:

$$\begin{cases} x^n y \equiv \underbrace{x(x(\dots(x y)\dots))}_{n \text{ vezes}} \text{ se } n \geq 1 \\ x^0 y \equiv y \end{cases}$$

Numerais de Church

Exemplos

$$\bar{0} := \lambda xy.y$$

$$\bar{1} := \lambda xy.xy$$

$$\bar{2} := \lambda xy.x(xy)$$

$$\bar{3} := \lambda xy.x(x(xy))$$

$$\bar{4} := \lambda xy.x(x(x(xy)))$$

...

Numerais de Church

Propriedade

Os Numerais de Church tem a propriedade de que, para quaisquer termos F e X ,

$$\bar{n}FX \triangleright_{\beta} F^n X.$$

Em outras palavras, o numeral de Church inserido na frente de uma aplicação de uma função ao seu argumento representa a aplicação repetida dessa função o mesmo número de vezes.

Numerais de Church

Propriedade

Exemplo:

$$\begin{aligned}
 \bar{2}FX &\equiv (\lambda xy.x(xy))FX \\
 &\equiv ((\lambda xy.x(xy))F)X \\
 \triangleright_{1\beta} & [F/x](\lambda y.x(xy))X \\
 &\equiv (\lambda y.F(Fy))X \\
 \triangleright_{1\beta} & [X/y]F(Fy) \\
 &\equiv F(FX) \\
 &\equiv F^2X
 \end{aligned}$$

Numerais de Church

Sucessor

O sucessor de um Numeral de Church pode ser obtido pela aplicação da expressão:

$$\overline{succ} := \lambda uxy.x(uxy)$$

ao respectivo numeral. É fácil provar que:

$$\overline{succ} \bar{n} \triangleright_{\beta} \overline{n+1}.$$

De fato, basta observar que:

$$(\lambda uxy.x(uxy))\bar{n} \triangleright_{\beta} \lambda xy.x(\bar{n}xy) \equiv \lambda x.\lambda y.xx^n y \equiv \lambda x.\lambda y.x^{n+1}y \equiv \overline{n+1}.$$

Numerais de Church

Sucessor

Exemplo:

$$\begin{aligned}
 \overline{succ} \bar{0} &\equiv (\lambda uxy.x(uxy))(\lambda xy.y) \\
 \triangleright_{1\beta} &[(\lambda xy.y)/u](\lambda xy.x(uxy)) \\
 &\equiv (\lambda xy.x((\lambda xy.y)xy)) \\
 \triangleright_{\beta} &(\lambda xy.xy) \\
 &\equiv \bar{1}
 \end{aligned}$$

Numerais de Church

Adição

A adição de dois Numerais de Church pode ser obtida pela aplicação da expressão:

$$\overline{add} := \lambda uvxy.ux(vxy)$$

aos respectivos operandos. Nesse caso, temos que:

$$\overline{add} \overline{m} \overline{n} \triangleright_{\beta} \overline{m+n}.$$

De fato, basta observar que:

$$\begin{aligned} (\lambda uvxy.ux(vxy))\overline{m} \overline{n} \triangleright_{\beta} \lambda xy.\overline{m}x(\overline{n}xy) &\equiv \lambda xy.\overline{m}x(x^n y) \\ &\equiv \lambda xy.x^m(x^n y) \equiv \lambda xy.x^{m+n}y \equiv \overline{m+n} \end{aligned}$$

Numerais de Church

Adição

Exemplo:

$$\begin{aligned}
 \overline{add} \ \bar{1} \ \bar{2} &\equiv (\lambda uvxy.ux(vxy))(\lambda xy.xy)(\lambda xy.x(xy)) \\
 \triangleright_{1\beta} & [((\lambda xy.xy)/u)(\lambda vxy.ux(vxy))](\lambda xy.x(xy)) \\
 &\equiv (\lambda vxy.(\lambda xy.xy)x(vxy))(\lambda xy.x(xy)) \\
 \triangleright_{\beta} & (\lambda vxy.x(vxy))(\lambda xy.x(xy)) \\
 \triangleright_{1\beta} & [(\lambda xy.x(xy))/v](\lambda xy.x(vxy)) \\
 &\equiv (\lambda xy.x((\lambda xy.x(xy))xy)) \\
 \triangleright_{\beta} & (\lambda xy.x(x(xy))) \\
 &\equiv \bar{3}
 \end{aligned}$$

Numerais de Church

Multiplicação

A multiplicação de dois Numerais de Church pode ser obtida pela aplicação da expressão:

$$\overline{mult} := \lambda uvx.u(vx)$$

aos respectivos operandos. Nesse caso, temos que:

$$\overline{mult} \overline{m} \overline{n} \triangleright_{\beta} \overline{m * n}$$

Numerais de Church

Multiplicação

De fato, temos que:

$$\begin{aligned}
 (\lambda uvx.u(vx)) \bar{m} \bar{n} &\triangleright_{\beta} \lambda x.\bar{m}(\bar{n}x) \\
 &\equiv \lambda x.\bar{m}((\lambda y.\lambda z.y^n z)x) \\
 &\triangleright_{\beta} \lambda x.\bar{m}(\lambda z.x^n z) \\
 &\equiv \lambda x.(\lambda u.\lambda v.u^m v)(\lambda z.x^n z) \\
 &\triangleright_{\beta} \lambda x.[\lambda z.x^n z/u](\lambda v.u^m v) \\
 &\equiv \lambda x.\lambda v.(\lambda z.x^n z)^m v \\
 &\equiv \lambda x.\lambda v.(\lambda z.x^n z)^{m-1}((\lambda z.x^n z)v)
 \end{aligned}$$

Numerais de Church

Multiplicação

Continuação:

$$\begin{aligned}
 \lambda x. \lambda v. (\lambda z. x^n z)^{m-1} ((\lambda z. x^n z) v) &\triangleright_{\beta} \lambda x. \lambda v. (\lambda z. x^n z)^{m-1} (x^n v) \\
 &\equiv \lambda x. \lambda v. (\lambda z. x^n z)^{m-2} ((\lambda z. x^n z) x^n v) \\
 &\triangleright_{\beta} \lambda x. \lambda v. (\lambda z. x^n z)^{m-2} (x^n (x^n v)) \\
 &\equiv \lambda x. \lambda v. (\lambda z. x^n z)^{m-2} (x^{2*n} v) \\
 &\triangleright_{\beta} \lambda x. \lambda v. x^{m*n} v \\
 &\equiv \overline{m * n}
 \end{aligned}$$

