Redução do Sujeito para Cálculos de Substituições Explícitas com Tipos Simples

Daniel Lima Ventura

Mestrando - Departamento de Matemática/UnB
Bolsista CNPq

Orientador: Mauricio Ayala-Rincón

2 de dezembro de 2005

III Seminário Informal(, mas Formal!) do Grupo de Teoria da Computação

Roteiro

Aplicações

- Usados na concepção e implementação de linguagens de programação
- Deteção de erros previamente à execução
- Tratamento de elementos sofisticados como classes e objetos (polimorfismo)
- Extração de programas de provas (Correspondência de Curry-Howard)

História

- Tratamento de paradoxos e inconsistências na formalização da matemática:
  - Auto-reprodução
- Tipos simples no λ-calculus [Church 1940]
- Tipos implícitos [Curry 1958]
- Linguagens tipadas: Fortran, Algol,...
- Linguagens com tipos implícitos ML [Milner 1980]

Redução do Sujeito

- Verificação de que qualquer computação preserva tipos dos objetos transformados

  Se $f(x)$ é do tipo $\mathbf{int} \rightarrow \mathbf{int}$, então $f(n)$ tem o mesmo tipo.

O λ-Calculus

- Introduzido por Alonzo Church [Chu32]

O λ-Calculus

- Introduzido por Alonzo Church [Chu32]
- Kleene e Rosser demonstraram que todas as funções recursivas parciais podem ser codificadas no λ-calculus [KR35]

O λ-Calculus

- Introduzido por Alonzo Church [Chu32]
- Kleene e Rosser demonstraram que todas as funções recursivas parciais podem ser codificadas no λ-calculus [KR35]
- Alan Turing mostrou que as máquinas de Turing e o λ-calculus são equivalentes (Tese de Church-Turing)

O λ-Calculus

- Introduzido por Alonzo Church [Chu32]
- Kleene e Rosser demonstraram que todas as funções recursivas parciais podem ser codificadas no λ-calculus [KR35]
- Alan Turing mostrou que as máquinas de Turing e o λ-calculus são equivalentes (Tese de Church-Turing)
- Alguns sistemas computacionais de ordem superior são baseados no λ-calculus, como λ-Prolog, Haskell, Lisp e ML.

Sintaxe

TERMOS $\quad a ::= x \mid (a\ a) \mid \lambda x.a$

Sintaxe

TERMOS $a ::= x \mid (a\ a) \mid \lambda x.a$

- Operadores Básicos
  - $(a\ b)$ APLICAÇÃO
  - $\lambda x.a$ ABSTRAÇÃO

Exemplos

- $((\lambda x.x + 1)\ 3) = 4$

Exemplos

- $((\lambda x.x + 1)\ 3) = 4$
- $\lambda x.x$ (identidade).

Exemplos

- $((\lambda x.x + 1)\ 3) = 4$
- $\lambda x.x$ (identidade).
- $\lambda x.xx$ (auto-aplicação).

Regras do λ-Calculus

- α-conversão

$$\lambda x.a \rightarrow \lambda y.[y/x]a$$

Regras do λ-Calculus

- α-conversão

$$\lambda x.a \rightarrow \lambda y.[y/x]a$$

- β-contração

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

Regras do λ-Calculus

- α-conversão

$$\lambda x.a \to \lambda y.[y/x]a$$

- β-contração

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

- η-contração

$$\lambda x.(a\ x) \to a, \ \text{se } x \notin FV(a)$$

Exemplos

- (α) $\qquad \lambda x.(\lambda y.(xzy)yx) \to_\alpha \lambda w.(\lambda y.(wzy)yw).$

Exemplos

- (α) $\lambda x.(\lambda y.(xzy)yx) \to_\alpha \lambda w.(\lambda y.(wzy)yw)$.
- (α) $\lambda x.(\lambda y.(xzy)yx) \to_\alpha \lambda z.(\lambda y.(zzy)yz)$ ERRADO

