

Combinadores

Base SK

Prof. Edson Alves

Campus UnB Gama: Faculdade de Ciências e Tecnologias em Engenharia

Base SK

Base SK



Estorninho (*starling*)

Base SK



Estorninho (*starling*)



Cernícalo americano (*kestrel*)

Combinador *I*

Combinador I

- ★ Das definições de I e K , segue que

$$Ix = x = Kxy$$

Combinador I

- ★ Das definições de I e K , segue que

$$Ix = x = Kxy$$

- ★ Como y é arbitrário, ele pode ser substituído por Kx :

$$Ix = Kx(Kx)$$

Combinador I

- ★ Das definições de I e K , segue que

$$Ix = x = Kxy$$

- ★ Como y é arbitrário, ele pode ser substituído por Kx :

$$Ix = Kx(Kx)$$

- ★ Da definição de S , temos que

$$I = SKK$$

Combinador I

- ★ Das definições de I e K , segue que

$$Ix = x = Kxy$$

- ★ Como y é arbitrário, ele pode ser substituído por Kx :

$$Ix = Kx(Kx)$$

- ★ Da definição de S , temos que

$$I = SKK$$

- ★ Note que o segundo K pode ser substituído por uma função arbitrária

Combinador I

- ★ Das definições de I e K , segue que

$$Ix = x = Kxy$$

- ★ Como y é arbitrário, ele pode ser substituído por Kx :

$$Ix = Kx(Kx)$$

- ★ Da definição de S , temos que

$$I = SKK$$

- ★ Note que o segundo K pode ser substituído por uma

função arbitrária



Albatroz de cauda curta (*'idiot bird'*)

Combinador *B*

Combinador *B*



Tordo-azul-oriental (*bluebird*)

Combinador *B*



★ Representado por Schönfinkel pela letra *Z*

Tordo-azul-oriental (*bluebird*)

Combinador B



Tordo-azul-oriental (*bluebird*)

- ★ Representado por Schönfinkel pela letra Z
- ★ Pela definição,

$$Bfgx = f(gx)$$

Combinador B



Tordo-azul-oriental (*bluebird*)

★ Representado por Schönfinkel pela letra Z

★ Pela definição,

$$Bfgx = f(gx)$$

★ Observe que

$$f(gx) = (Kfx)(gx) = S(Kf)gx = (KSf)(Kf)gx$$

Combinador B



Tordo-azul-oriental (*bluebird*)

★ Representado por Schönfinkel pela letra Z

★ Pela definição,

$$Bfgx = f(gx)$$

★ Observe que

$$f(gx) = (Kfx)(gx) = S(Kf)gx = (KSf)(Kf)gx$$

★ Uma nova fusão em f nos leva a

$$f(gx) = S(KS)Kfgx$$

Combinador B



Tordo-azul-oriental (*bluebird*)

★ Representado por Schönfinkel pela letra Z

★ Pela definição,

$$Bfgx = f(gx)$$

★ Observe que

$$f(gx) = (Kfx)(gx) = S(Kf)gx = (KSf)(Kf)gx$$

★ Uma nova fusão em f nos leva a

$$f(gx) = S(KS)Kfgx$$

★ Portanto, $B = S(KS)K$

Combinador C

Combinador C

- ★ Corresponde ao combinador T de Schönfinkel



Cardeal-do-norte (*cardinal*)

Combinador C

★ Corresponde ao combinador T de Schönfinkel

★ Vale a seguinte sequência de transformações:

$$\begin{aligned} fxy &= fx(Kyx) = (fx)(Kyx) = Sf(Ky)x \\ &= (Sf)(Ky)y = B(Sf)Kyx = (Sf)(Ky)y \\ &= B(Sf)Kyx = BBSfKyx = (BBSf)Kyx \\ &= (BBSf)(KKf)yx = S(BBS)(KK)fyx \end{aligned}$$



Cardeal-do-norte (*cardinal*)

Combinador C

★ Corresponde ao combinador T de Schönfinkel

★ Vale a seguinte sequência de transformações:

$$\begin{aligned} fxy &= fx(Kyx) = (fx)(Kyx) = Sf(Ky)x \\ &= (Sf)(Ky)y = B(Sf)Kyx = (Sf)(Ky)y \\ &= B(Sf)Kyx = BBSfKyx = (BBSf)Kyx \\ &= (BBSf)(KKf)yx = S(BBS)(KK)fyx \end{aligned}$$