Numerais de Church

Multiplicação

Exemplo:

$$\begin{aligned}
\overline{mult} \bar{2} \bar{2} &\equiv (\lambda uvx.u(vx))\bar{2} \bar{2} \\
\triangleright_{1\beta} &([\bar{2}/u](\lambda vx.u(vx)))\bar{2} \\
&\equiv (\lambda vx.\bar{2}(vx))\bar{2} \\
\triangleright_{1\beta} &[\bar{2}/v](\lambda x.\bar{2}(vx)) \\
&\equiv \lambda x.\bar{2}(\bar{2}x) \\
\triangleright_{1\beta} &\lambda x.\bar{2}(\lambda y.x(xy)) \\
\triangleright_{1\beta} &\lambda x.\lambda y.(\lambda y.x(xy))((\lambda y.x(xy))y) \\
\triangleright_{1\beta} &\lambda x.\lambda y.(\lambda y.x(xy))(x(xy)) \\
\triangleright_{1\beta} &\lambda x.\lambda y.x(x(x(xy))) \\
&\equiv \bar{4}
\end{aligned}$$

Numerais de Church

Exponenciação

A exponenciação de dois Numerais de Church pode ser obtida pela aplicação da expressão:

$$\overline{exp} := \lambda uv.vu$$

aos respectivos operandos. Nesse caso, temos que:

$$\overline{exp} \overline{m} \overline{n} \triangleright_{\beta} \overline{m}^{\overline{n}}.$$

Numerais de Church

Exponenciação

De fato, temos que:

$$\begin{aligned}
 (\lambda uv.vu) \overline{m} \overline{n} &\triangleright_{\beta} \overline{n} \overline{m} \\
 &\equiv (\lambda x.\lambda y.x^n y) \overline{m} \\
 &\triangleright_{\beta} \lambda y.(\overline{m})^n y \\
 &\equiv \lambda y.(\overline{m}(\overline{m}(\dots(\overline{m}(\overline{m}y)))))) \\
 &\triangleright_{\beta} \lambda y.(\overline{m}(\overline{m}(\dots(\overline{m}(\lambda w.y^m w)))))) \\
 &\triangleright_{\beta} \lambda y.(\overline{m}(\overline{m}(\dots(\lambda w.y^{m^2} w)))) \\
 &\triangleright_{\beta} \lambda y.(\lambda w.y^{m^n} w) \\
 &\equiv \overline{m^n}
 \end{aligned}$$

Numerais de Church

Exponenciação

Exemplo:

$$\begin{aligned}
 \overline{exp} \bar{2} \bar{2} &\equiv (\lambda u. \lambda v. vu) \bar{2} \bar{2} \\
 &\triangleright_{\beta} (\lambda v. v \bar{2}) \bar{2} \\
 &\triangleright_{\beta} \bar{2} \bar{2} \\
 &\equiv (\lambda x. \lambda y. x(xy)) \bar{2} \\
 &\triangleright_{\beta} \lambda y. \bar{2}(\bar{2}y) \\
 &\equiv \lambda y. \bar{2}((\lambda x. \lambda z. x(xz))y) \\
 &\triangleright_{\beta} \lambda y. \bar{2}(\lambda z. y(yz)) \\
 &\equiv \lambda y. (\lambda x. \lambda w. x(xw))(\lambda z. y(yz))
 \end{aligned}$$

Numerais de Church

Exponenciação

Exemplo (continuação):

$$\begin{aligned}
 \lambda y.(\lambda x.\lambda w.x(xw))(\lambda z.y(yz)) &\triangleright_{\beta} \lambda y.[\lambda z.y(yz)/x](\lambda w.x(xw)) \\
 &\equiv \lambda y.\lambda w.[\lambda z.y(yz)/x](x(xw)) \\
 &\equiv \lambda y.\lambda w.(\lambda z.y(yz))((\lambda z.y(yz))w) \\
 &\triangleright_{\beta} \lambda y.\lambda w.(\lambda z.y(yz))(y(yw)) \\
 &\triangleright_{\beta} \lambda y.\lambda w.[y(yw)/z](y(yz)) \\
 &\equiv \lambda y.\lambda w.(y(y(y(yw)))) \\
 &\equiv \bar{4}
 \end{aligned}$$

Numerais de Church

Outras operações

- ▶ Predecessor ($n - 1$ se $n > 0$ ou 0 caso contrário):

$$\overline{pred} := \lambda n. \lambda f. \lambda x. n(\lambda g. \lambda h. h(gf))(\lambda u. x)(\lambda u. u)$$

- ▶ Subtração ($m - n$ se $m \geq n$ ou 0 caso contrário):

$$\overline{sub} := \lambda m. \lambda n. (n \overline{pred})m$$

Numerais de Church

Expressões compostas

Através da combinação das expressões lambda anteriores, é possível representar expressões aritméticas mais complexas, como é o caso de:

$$(2^3 + 4) * 5$$

que é denotada:

$$\overline{mult} (\overline{add} (\overline{exp} \overline{2} \overline{3}) \overline{4}) \overline{5} \equiv$$

$$\underbrace{(\lambda uvx.u(vx))}_{\overline{mult}} \underbrace{((\lambda uvxy.u(xvy)))}_{\overline{add}} \underbrace{((\lambda uv.vu) \overline{2} \overline{3})}_{\overline{exp}} \overline{4}) \overline{5} \triangleright_{\beta} \overline{60}.$$

$$\underbrace{\underbrace{\underbrace{\overline{2} \overline{3}}_{2^3} \overline{4}}_{2^3+4}}_{(2^3+4)*5} \overline{5}$$

Numerais de Church

Exercícios

Determinar:

- ▶ $\overline{succ} \bar{1}$;
- ▶ $\overline{add} \bar{2} \bar{3}$;
- ▶ $\overline{mult} \bar{0} \bar{3}$;
- ▶ $\overline{exp} \bar{3} \bar{2}$;
- ▶ $\overline{pred} \bar{2}$;
- ▶ $\overline{sub} \bar{3} \bar{1}$;
- ▶ $\overline{add} (\overline{add} \bar{1} \bar{2})(\overline{mult} \bar{2} \bar{3})$.