Exemplos

- (α) $\lambda x.(\lambda y.(xzy)yx) \to_\alpha \lambda w.(\lambda y.(wzy)yw).$
- (α) $\lambda x.(\lambda y.(xzy)yx) \to_\alpha \lambda z.(\lambda y.(zzy)yz)$ ERRADO
- (β) $(\lambda x.(x+2))\,5 \to_\beta 5+2$

Exemplos

- (α) $\lambda x.(\lambda y.(xzy)yx) \to_\alpha \lambda w.(\lambda y.(wzy)yw)$.
- (α) $\lambda x.(\lambda y.(xzy)yx) \to_\alpha \lambda z.(\lambda y.(zzy)yz)$ ERRADO
- (β) $(\lambda x.(x+2))\,5 \to_\beta 5+2$
- (β) $(\lambda x.(\lambda y.(yx)))\,y \to_\beta \lambda y.(yy)$ ERRADO
  $(\lambda x.(\lambda y.(yx)))\,y \to_\beta \lambda z.(zy)$ CERTO

Notação de Bruijn

- Desenvolvido pelo matemático holandês Nicolaas Govert de Bruijn[dB72].
- Tem as mesmas propriedades do λ-calculus com nomes.
- Elimina necessidade de α-conversões.

Sintaxe

TERMOS $a ::= \underline{n} \mid (a\ a) \mid \lambda.a$

- Índice $\underline{i}$ com valor menor que a profundidade em que se encontra representa uma variável ligada pelo i-ésimo abstrator.
- Índice $\underline{i}$ com valor maior representa variável livre.

Exemplos

- $\lambda.(\lambda.(\underline{1}\,\underline{4}\,\underline{2}\,)\underline{1}\,)$

Exemplos

- $\lambda.(\lambda.(\underline{1}\ \underline{4}\ \underline{2})\underline{1})$
- $\lambda.\underline{1} \simeq \lambda x.x \simeq \lambda y.y$

Exemplos

- $\lambda.(\lambda.(\underline{1}\,\underline{4}\,\underline{2}\,)\underline{1}\,)$
- $\lambda.\underline{1} \simeq \lambda x.x \simeq \lambda y.y$

Operações de β e η contração precisam de mecanismo que atualize as variáveis livres.

Cálculos de Substituições Explícitas

- É uma extensão do λ-calculus.
- Explicita o processo de substituição.
- Retarda substituições até o momento que se tornam necessárias.

$\lambda\sigma$-Calculus[ACCL91]

SINTAXE

Termos	$a ::= \underline{1} \mid (a\ a) \mid \lambda.a \mid a[s]$
Substituições	$s ::= id \mid \uparrow \mid a.s \mid s \circ s$

$\lambda\sigma$-Calculus[ACCL91]

SINTAXE

Termos	$a ::= \underline{1} \mid (a\ a) \mid \lambda.a \mid a[s]$
Substituições	$s ::= id \mid \uparrow \mid a.s \mid s \circ s$

- Dois tipos de objetos: Termos e Substituições

$\lambda\sigma$-Calculus[ACCL91]

SINTAXE

Termos	$a ::= \underline{1} \mid (a\ a) \mid \lambda.a \mid a[s]$
Substituições	$s ::= id \mid \uparrow \mid a.s \mid s \circ s$

- Dois tipos de objetos: Termos e Substituições
- A β-redução é simulada pelo termo a[s] (*closure*)