★ Portanto, $C = S(BBS)(KK)$



Cardeal-do-norte (*cardinal*)

Função de incompatibilidade U

Funções de incompatibilidade

O conectivo fundamental de Schönfinkel

$$fx \mid^x gx,$$

depende de duas funções proposicionais de um argumento f e g , logo tem forma $U(f, g)$. Assim, usando a transformação de funções de múltiplos argumentos,

$$Ufg = fx \mid^x gx,$$

é a equação que define a função de incompatibilidade U .

Principais resultados do artigo

Principais resultados do artigo

- ★ Schönfinkel afirma que qualquer fórmula lógica de primeira ordem pode ser expressa em termos das funções particulares I, C, T, Z, S e U .

Principais resultados do artigo

- ★ Schönfinkel afirma que qualquer fórmula lógica de primeira ordem pode ser expressa em termos das funções particulares I, C, T, Z, S e U .
- ★ Dadas as reduções já apresentadas, de fato qualquer fórmula lógica de primeira ordem pode ser expressa em termos C, S e U .

Principais resultados do artigo

- ★ Schönfinkel afirma que qualquer fórmula lógica de primeira ordem pode ser expressa em termos das funções particulares I, C, T, Z, S e U .
- ★ Dadas as reduções já apresentadas, de fato qualquer fórmula lógica de primeira ordem pode ser expressa em termos C, S e U .
- ★ Schönfinkel enuncia, mas não prova, este resultado. Ele dá apenas exemplos destas representações.

Principais resultados do artigo

- ★ Schönfinkel afirma que qualquer fórmula lógica de primeira ordem pode ser expressa em termos das funções particulares I, C, T, Z, S e U .
- ★ Dadas as reduções já apresentadas, de fato qualquer fórmula lógica de primeira ordem pode ser expressa em termos C, S e U .
- ★ Schönfinkel enuncia, mas não prova, este resultado. Ele dá apenas exemplos destas representações.
- ★ Ele esteve muito próximo de um resultado fundamental, mas não enunciou: “Qualquer combinador pode ser expresso em termos de S e K apenas (base SK)”.

Principais resultados do artigo

Principais resultados do artigo

- ★ Na última seção, Schönfinkel introduz a função J , definida pelas igualdades

$$JC = U, \quad JS = C, \quad \text{e} \quad Jx = S,$$

onde x é qualquer termo diferente de C e S .

Principais resultados do artigo

- * Na última seção, Schönfinkel introduz a função J , definida pelas igualdades

$$JC = U, \quad JS = C, \quad \text{e} \quad Jx = S,$$

onde x é qualquer termo diferente de C e S .

- * Ele argumenta que J é diferente de C e S , pois tem apenas 3 valores, enquanto que C e S têm infinitos.

Principais resultados do artigo

- ★ Na última seção, Schönfinkel introduz a função J , definida pelas igualdades

$$JC = U, \quad JS = C, \quad \text{e} \quad Jx = S,$$

onde x é qualquer termo diferente de C e S .

★ Ele argumenta que J é diferente de C e S , pois tem apenas 3 valores, enquanto que C e S têm infinitos.

- ★ Daí ele apresenta as seguintes reduções:

$$JJ = S, \quad J(JJ) = JS = C \quad \text{e} \quad J[J(JJ)] = JC = U$$

Principais resultados do artigo

- * Na última seção, Schönfinkel introduz a função J , definida pelas igualdades

$$JC = U, \quad JS = C, \quad \text{e} \quad Jx = S,$$

onde x é qualquer termo diferente de C e S .

- * Ele argumenta que J é diferente de C e S , pois tem apenas 3 valores, enquanto que C e S têm infinitos.

- * Daí ele apresenta as seguintes reduções:

$$JJ = S, \quad J(JJ) = JS = C \quad \text{e} \quad J[J(JJ)] = JC = U$$

- * Portanto, qualquer forma lógica pode ser expressão apenas por J e parêntesis.