Booleanos de Church

Definição

O Cálculo Lambda puro também permite a representação de valores e operações lógicas:

- ▶ $\overline{true} := \lambda x.\lambda y.x$
(projeção do primeiro argumento);
- ▶ $\overline{false} := \lambda x.\lambda y.y$
(projeção do segundo argumento);
Observar que $\overline{false} \equiv \overline{0}$.

Boleanos de Church

AND

$$\overline{and} := \lambda x. \lambda y. xyx$$

É possível provar que:

$$\overline{and} \overline{m} \overline{n} \triangleright_{\beta} \overline{m \text{ and } n}$$

De fato:

$$(\lambda x. \lambda y. xyx) \overline{m} \overline{n} \triangleright_{\beta} \overline{m} \overline{n} \overline{m}$$

- ▶ Se $m = TRUE$, então projeta como resultado o valor de n ;
- ▶ Se $m = FALSE$, projeta como resultado o próprio valor de m .

Booleanos de Church

AND

Exemplos:

$$\begin{aligned}
 (\lambda x. \lambda y. xyx) \overline{true} \overline{true} &\triangleright_{\beta} \overline{true} \underbrace{\overline{true} \overline{true}} \\
 &\triangleright_{\beta} \overline{true}
 \end{aligned}$$

$$\begin{aligned}
 (\lambda x. \lambda y. xyx) \overline{false} \overline{true} &\triangleright_{\beta} \overline{false} \overline{true} \underbrace{\overline{false}} \\
 &\triangleright_{\beta} \overline{false}
 \end{aligned}$$

Booleanos de Church

OR

$$\overline{or} := \lambda x.\lambda y.xxy$$

É possível provar que:

$$\overline{or} \overline{m} \overline{n} \triangleright_{\beta} \overline{\overline{m} or \overline{n}}$$

De fato:

$$(\lambda x.\lambda y.xxy) \overline{m} \overline{n} \triangleright_{\beta} \overline{\overline{m} \overline{m} \overline{n}}$$

- ▶ Se $m = TRUE$, então projeta como resultado o próprio valor de m ;
- ▶ Se $m = FALSE$, projeta como resultado o valor de n .

Booleanos de Church

OR

Exemplos:

$$\begin{aligned}
 (\lambda x. \lambda y. xxy) \overline{true} \overline{false} &\triangleright_{\beta} \overline{true} \underbrace{\overline{true}} \overline{false} \\
 &\triangleright_{\beta} \overline{true}
 \end{aligned}$$

$$\begin{aligned}
 (\lambda x. \lambda y. xxy) \overline{false} \overline{true} &\triangleright_{\beta} \overline{false} \overline{false} \underbrace{\overline{true}} \\
 &\triangleright_{\beta} \overline{true}
 \end{aligned}$$

Booleanos de Church

NOT

$$\overline{not} := \lambda x.\lambda y.\lambda z.xzy$$

É possível provar que:

$$\overline{not} \overline{m} \triangleright_{\beta} \overline{not(m)}$$

De fato:

$$(\lambda x.\lambda y.\lambda z.xzy) \overline{m} \triangleright_{\beta} \lambda y.\lambda z.\overline{m}zy$$

- ▶ Se $m = TRUE$, então $\lambda y.\lambda z.\overline{m}zy \triangleright_{\beta} \lambda y.\lambda z.z \equiv \overline{false}$;
- ▶ Se $m = FALSE$, então $\lambda y.\lambda z.\overline{m}zy \triangleright_{\beta} \lambda y.\lambda z.y \equiv \overline{true}$.

Booleanos de Church

NOT

Exemplos:

$$\begin{aligned}
 (\lambda x. \lambda y. \lambda z. xzy) \overline{true} &\triangleright_{\beta} \lambda y. \lambda z. (\overline{true})zy \\
 &\triangleright_{\beta} \lambda y. \lambda z. z \\
 &\equiv \overline{false}
 \end{aligned}$$

$$\begin{aligned}
 (\lambda x. \lambda y. \lambda z. xzy) \overline{false} &\triangleright_{\beta} \lambda y. \lambda z. (\overline{false})zy \\
 &\triangleright_{\beta} \lambda y. \lambda z. y \\
 &\equiv \overline{true}
 \end{aligned}$$

Booleanos de Church

XOR

$$\overline{xor} := \lambda x.\lambda y.\lambda z.\lambda w.x(ywz)(yzw)$$

É possível provar que:

$$\overline{xor} \overline{m} \overline{n} \triangleright_{\beta} \overline{\overline{m} xor \overline{n}}$$

Boleanos de Church

XOR

De fato:

$$(\lambda x.\lambda y.\lambda z.\lambda w.x(ywz)(yzw)) \overline{m} \overline{n} \triangleright_{\beta} \lambda z.\lambda w.\overline{m}(\overline{n}wz)(\overline{n}zw)$$

- ▶ Se $m = TRUE$, então $\lambda z.\lambda w.\overline{m}(\overline{n}wz)(\overline{n}zw) \triangleright_{\beta} \lambda z.\lambda w.\overline{n}wz$;
 - ▶ Se $n = TRUE$, então $\lambda z.\lambda w.\overline{n}wz \triangleright_{\beta} \lambda z.\lambda w.w \equiv \overline{false}$;
 - ▶ Se $n = FALSE$, então $\lambda z.\lambda w.\overline{n}wz \triangleright_{\beta} \lambda z.\lambda w.z \equiv \overline{true}$.
- ▶ Se $m = FALSE$, então $\lambda z.\lambda w.\overline{m}(\overline{n}wz)(\overline{n}zw) \triangleright_{\beta} \lambda z.\lambda w.\overline{n}zw$;
 - ▶ Se $n = TRUE$, então $\lambda z.\lambda w.\overline{n}zw \triangleright_{\beta} \lambda z.\lambda w.z \equiv \overline{true}$;
 - ▶ Se $n = FALSE$, então $\lambda z.\lambda w.\overline{n}zw \triangleright_{\beta} \lambda z.\lambda w.w \equiv \overline{false}$.