$$(\lambda.a)b \to a[b.id] \quad (Beta)$$

$\lambda\sigma + (Eta)$

$(\lambda_A.a)b$	$\longrightarrow$	$a[b.id]$	$(Beta)$
$(a\ b)[s]$	$\longrightarrow$	$(a[s]\ b[s])$	(App)
$1[a.s]$	$\longrightarrow$	a	$(VarCons)$
$a[id]$	$\longrightarrow$	a	(Id)
$(\lambda_A.a)[s]$	$\longrightarrow$	$\lambda_A.(a[1.(s\circ\uparrow)])$	(Abs)
$(a[s])[t]$	$\longrightarrow$	$a[s\circ t]$	$(Clos)$
$id\circ s$	$\longrightarrow$	s	(IdL)
$\uparrow\circ(a.s)$	$\longrightarrow$	s	$(ShiftCons)$
$(s_1\circ s_2)\circ s_3$	$\longrightarrow$	$s_1\circ(s_2\circ s_3)$	$(AssEnv)$
$(a.s)\circ t$	$\longrightarrow$	$a[t].(s\circ t)$	$(MapEnv)$
$s\circ id$	$\longrightarrow$	s	(IdR)
$1.\uparrow$	$\longrightarrow$	id	$(VarShift)$
$1[s].(\uparrow\circ s)$	$\longrightarrow$	s	$(Scons)$
$\lambda_A.(a\ \underline{1})$	$\longrightarrow$	b se $a =_\sigma b[\uparrow]$	(Eta)

λs_e-Calculus[KR97]

SINTAXE

Termos $\quad a ::= \underline{n} \mid (a\ b) \mid \lambda_A.a \mid a\,\sigma^i b \mid \varphi^j_k\, a$

λs_e-Calculus[KR97]

SINTAXE

Termos $\quad a ::= \underline{n} \,|\, (a\ b) \,|\, \lambda_A.a \,|\, a\,\sigma^i b \,|\, \varphi^j_k\, a$

- Apenas Termos como objeto

λs_e-Calculus[KR97]

SINTAXE

Termos $\quad a ::= \underline{n} \,|\, (a\ b) \,|\, \lambda_A.a \,|\, a\,\sigma^i b \,|\, \varphi^j_k\, a$

- Apenas Termos como objeto
- Operador σ para substituições e φ para atualizações.

λs_e-Calculus[KR97]

SINTAXE

Termos $\quad a ::= \underline{n} \,|\, (a\ b) \,|\, \lambda_A.a \,|\, a\,\sigma^i b \,|\, \varphi^j_k\, a$

- Apenas Termos como objeto
- Operador σ para substituições e φ para atualizações.
- A β-redução é simulada pelo σ

$$(\lambda.a\ b) \to a\,\sigma^1 b \quad (\sigma\text{-generation})$$

$\lambda s_e + (Eta)$

$$
\begin{array}{llll}
(\lambda_A.a\ \ b) & \longrightarrow & a\,\sigma^1 b & (\sigma\text{-generation}) \\
(\lambda_A.a)\,\sigma^i b & \longrightarrow & \lambda_A.(a\,\sigma^{i+1} b) & (\sigma\text{-}\lambda\text{-transition}) \\
(a_1\ \ a_2)\,\sigma^i b & \longrightarrow & ((a_1\,\sigma^i b)\ \ (a_2\,\sigma^i b)) & (\sigma\text{-app-transition}) \\
\underline{n}\,\sigma^i b & \longrightarrow & \begin{cases} \underline{n-1} & \text{se } n > i \\ \varphi^i_0\, b & \text{se } n = i \\ \underline{n} & \text{se } n < i \end{cases} & (\sigma\text{-destruction}) \\
\varphi^i_k\,(\lambda_A.a) & \longrightarrow & \lambda_A.(\varphi^i_{k+1}\, a) & (\varphi\text{-}\lambda\text{-transition}) \\
\varphi^i_k\,(a_1\ \ a_2) & \longrightarrow & ((\varphi^i_k\, a_1)\ \ (\varphi^i_k\, a_2)) & (\varphi\text{-app-transition}) \\
\varphi^i_k\,\underline{n} & \longrightarrow & \begin{cases} \underline{n+i-1} & \text{se } n > k \\ \underline{n} & \text{se } n \leq k \end{cases} & (\varphi\text{-destruction}) \\
(a_1\,\sigma^i a_2)\,\sigma^j b & \longrightarrow & (a_1\,\sigma^{j+1} b)\,\sigma^i (a_2\,\sigma^{j-i+1} b) \quad \text{se } i \leq j & (\sigma\text{-}\sigma\text{-transition}) \\
(\varphi^i_k\, a)\,\sigma^j b & \longrightarrow & \varphi^{i-1}_k\, a \quad \text{se } k < j < k+i & (\sigma\text{-}\varphi\text{-transition 1}) \\
(\varphi^i_k\, a)\,\sigma^j b & \longrightarrow & \varphi^i_k\,(a\,\sigma^{j-i+1} b) \quad \text{se } k+i \leq j & (\sigma\text{-}\varphi\text{-transition 2}) \\
\varphi^i_k\,(a\,\sigma^j b) & \longrightarrow & (\varphi^i_{k+1}\, a)\,\sigma^j (\varphi^i_{k+1-j}\, b) \quad \text{se } j \leq k+1 & (\varphi\text{-}\sigma\text{-transition}) \\
\varphi^i_k\,(\varphi^j_l\, a) & \longrightarrow & \varphi^j_l\,(\varphi^i_{k+1-j}\, a) \quad \text{se } l+j \leq k & (\varphi\text{-}\varphi\text{-transition 1}) \\
\varphi^i_k\,(\varphi^j_l\, a) & \longrightarrow & \varphi^{j+i-1}_l\, a \quad \text{se } l \leq k < l+j & (\varphi\text{-}\varphi\text{-transition 2}) \\
\lambda_A.(a\ \underline{1}) & \longrightarrow & b \quad \text{se } a =_{s_e} \varphi^2_0\, b & (\text{Eta})
\end{array}
$$