Combinadores

Definição de combinador

Sejam $x_1, x_2, x_3, \dots, x_N$ termos sem quaisquer restrições. Um combinador \mathcal{C} é uma função destas variáveis tal que seu valor depende única e exclusivamente de aplicação destes termos e de combinadores previamente definidos.

Cálculo *SK*

Cálculo *SK*

- ★ É um sistema computacional minimalista

Cálculo *SK*

- ★ É um sistema computacional minimalista
- ★ Todos os objetos são funções unárias

Cálculo *SK*

- ★ É um sistema computacional minimalista
- ★ Todos os objetos são funções unárias
- ★ Há apenas duas primitivas: *S* e *K*

Cálculo *SK*

- ★ É um sistema computacional minimalista
- ★ Todos os objetos são funções unárias
- ★ Há apenas duas primitivas: *S* e *K*
- ★ Os parêntesis, que completam a sintaxe, são usados apenas para alterar a ordem de avaliação das expressões

Cálculo *SK*

- ★ É um sistema computacional minimalista
- ★ Todos os objetos são funções unárias
- ★ Há apenas duas primitivas: *S* e *K*
- ★ Os parêntesis, que completam a sintaxe, são usados apenas para alterar a ordem de avaliação das expressões
- ★ A avaliação é feita da esquerda para a direita (formalmente, redução beta)

Redução beta

Redução beta

- ★ Inicia aplicando a subexpressão mais à esquerda, se houverem argumentos suficiente

Redução beta

- ★ Inicia aplicando a subexpressão mais à esquerda, se houverem argumentos suficiente
- ★ Subexpressões à direita não são avaliadas (*lazy evaluation*)

Redução beta

- ★ Inicia aplicando a subexpressão mais à esquerda, se houverem argumentos suficiente
- ★ Subexpressões à direita não são avaliadas (*lazy evaluation*)
- ★ Isto ocorre para evitar que termos que seriam descartados não se expandam infinitamente ou que não terminem

Redução beta

- ★ Inicia aplicando a subexpressão mais à esquerda, se houverem argumentos suficiente
- ★ Subexpressões à direita não são avaliadas (*lazy evaluation*)
- ★ Isto ocorre para evitar que termos que seriam descartados não se expandam infinitamente ou que não terminem
- ★ Por exemplo, a redução beta de $SII(SII)$ não termina, mas

$$KS(SII(SII)) = S$$

Combinador M

Combinador *M*



Tordo-imitador (*mockingbird*)

Combinador M

- ★ Por definição, $Mx = xx$



Tordo-imitador (*mockingbird*)

Combinador M

★ Por definição, $Mx = xx$

★ Observe que:

$$Ma = aa = Ia(Ia) = SIIa$$



Tordo-imitador (*mockingbird*)

Combinador M

★ Por definição, $Mx = xx$

★ Observe que:

$$Ma = aa = Ia(Ia) = SIIa$$

★ Portanto, $M = SII$



Tordo-imitador (*mockingbird*)

Combinador M

★ Por definição, $Mx = xx$

★ Observe que:

$$Ma = aa = Ia(Ia) = SIIa$$

★ Portanto, $M = SII$

★ A redução beta de MM não termina



Tordo-imitador (*mockingbird*)

To Mock a Mockingbird

To Mock a Mockingbird

Definições:

To Mock a Mockingbird

Definições:

- ★ As letras maiúsculas (A, B, \dots) são pássaros

To Mock a Mockingbird

Definições:

- ★ As letras maiúsculas (A, B, \dots) são pássaros

- ★ Quando o pássaro A ouve o nome do pássaro B , ele responde o nome do pássaro AB

To Mock a Mockingbird

Definições:

- ★ As letras maiúsculas (A, B, \dots) são pássaros
- ★ Quando o pássaro A ouve o nome do pássaro B , ele responde o nome do pássaro AB
- ★ Em geral, $AB \neq BA$

To Mock a Mockingbird

Definições:

- ★ As letras maiúsculas (A, B, \dots) são pássaros
- ★ Quando o pássaro A ouve o nome do pássaro B , ele responde o nome do pássaro AB
- ★ Em geral, $AB \neq BA$
- ★ Para três pássaros quaisquer A, B, C , temos que $(AB)C$ e $(AC)B$ não são, necessariamente, iguais