Booleanos de Church

XOR

Exemplos:

$$(\lambda x. \lambda y. \lambda z. \lambda w. x(ywz)(yzw)) \overline{true} \overline{false} \triangleright_{\beta}$$

$$\lambda z. \lambda w. \overline{true}((\overline{false})wz)((\overline{false}))zw \triangleright_{\beta}$$

$$\lambda z. \lambda w. (\overline{false})wz \triangleright_{\beta}$$

$$\lambda z. \lambda w. z \equiv \overline{true}$$

$$(\lambda x. \lambda y. \lambda z. \lambda w. x(ywz)(yzw)) \overline{false} \overline{false} \triangleright_{\beta}$$

$$\lambda z. \lambda w. \overline{false}((\overline{false})wz)((\overline{false}))zw \triangleright_{\beta}$$

$$\lambda z. \lambda w. (\overline{false})zw \triangleright_{\beta}$$

$$\lambda z. \lambda w. w \equiv \overline{false}$$

Booleanos de Church

IF

$$\overline{if} := \lambda x.\lambda y.\lambda z.xyz$$

É possível provar que:

$$\overline{if} \ \bar{e} \ \bar{m} \ \bar{n} \triangleright_{\beta} \bar{m} \text{ se } e = TRUE$$

$$\overline{if} \ \bar{e} \ \bar{m} \ \bar{n} \triangleright_{\beta} \bar{n} \text{ se } e = FALSE$$

De fato:

$$(\lambda x.\lambda y.\lambda z.xyz) \ \bar{e} \ \bar{m} \ \bar{n} \triangleright_{\beta} \bar{e} \ \bar{m} \ \bar{n}$$

- ▶ Se $e = TRUE$, então projeta m como resultado;
- ▶ Se $e = FALSE$, projeta n como resultado.

Booleanos de Church

IF

Exemplos:

$$\begin{aligned}
 (\lambda x. \lambda y. \lambda z. xyz) \overline{true} \overline{m} \overline{n} &\triangleright_{\beta} \overline{true} \overline{m} \overline{n} \\
 &\triangleright_{\beta} \overline{m}
 \end{aligned}$$

$$\begin{aligned}
 (\lambda x. \lambda y. \lambda z. xyz) \overline{false} \overline{m} \overline{n} &\triangleright_{\beta} \overline{false} \overline{m} \overline{n} \\
 &\triangleright_{\beta} \overline{n}
 \end{aligned}$$

Boleanos de Church

Exercícios

Avaliar:

- ▶ $\overline{\text{and true false}}$
- ▶ $\overline{\text{or false false}}$
- ▶ $\overline{\text{xor false true}}$
- ▶ $\overline{\text{or (and true false) true}}$;
- ▶ $\overline{\text{not (xor true (not true))}}$.

Construir expressões lambda para os operadores:

- ▶ Implicação (\Rightarrow) (dica: usar $\overline{\text{not}}$ e $\overline{\text{or}}$);
- ▶ Bi-implicação (\Leftrightarrow) (dica: usar $\overline{\text{and}}$).

Boleanos de Church

Expressões compostas - ZERO

Através da combinação das expressões anteriores, é possível representar funções mais complexas, que fazem uso de valores e operadores lógicos e aritméticos simultaneamente, como é o caso da função que testa se o argumento é zero e retorna \overline{true} ou \overline{false} :

$$\overline{zero} := \lambda x.x(\lambda y.\overline{false}) \overline{true}.$$

Booleanos de Church

Expressões compostas - ZERO

De fato, para $n = 0$:

$$\begin{aligned}
 (\lambda x.x(\lambda y.\overline{false}) \overline{true}) \overline{0} &\triangleright_{\beta} \\
 \overline{0} (\lambda y.\overline{false}) \overline{true} &\equiv \\
 \overline{false} (\lambda y.\overline{false}) \overline{true} &\triangleright_{\beta} \\
 \overline{true} &
 \end{aligned}$$

Booleanos de Church

Expressões compostas - ZERO

E para $n > 0$:

$$\begin{aligned}
 & (\lambda x.x(\lambda y.\overline{false}) \overline{true}) \overline{n} \triangleright_{\beta} \\
 & \quad \overline{n} (\lambda y.\overline{false}) \overline{true} \equiv \\
 & (\lambda z.\lambda w.z^n w)(\lambda y.\overline{false}) \overline{true} \triangleright_{\beta} \\
 & \quad (\lambda w.(\lambda y.\overline{false})^n w) \overline{true} \triangleright_{\beta} \\
 & (\lambda w.(\lambda y.\overline{false})^{n-1}((\lambda y.\overline{false})w)) \overline{true} \triangleright_{\beta} \\
 & \quad (\lambda w.(\lambda y.\overline{false})^{n-1} \overline{false}) \overline{true} \triangleright_{\beta} \\
 & \quad \dots \\
 & (\lambda w.(\lambda y.\overline{false}) \overline{false}) \overline{true} \triangleright_{\beta} \\
 & \quad (\lambda w.\overline{false}) \overline{true} \triangleright_{\beta} \\
 & \quad \quad \overline{false}
 \end{aligned}$$

Booleanos de Church

Expressões compostas - LEQ

Função que testa se o primeiro argumento é menor ou igual que o segundo:

$$\overline{leq} := \lambda x.\lambda y.\overline{zero} (\overline{sub} \ x \ y).$$

De fato:

$$\begin{aligned} (\lambda x.\lambda y.\overline{zero} (\overline{sub} \ x \ y)) \ \overline{m} \ \overline{n} &\triangleright_{\beta} \\ \overline{zero} (\overline{sub} \ \overline{m} \ \overline{n}) &\triangleright_{\beta} \\ \overline{zero} (\overline{m - n}) & \end{aligned}$$

Booleanos de Church

Exercícios

Desenvolver expressões lambda para os seguintes operadores relacionais:

- ▶ \overline{gt} (maior que);
- ▶ \overline{equal} (igualdade);
- ▶ $\overline{notequal}$ (desigualdade).

Definição

Um termo P é dito “ β -igual” ou “ β -conversível” a um termo Q , denotado $P =_{\beta} Q$ se e somente se Q puder ser obtido a partir de P por uma seqüência finita (eventualmente vazia) de contrações- β , contrações- β reversas e mudanças de variáveis ligadas.

Ou seja, $P =_{\beta} Q$ se e somente se existir $P_0, \dots, P_n (n \geq 0)$ tal que:

$$(\forall i \leq n - 1), (P_i \triangleright_{1\beta} P_{i+1}) \text{ ou } (P_{i+1} \triangleright_{1\beta} P_i) \text{ ou } (P_i \equiv_{\alpha} P_{i+1}),$$

$$P_0 \equiv P,$$

$$P_n \equiv Q.$$

Definição

Exemplo

Sejam $P \equiv (\lambda xyz.xzy)(\lambda xy.x)$ e $Q \equiv (\lambda xy.x)(\lambda x.x)$.