Tipos Simples

SINTAXE

TIPOS	$A ::= K \mid A \rightarrow B$
TERMOS	$a ::= x \mid (a\ a) \mid \lambda x{:}B.a$

Tipos Simples

SINTAXE

TIPOS $A ::= K \mid A \rightarrow B$

TERMOS $a ::= x \mid (a\ a) \mid \lambda x{:}B.a$

- Um λ-termo a tem tipo B (a:B)

Tipos Simples

SINTAXE

TIPOS	$A ::= K \mid A \rightarrow B$
TERMOS	$a ::= x \mid (a\ a) \mid \lambda x{:}B.a$

- Um λ-termo a tem tipo B (a:B)
- Contexto $\Gamma = \{x_1{:}A_1, x_2{:}A_2, \ldots, x_n{:}A_n\}$

Tipos Simples

SINTAXE

TIPOS $A ::= K \mid A \rightarrow B$

TERMOS $a ::= x \mid (a\ a) \mid \lambda x{:}B.a$

- Um λ-termo a tem tipo B (a:B)
- Contexto $\Gamma = \{x_1{:}A_1, x_2{:}A_2, \ldots, x_n{:}A_n\}$
- Um λ-termo a tem tipo B em um contexto Γ

$$\Gamma \vdash a : B$$

Exemplo

- $\vdash \lambda x\colon \mathbf{int}.(x+1) : \mathbf{int} \to \mathbf{int}$

Exemplo

- $\vdash \lambda x\colon \mathbf{int}.(x+1) : \mathbf{int} \to \mathbf{int}$
- $x{:}\mathbf{int} \vdash \lambda y\colon \mathbf{int}.(x+y+1) : \mathbf{int} \to \mathbf{int}$

O Sistema TA_λ

$$\frac{x \notin \Gamma}{x{:}A, \Gamma \vdash x : A}(Start) \qquad \frac{x \notin \Gamma \quad \Gamma \vdash a : B}{x{:}A, \Gamma \vdash a : B}(Weak)$$

$$\frac{x{:}A, \Gamma \vdash a : B}{\Gamma \vdash \lambda_{x:A}.a : A \to B}(Abs) \qquad \frac{\Gamma \vdash a : B \to A \quad \Gamma \vdash b : B}{\Gamma \vdash (a\ b) : A}(App)$$