Então $P =_{\beta} Q$, ou seja:

$$(\lambda xyz.xzy)(\lambda xy.x) =_{\beta} (\lambda xy.x)(\lambda x.x)$$

De fato, pode-se notar inicialmente que:

$$\begin{aligned} (\lambda xyz.xzy)(\lambda xy.x) &\equiv_{\alpha} (\lambda xyz.xzy)(\lambda uv.u) \\ &\triangleright_{1\beta} \lambda yz.(\lambda uv.u)zy \\ &\triangleright_{1\beta} \lambda yz.(\lambda v.z)y \\ &\triangleright_{1\beta} \lambda yz.z \end{aligned}$$

Definição

Exemplo

Além disso, que:

$$\begin{aligned}
 (\lambda xy.x)(\lambda x.x) &\equiv_{\alpha} (\lambda xy.x)(\lambda w.w) \\
 &\triangleright_{1\beta} \lambda y.(\lambda w.w) \\
 &\equiv \lambda yw.w \\
 &\equiv_{\alpha} \lambda yz.z
 \end{aligned}$$

Definição

Exemplo

Finalmente, pode-se considerar $P_0, P_1, P_2, P_3, P_4, P_5, P_6, P_7$ tais que:

$$P = P_0 = (\lambda xyz.xzy)(\lambda xy.x)$$

$$P_1 = (\lambda xyz.xzy)(\lambda uv.u)$$

$$P_2 = \lambda yz.(\lambda uv.u)zy$$

$$P_3 = \lambda yz.(\lambda v.z)y$$

$$P_4 = \lambda yz.z$$

$$P_5 = \lambda yw.w$$

$$P_6 = (\lambda xy.x)(\lambda w.w)$$

$$Q = P_7 = (\lambda xy.x)(\lambda x.x)$$

Para concluir que $P =_{\beta} Q$, basta observar que $P_0 \equiv_{\alpha} P_1$, $P_1 \triangleright_{1\beta} P_2$, $P_2 \triangleright_{1\beta} P_3$, $P_3 \triangleright_{1\beta} P_4$, $P_4 \equiv_{\alpha} P_5$, $P_6 \triangleright_{1\beta} P_5$ e $P_6 \equiv_{\alpha} P_7$.

Lemas

Lema: $P =_{\beta} Q, P \equiv_{\alpha} P', Q \equiv_{\alpha} Q' \Rightarrow P' =_{\beta} Q'$.

Lema: $M =_{\beta} M', N =_{\beta} N' \Rightarrow [N/x]M =_{\beta} [N'/x]M'$.

Teorema de Church-Rosser para $=_{\beta}$

Se $P =_{\beta} Q$, então existe um termo T tal que:

$$P \triangleright_{\beta} T \quad \text{e} \quad Q \triangleright_{\beta} T.$$

Dois termos β -conversíveis representam a mesma operação, uma vez que ambos podem ser reduzidos para o mesmo termo.

Corolários

Corolário: Se $P =_{\beta} Q$ e Q é uma forma normal- β , então $P \triangleright_{\beta} Q$

Corolário: Se $P =_{\beta} Q$, então ambos P e Q possuem a mesma forma normal- β ou então nenhum dos dois possui nenhuma forma normal- β .

Corolário: Se P e Q são formas normais- β e $P =_{\beta} Q$, então $P \equiv_{\alpha} Q$.

Corolário (unicidade da forma normal): Um termo é β -igual a no máximo uma forma normal- β , a menos de mudanças de variáveis ligadas.

Ponto fixo

Definição

O “ponto fixo” de uma função f é o valor x tal que $f(x) = x$. Ou seja, corresponde ao objeto que não se modifica pela aplicação da função. Uma função pode ter um, mais de um ou nenhum ponto fixo.

- ▶ A função $f(x) = x^2$ possui como pontos fixos os valores 0 e 1, pois $0^2 = 0$ e $1^2 = 1$;
- ▶ A função $f(x) = x + 1$ não possui nenhum ponto fixo, pois $f(x) \neq x, \forall x \geq 0$;
- ▶ A função $f(x) = x * 2$ possui como ponto fixo apenas o valor 0, pois $0 * 2 = 0$.

Ponto fixo

Existência

No Cálculo Lambda, é possível demonstrar que todo termo possui um ponto fixo. Ou seja, para todo termo X existe um termo P , dependente de X , tal que:

$$XP =_{\beta} P.$$

Além disso, esse ponto fixo pode ser obtido pela aplicação de um operador Y , de tal forma que, para todo termo X , YX é um ponto fixo de X .

Ponto fixo

Teorema

No Cálculo Lambda, existe um operador Y tal que, para todo termo x ,

$$Yx =_{\beta} x(Yx) \quad \text{e, mais forte ainda,} \quad Yx \triangleright_{\beta} x(Yx).$$

Prova:

- ▶ Basta considerar $Y \equiv UU$, com $U \equiv \lambda ux.x(ux)$.

Esse termo foi inventado por Alan Turing em 1937, e também é conhecido por Y_{Turing} .

Ponto fixo

Teorema

De fato:

$$\begin{aligned}
 Yx &\equiv (\lambda u. \lambda x. x(ux))Ux \\
 \triangleright_{\beta} & [U/u](\lambda x. x(ux))x \\
 &\equiv (\lambda x. x(UUx))x \\
 \triangleright_{\beta} & x(UUx) \\
 &\equiv x(Yx)
 \end{aligned}$$

Ponto fixo

Teorema

Y_{Turing} não é o único operador de ponto fixo conhecido. Existem outros, entre os quais $Y_{Curry-Ros}$, inventado por Haskell Curry:

$$Y_{Curry-Ros} \equiv \lambda x.VV, V \equiv \lambda y.x(yy)$$

Exercício: provar que $Y_{Curry-Ros}$ é um operador de ponto fixo, ou seja, que $YX =_{\beta} X(YX)$ para todo termo X .

Ponto fixo

Corolário

O seguinte corolário decorre do teorema anterior: no Cálculo Lambda, para todo termo Z e $n \geq 0$, a equação $xy_1y_2\dots y_n = Z$, onde x, y_1, y_2, \dots, y_n são variáveis, pode ser resolvida para x . Ou seja, existe um termo X tal que:

$$[X/x](xy_1y_2\dots y_n) = [X/x]Z$$

ou seja:

$$Xy_1y_2\dots y_n = [X/x]Z.$$

Prova:

Basta escolher $X \equiv Y(\lambda xy_1y_2\dots y_n.Z)$.