Exemplos

$$\frac{x{:}A \vdash x : A\,(Start)}{\vdash \lambda_{x:A}.x : A \to A}(Abs) \qquad \frac{x{:}A \to A \vdash x : A \to A\,(Start)}{\vdash \lambda_{x:A\to A}.x : (A \to A) \to (A \to A)}(Abs)$$

$$\frac{\vdash \lambda_{x:A\to A}.x : (A \to A) \to (A \to A) \quad \lambda_{x:A}.x : A \to A}{\Gamma \vdash (\lambda_{x:A\to A}.x\ \lambda_{x:A}.x) : A \to A}(App)$$

O Sistema TA_{dB}

$$A.\Gamma \vdash \underline{1} : A \ (var) \qquad \frac{\Gamma \vdash \underline{n} : B}{A.\Gamma \vdash \underline{n+1} : B} \ (varn)$$

$$\frac{A.\Gamma \vdash b : B}{\Gamma \vdash \lambda_A.b : A \to B} \ (lambda) \qquad \frac{\Gamma \vdash a : A \to B \quad \Gamma \vdash b : A}{\Gamma \vdash (a\, b) : B} \ (app)$$

Exemplos

- $$\frac{A \vdash \underline{1} \ : A \ (var)}{\vdash \lambda_A.\underline{1} \ : A \rightarrow A}(lambda)$$

Exemplos

- $\dfrac{A \vdash \underline{1}\ : A\ (var)}{\vdash \lambda_A.\underline{1}\ : A \rightarrow A}(lambda)$

- $\lambda.(\underline{1}\ \underline{1}\ )$ não é tipável

O Sistema $TA_{\lambda\sigma}$

TERMOS

$$A.\Gamma \vdash \underline{1} : A \ (var) \qquad \frac{A.\Gamma \vdash b : B}{\Gamma \vdash \lambda_A.b : A \rightarrow B} \ (lambda)$$

$$\frac{\Gamma \vdash a : A \rightarrow B \quad \Gamma \vdash b : A}{\Gamma \vdash (a\ b) : B} \ (app) \qquad \frac{\Gamma \vdash s \triangleright \Gamma' \quad \Gamma' \vdash a : A}{\Gamma \vdash a[s] : A} \ (clos)$$

SUBSTITUIÇÕES

$$\Gamma \vdash id \triangleright \Gamma \ (id) \qquad A.\Gamma \vdash \uparrow \triangleright \Gamma \ (shift)$$

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash s \triangleright \Gamma'}{\Gamma \vdash a.s \triangleright A.\Gamma'} \ (cons) \qquad \frac{\Gamma \vdash s'' \triangleright \Gamma'' \quad \Gamma'' \vdash s' \triangleright \Gamma'}{\Gamma \vdash s' \circ s'' \triangleright \Gamma'} \ (comp)$$

O Sistema TA_{s_e}

$$A.\Gamma \vdash \underline{1} : A \; (Var) \qquad \frac{\Gamma \vdash \underline{n} : B}{A.\Gamma \vdash \underline{n+1} : B} \; (Varn)$$

$$\frac{A.\Gamma \vdash b : B}{\Gamma \vdash \lambda_A.b : A \to B} \; (Lambda) \qquad \frac{\Gamma \vdash a : A \to B \quad \Gamma \vdash b : A}{\Gamma \vdash (a\, b) : B} \; (App)$$

$$\frac{\Gamma_{\leq k}.\Gamma_{\geq k+i} \vdash a : A}{\Gamma \vdash \varphi^i_k\, a : A} \; (Phi) \qquad \frac{\Gamma_{\geq i} \vdash b : B \quad \Gamma_{<i}.B.\Gamma_{\geq i} \vdash a : A}{\Gamma \vdash a\, \sigma^i\, b : A} \; (Sigma)$$

Propriedades importantes de sistemas de tipos

- Redução do Sujeito
  Se $\Gamma \vdash a : A$ e $a \rightarrow_{\beta\eta} b$, então $\Gamma \vdash b : A$.

Propriedades importantes de sistemas de tipos

- Redução do Sujeito
  Se $\Gamma \vdash a : A$ e $a \longrightarrow_{\beta\eta} b$, então $\Gamma \vdash b : A$.
- Verificação de tipos (Decidibilidade)

Propriedades importantes de sistemas de tipos

- Redução do Sujeito
  Se $\Gamma \vdash a : A$ e $a \rightarrow_{\beta\eta} b$, então $\Gamma \vdash b : A$.
- Verificação de tipos (Decidibilidade)
- Inferência de Tipos (Decidibilidade)

Propriedades importantes de sistemas de tipos

- Redução do Sujeito
  Se $\Gamma \vdash a : A$ e $a \longrightarrow_{\beta\eta} b$, então $\Gamma \vdash b : A$.
- Verificação de tipos (Decidibilidade)
- Inferência de Tipos (Decidibilidade)
- Existência de Habitantes em Tipos (Decidibilidade)

Redução do Sujeito para $\lambda\sigma$ e λs_e

- Propriedade vale para $\lambda\sigma$[ACCL91] e para λs[KR96]

Redução do Sujeito para $\lambda\sigma$ e λs_e

- Propriedade vale para $\lambda\sigma$[ACCL91] e para λs[KR96]
- Detectou-se que não existia demonstração da propriedade para λs_e

Redução do Sujeito para $\lambda\sigma$ e λs_e

- Propriedade vale para $\lambda\sigma$[ACCL91] e para λs[KR96]
- Detectou-se que não existia demonstração da propriedade para λs_e
- Detectou-se que a propriedade nao vale para $\lambda\sigma$ nem para λs_e estendidas com η-contração [DHK00], [ARK01]

Redução do Sujeito para λs_e

Teorema

(Redução de Sujeito para λs_e) *Seja a um λs_e-termo. Se $\Gamma \vdash_{T\lambda s_e} a : A$ e $a \rightarrow_{\lambda s_e} a'$, então $\Gamma \vdash_{T\lambda s_e} a' : A$.*

Contra-Exemplo $\lambda\sigma + (Eta)$

$$\lambda_A.(a\ \underline{1}\,) \to b \quad \text{se} \quad a =_\sigma b[\uparrow] \quad (Eta)$$

- Tem-se que $\underline{2} =_\sigma \big(\underline{2}\,[\lambda(\underline{1}\ \underline{1}\,).\underline{1}\,.\uparrow]\big)[\uparrow]$.

  Portanto

$$\lambda(\underline{2}\ \underline{1}\,) \longrightarrow_\eta \underline{2}\,[\lambda(\underline{1}\ \underline{1}\,).\underline{1}\,.\uparrow]$$

Solução

- Formular um novo cálculo que implemente a η-redução de maneira explícita

Solução

- Formular um novo cálculo que implemente a η-redução de maneira explícita
- Para $\lambda(a\ \underline{1}\,) \to b$, constrói-se b a partir de a