Ponto fixo

Corolário

De fato:

$$\begin{aligned}
 Xy_1y_2\dots y_n &\equiv \underbrace{Y(\lambda xy_1y_2\dots y_n.Z)}_{YF} y_1y_2\dots y_n \\
 \triangleright_{\beta} &\underbrace{(\lambda xy_1y_2\dots y_n.Z)(Y(\lambda xy_1y_2\dots y_n.Z))}_{F(YF)} y_1y_2\dots y_n \\
 \triangleright_{\beta} &(\lambda y_1y_2\dots y_n.[Y(\lambda xy_1y_2\dots y_n.Z)/x]Z)y_1y_2\dots y_n \\
 &\equiv (\lambda y_1y_2\dots y_n.[X/x]Z)y_1y_2\dots y_n \\
 \triangleright_{\beta} &(\lambda y_2\dots y_n.[X/x]Z)y_2\dots y_n \\
 &\dots \\
 \triangleright_{\beta} &[X/x]Z
 \end{aligned}$$

Ponto fixo

Aplicação

A importância desse corolário reside no fato de que ele permite a resolução de formulações recursivas, sendo útil na construção de termos lambda que representam as funções correspondentes. É importante observar:

- ▶ No Cálculo Lambda as funções são anônimas, ou seja, elas não são identificadas;
- ▶ Dessa maneira, não é possível representar diretamente funções que invocam a si mesmas;
- ▶ No entanto, a partir da equação de define a função recursiva em questão, o uso operador de ponto fixo permite obter a expressão lambda que representa tal função.

Definições recursivas

Exemplo - consumir argumento

Considere a definição recursiva:

$$xy =_{\beta} x.$$

Essa equação representa uma função que “engole” o seu argumento. Pela aplicação do corolário, a solução em x dessa equação é dada por:

$$X \equiv Y(\lambda xy_1.Z) \equiv Y(\lambda xy.x)$$

Para isso, basta considerar $y_1 = y$ e $Z = x$. De fato:

$$\begin{aligned} Y(\lambda xy.x)y &\triangleright_{\beta} (\lambda xy.x)(Y(\lambda xy.x))y \\ &\triangleright_{\beta} (\lambda y.(Y(\lambda xy.x)))y \\ &\triangleright_{\beta} Y(\lambda xy.x) \end{aligned}$$

Definições recursivas

Exemplo - inverter argumentos

Considere a definição recursiva:

$$xyz =_{\beta} xzy.$$

Essa equação representa uma função que inverte a ordem dos seus dois argumentos. Pela aplicação do corolário, a solução em x dessa equação é dada por:

$$X \equiv Y(\lambda xy_1y_2.Z) \equiv Y(\lambda xyz.xzy)$$

Para isso, basta considerar $y_1 = y$, $y_2 = z$ e $Z = xzy$. De fato:

$$\begin{aligned} Y(\lambda xyz.xzy)yz &\triangleright_{\beta} (\lambda xyz.xzy)(Y(\lambda xyz.xzy))yz \\ &\triangleright_{\beta} Y(\lambda xyz.xzy)zy \end{aligned}$$

Definições recursivas

Exemplo - fatorial

Exemplo clássico de definição recursiva:

$$fat(n) = \begin{cases} 1 & \text{se } n = 0; \\ n * fat(n - 1) & \text{se } n > 0. \end{cases}$$

Essa equação pode ser escrita como:

$$xy =_{\beta} \overline{if} (\overline{zero} \ y) \ \overline{I} (\overline{mult} \ y \ (x \ (\overline{sub} \ y \ \overline{I})))$$

Definições recursivas

Exemplo - fatorial

A solução em x é obtida considerando-se $Y(\lambda xy_1.Z)$, onde:

$$\begin{aligned} y_1 &= y \\ Z &= \overline{if} (\overline{zero} y) \overline{1} (\overline{mult} y (x (\overline{sub} y \overline{1}))) \end{aligned}$$

ou seja:

$$\overline{fat} \equiv Y(\lambda xy. \underbrace{\overline{if} (\overline{zero} y) \overline{1} (\overline{mult} y (x (\overline{sub} y \overline{1})))}_{Z}}_{F}) \equiv YF$$

Definições recursivas

Exemplo - fatorial

Exemplo: $fat(3)$

$$YF \bar{3} \triangleright_{\beta}$$

$$F(YF) \bar{3} \triangleright_{\beta}$$

$$(\lambda y. \overline{if} (\overline{zero} y) \bar{1} (\overline{mult} y ((YF)(\overline{sub} y \bar{1})))) \bar{3} \triangleright_{\beta}$$

$$\overline{if} (\overline{zero} \bar{3}) \bar{1} (\overline{mult} \bar{3} ((YF)(\overline{sub} \bar{3} \bar{1}))) \triangleright_{\beta}$$

$$\overline{mult} \bar{3} ((YF) \bar{2}) \triangleright_{\beta}$$

$$\overline{mult} \bar{3} (F(YF) \bar{2}) \triangleright_{\beta}$$

$$\overline{mult} \bar{3} (\overline{mult} \bar{2} ((YF) \bar{1})) \triangleright_{\beta}$$

Definições recursivas

Exemplo - fatorial

continuação:

$$\begin{aligned}
 & \overline{\text{mult}} \ 3 \ (\overline{\text{mult}} \ 2 \ (F(YF) \ \overline{1})) \quad \triangleright_{\beta} \\
 & \overline{\text{mult}} \ 3 \ (\overline{\text{mult}} \ 2 \ (\overline{\text{mult}} \ 1 \ ((YF) \ \overline{0}))) \quad \triangleright_{\beta} \\
 & \overline{\text{mult}} \ 3 \ (\overline{\text{mult}} \ 2 \ (\overline{\text{mult}} \ 1 \ (F(YF) \ \overline{0}))) \quad \triangleright_{\beta} \\
 & \overline{\text{mult}} \ 3 \ (\overline{\text{mult}} \ 2 \ (\overline{\text{mult}} \ 1 \ \overline{1})) \quad \triangleright_{\beta} \\
 & \overline{\text{mult}} \ 3 \ (\overline{\text{mult}} \ 2 \ \overline{1}) \quad \triangleright_{\beta} \\
 & \overline{\text{mult}} \ 3 \ \overline{2} \quad \triangleright_{\beta} \\
 & \overline{6}
 \end{aligned}$$

Definições recursivas

Exercício

Obter uma expressão lambda que calcula o n -ésimo termo da seqüência de Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, ...

$$\left\{ \begin{array}{l} f(1) = 0; \\ f(2) = 1; \\ f(n) = f(n-1) + f(n-2) \text{ para } n \geq 3. \end{array} \right.$$

História

- ▶ Os primeiros resultados acerca da indecidibilidade em toda a história foram descobertos por Church através do Cálculo Lambda;
- ▶ Eles tratam da inexistência de algoritmos para determinar (i) se duas expressões lambdas satisfazem à relação $=_{\beta}$ (isto é, se elas representam a mesma operação) e (ii) determinar se uma expressão lambda possui forma normal ou não;
- ▶ A partir desses resultados, Church foi capaz de deduzir a indecidibilidade da lógica de predicados pura de primeira ordem em 1936.

História

- ▶ Com isso, ele respondeu uma questão formulada por David Hilbert anos antes (*Entscheidungsproblem*);
- ▶ *Entscheidungsproblem*: determinar a existência de um algoritmo que decide se uma certa afirmação pode ser provada a partir de axiomas usando as regras da lógica;
- ▶ A demonstração a seguir foi feita por Dana Scott em 1963 e redescoberta independentemente por Curry em 1972.

Número de Gödel

- ▶ Para um certo termo X , o “número de Gödel” desse termo, denotado,

$$gd(X)$$

é o numero natural que representa, de forma unívoca, o termo X ;

- ▶ A codificação é feita de tal forma que X pode ser obtido a partir de $gd(X)$, ou seja, a função de codificação é injetora;
- ▶ O algoritmo usado por Gödel é baseado do Teorema Fundamental da Aritmética, que atribui números diferentes para os símbolos do alfabeto, e depois utiliza o produto de números primos para obter uma representação única para cada sentença formada por tais símbolos;
- ▶ Existem outras maneiras de se fazer tal codificação.

Numeral de Church do Número de Gödel

O “Numeral de Church do Número de Gödel” de um termo X , denotado:

$$[X]$$

é definido como sendo o Numeral de Church correspondente ao Número de Gödel associado ao termo X , ou seja,

$$[X] \equiv \overline{gd(X)}$$

Numeral de Church do Número de Gödel

Observar que:

- ▶ X é um termo;
- ▶ $gd(X)$ é um número natural;
- ▶ $[X]$ é um termo;

Exemplo:

Suponha que $X \equiv uv$ e que $gd(uv) = 5$. Então:

$$[uv] \equiv \lambda x.\lambda y.x(x(x(xy))))$$

.

Funções auxiliares

Admite-se a existência das seguintes funções auxiliares:

- ▶ $\tau(gd(X), gd(Y)) = gd((XY))$;
Obtém o Número de Gödel de uma expressão composta a partir da composição dos Números de Gödel das expressões componentes;
- ▶ $\nu(n) = gd(\bar{n})$;
Obtém o Número de Gödel de um número inteiro representado como Numeral de Church.

E considera-se que as mesmas sejam representadas, respectivamente, por expressões lambda correspondentes:

- ▶ $\bar{\tau} := T$, de tal forma que $T[X][Y] =_{\beta} [(XY)]$;
- ▶ $\bar{\nu} := N$, de tal forma que $N\bar{n} =_{\beta} [\bar{n}]$.

Conjuntos recursivamente separáveis

Sejam \mathcal{A} e \mathcal{B} dois conjuntos de números naturais quaisquer. Esse par de conjuntos é dito “recursivamente separável” se e somente se existir uma função total ϕ , definida sobre o domínio $\mathcal{A} \cup \mathcal{B}$, tal que:

$$n \in \mathcal{A} \Rightarrow \phi(n) = 0$$

$$n \in \mathcal{B} \Rightarrow \phi(n) = 1$$

ou seja, o par \mathcal{A} e \mathcal{B} é recursivamente separável se e somente se (i) $\mathcal{A} \cap \mathcal{B} = \emptyset$ e (ii) existir um algoritmo que decida se um número qualquer pertencente à $\mathcal{A} \cup \mathcal{B}$ pertence à \mathcal{A} ou \mathcal{B} .

Conjuntos recursivamente separáveis

Sejam \mathcal{P} e \mathcal{Q} dois conjuntos de expressões lambda quaisquer e suponha que:

$$X \in \mathcal{P} \Leftrightarrow gd(X) \in \mathcal{A}$$

e

$$X \in \mathcal{Q} \Leftrightarrow gd(X) \in \mathcal{B}$$

ou seja, \mathcal{A} e \mathcal{B} contém, respectivamente, os Números de Gödel das expressões contidas em \mathcal{P} e \mathcal{Q} .

- ▶ Diz-se que \mathcal{P} e \mathcal{Q} são recursivamente separáveis se e somente se \mathcal{A} e \mathcal{B} também o forem;
- ▶ Um conjunto \mathcal{A} (de expressões ou de números) é dito “recursivo” ou “decidível” se \mathcal{A} e o seu complemento forem recursivamente separáveis.

Fechamento na igualdade- β

Um conjunto de termos \mathcal{A} é dito “fechado em relação à operação de conversão (ou igualdade)” se e somente se, para todos os termos X e Y :

$$(X \in \mathcal{A} \text{ e } X =_{\beta} Y) \Rightarrow (Y \in \mathcal{A})$$

Teorema da Indecidibilidade de Scott-Curry

“Nenhum par de conjuntos de termos lambda que sejam (i) não-vazios e (ii) fechados em relação à operação de conversão, é recursivamente separável.”

Teorema da Indecidibilidade de Scott-Curry

Sejam \mathcal{A} e \mathcal{B} dois conjuntos de termos não-vazios e fechados em relação à operação de conversão. Suponhamos que eles sejam recursivamente separáveis, ou seja, que exista uma função total ϕ tal que:

$$X \in \mathcal{A} \Rightarrow \phi(gd(X)) = 0$$

$$X \in \mathcal{B} \Rightarrow \phi(gd(X)) = 1$$

Logo¹, existe um termo F que representa ϕ . Ou seja,

$$X \in \mathcal{A} \Rightarrow F[X] =_{\beta} \bar{0}$$

$$X \in \mathcal{B} \Rightarrow F[X] =_{\beta} \bar{1}$$

¹A equivalência entre funções recursivas totais e expressões lambda é demonstrada no capítulo 4 do livro de Hindley/Seldin.

Teorema da Indecidibilidade de Scott-Curry

Sejam A e B dois termos tais que $A \in \mathcal{A}$ e $B \in \mathcal{B}$ e considere-se o termo J , construído a partir de A e B , de seguinte forma:

$$\begin{aligned} J &\equiv H[H] \\ H &\equiv \lambda y. \mathbf{D}BA(F(Ty(Ny))) \end{aligned}$$

com $\mathbf{D} \equiv \lambda xyz.z(\mathbf{K}y)x$ e $\mathbf{K} \equiv \lambda xy.x$. \mathbf{D} recebe três argumentos e retorna o primeiro se o terceiro for 0 ou o segundo se o terceiro for 1 (na verdade, qualquer valor maior que 0). Ou seja,

$$\mathbf{D}BA\bar{0} =_{\beta} B$$

$$\mathbf{D}BA\bar{1} =_{\beta} A$$

Teorema da Indecidibilidade de Scott-Curry

De fato:

$$\begin{aligned}
 \mathbf{D}BA\bar{0} &\equiv (\lambda xyz.z(\mathbf{K}y)x)BA\bar{0} \\
 &=_{\beta} \bar{0}(\mathbf{K}A)B \\
 &=_{\beta} B
 \end{aligned}$$

$$\begin{aligned}
 \mathbf{D}BA\bar{1} &\equiv (\lambda xyz.z(\mathbf{K}y)x)BA\bar{1} \\
 &=_{\beta} \bar{1}(\mathbf{K}A)B \\
 &\equiv (\lambda x.\lambda y.xy)(\mathbf{K}A)B \\
 &=_{\beta} (\lambda y.(\mathbf{K}A)y)B \\
 &=_{\beta} (\mathbf{K}A)B \\
 &=_{\beta} A
 \end{aligned}$$

Teorema da Indecidibilidade de Scott-Curry

Por outro lado, é possível demonstrar (slide seguinte) que:

$$J =_{\beta} \mathbf{DBA}(F[J])$$

Logo, $J =_{\beta} A$ ou $J =_{\beta} B$. Como $A \in \mathcal{A}$ e $B \in \mathcal{B}$ e, além disso, \mathcal{A} e \mathcal{B} são fechados em relação à igualdade beta, podemos concluir que o termo J deve necessariamente pertencer à um destes dois conjuntos. Ou seja, o termo J pertence ao domínio da função $\phi(F)$.

Teorema da Indecidibilidade de Scott-Curry

$$\begin{aligned}
J &\equiv H[H] \\
&\equiv (\lambda y. \mathbf{DBA}(F(Ty(Ny))))[H] \\
&=_{\beta} \mathbf{DBA}(F(T[H](N[H]))) \\
&=_{\beta} \mathbf{DBA}(F(T[H] \underbrace{[[H]]}_2)) \\
&=_{\beta} \mathbf{DBA}(F \underbrace{[H[H]]}_3) \\
&\equiv \mathbf{DBA}(\underbrace{F[J]}_4)
\end{aligned}$$

²Pela definição de N .

³Pela definição de T .

⁴Pela definição de J .

Teorema da Indecidibilidade de Scott-Curry

Seja $j = gd(J)$.

Então, de acordo com a hipótese, $\phi(j) = 0$ ou $\phi(j) = 1$.

Suponhamos que $\phi(j) = 0$.

Logo:

$$\begin{aligned}
 \phi(j) = 0 &\Rightarrow F[J] =_{\beta} \bar{0} \\
 &\Rightarrow J =_{\beta} B \\
 &\Rightarrow J \in \mathcal{B} \\
 &\Rightarrow \phi(j) = 1
 \end{aligned}$$

Teorema da Indecidibilidade de Scott-Curry

Suponhamos, por outro lado, que $\phi(j) = 1$.

Portanto:

$$\begin{aligned}\phi(j) = 1 &\Rightarrow F[J] =_{\beta} \bar{1} \\ &\Rightarrow J =_{\beta} A \\ &\Rightarrow J \in \mathcal{A} \\ &\Rightarrow \phi(j) = 0\end{aligned}$$

Teorema da Indecidibilidade de Scott-Curry

Conclusão:

- ▶ Qualquer que seja o caso ($\phi(j) = 0$ ou $\phi(j) = 1$) temos uma contradição;
- ▶ Logo, a hipótese não pode ser verdadeira e os conjuntos \mathcal{A} e \mathcal{B} não são recursivamente separáveis.

Teorema da Indecidibilidade de Scott-Curry

Corolários

- ▶ *Se \mathcal{A} é um conjunto de termos fechado em relação à operação de conversão, e tanto \mathcal{A} quanto o seu complemento são não-vazios, então \mathcal{A} não é decidível;*
Basta considerar como \mathcal{B} o complemento de \mathcal{A} , ou seja, o conjunto dos termos que não pertencem à \mathcal{A} ;
 \mathcal{A} e \mathcal{B} são fechados em relação à operação de conversão e não-vazios.

Conclusão:

Não existe algoritmo para determinar se um certo termo pertence ou não pertence à um conjunto de termos fechado em relação à operação de conversão.

Teorema da Indecidibilidade de Scott-Curry

Corolários

- ▶ *O conjunto de todos os termos que possuem forma normal não é decidível;*

Basta considerar esse conjunto como \mathcal{A} e o seu complemento (termos que não possuem formas normais) como \mathcal{B} ;

\mathcal{A} e \mathcal{B} são fechados em relação à operação de conversão e não-vazios.

Conclusão:

Não existe algoritmo para determinar se um certo termo possui ou não possui forma normal.

Teorema da Indecidibilidade de Scott-Curry

Corolários

- A relação $=_{\beta}$ não é decidível, ou seja, não existe função total ψ tal que:

$$X =_{\beta} Y \Rightarrow \psi(\text{gd}(X), \text{gd}(Y)) = 0$$

$$X \neq_{\beta} Y \Rightarrow \psi(\text{gd}(X), \text{gd}(Y)) = 1$$

Basta considerar \mathcal{A} como o conjunto de termos conversível a um certo termo em particular (por exemplo, Y), e \mathcal{B} como o seu complemento; \mathcal{A} e \mathcal{B} são fechados em relação à operação de conversão e não-vazios.

Conclusão:

Não existe algoritmo para determinar se dois termos representam a mesma operação.