O sistema R_η

$$
\begin{array}{llll}
(a\ b)[s] & \longrightarrow & (a[s]\ b[s]) \quad \text{se } \diamond \text{ ocorre em } s & (\eta\text{-App}) \\
\underline{1}\,[a.s] & \longrightarrow & a \quad \text{se } \diamond \text{ ocorre em } s & (\eta\text{-VarCons}) \\
(\lambda_A.a)[s] & \longrightarrow & \lambda_A.(a[1.(s \circ \uparrow)]) \quad \text{se } \diamond \text{ ocorre em } s & (\eta\text{-Abs}) \\
(a[s])[t] & \longrightarrow & \begin{cases} a[s \circ t] & \text{se } \diamond \text{ ocorre em t e } a = \underline{1} \\ a[N_{R_\eta}(s \circ t)] & \text{se } \diamond \text{ ocorre em t e} \\ & N_{R_\eta}(s \circ t) \in \Lambda_\sigma \\ \mathit{erro} & \text{se } \diamond \text{ ocorre em t e} \\ & N_{R_\eta}(s \circ t) \notin \Lambda_\sigma \end{cases} & (\eta\text{-Clos}) \\
(s_1 \circ s_2) \circ t & \longrightarrow & s_1 \circ (s_2 \circ t) \quad \text{se } \diamond \text{ ocorre em } t & (\eta\text{-AssEnv}) \\
(a.s) \circ t & \longrightarrow & a[t].(s \circ t) \quad \text{se } \diamond \text{ ocorre em } s \text{ ou } t & (\eta\text{-MapEnv}) \\
\uparrow \circ\, (a.s) & \longrightarrow & s \quad \text{se } \diamond \text{ ocorre em } s & (\eta\text{-ShiftConsTail}) \\
\uparrow \circ (\diamond.s) & \longrightarrow & s & (\eta\text{-ShiftConsHead}) \\
\diamond[\uparrow] & \longrightarrow & \diamond & (\eta\text{-Cons}) \\
id \circ s & \longrightarrow & \mathit{erro} \quad \text{se } \diamond \text{ ocorre em } s & (\eta\text{-IdL}) \\
1[s].\uparrow \circ\, s & \longrightarrow & \mathit{erro} \quad \text{se } \diamond \text{ ocorre em } s & (\eta\text{-Scons}) \\
\underline{1}\,[\diamond.s] & \longrightarrow & \mathit{erro} & (\text{Erro})
\end{array}
$$

(Eta) para $\lambda\sigma$

$$\lambda_A.(a\ \underline{1}\,) \rightarrow N_{R_\eta}(a[\diamond.id]) \quad \text{se } N_{R_\eta}(a[\diamond.id]) \text{ é um } \lambda\sigma\text{-termo}$$

(Eta) para $\lambda\sigma$

$$\lambda_A.(a\ \underline{1}\ ) \longrightarrow N_{R_\eta}(a[\diamond.id]) \quad \text{se } N_{R_\eta}(a[\diamond.id]) \text{ é um } \lambda\sigma\text{-termo}$$

Teorema

(Redução de Sujeito para (Eta)) *Seja* $\lambda_B.(a\ \underline{1}\ )$ *um* $\lambda\sigma$*-termo, tal que* $\Gamma \vdash \lambda_B.(a\ \underline{1}\ ) : A$*. Se* $\lambda_B.(a\ \underline{1}\ ) \longrightarrow_{Eta} b$*, então* $\Gamma \vdash b : A$

O sistema $R_{\eta s_e}$

$$
\begin{array}{llll}
(a\ b)\,\eta^i & \longrightarrow & (a\,\eta^i\ b\,\eta^i) & (\eta\text{-app-transition}) \\
(\lambda_A.a)\,\eta^i & \longrightarrow & \lambda_A.a\,\eta^{i+1} & (\eta\text{-}\lambda\text{-transition}) \\
\underline{n}\,\eta^i & \longrightarrow & \begin{cases} \underline{n} & \text{se } n < i \\ erro & \text{se } n = i \\ \underline{n-1} & \text{se } n > i \end{cases} & (\eta\text{-destruction2}) \\
(a\,\sigma^j b)\,\eta^i & \longrightarrow & (a\,\eta^i)\,\sigma^{j-1} b \quad \text{se } i < j & (\eta\text{-}\sigma\text{-transition 1}) \\
(a\,\sigma^j b)\,\eta^i & \longrightarrow & (a\,\eta^{i+1})\,\sigma^j (b\,\eta^{i-j+1}) \quad \text{se } i \geq j & (\eta\text{-}\sigma\text{-transition 2}) \\
(\varphi^j_k\ a)\,\eta^i & \longrightarrow & \varphi^j_{k-1}\,(a\,\eta^i) \quad \text{se } i \leq k & (\eta\text{-}\varphi\text{-transition 1}) \\
(\varphi^j_k\ a)\,\eta^i & \longrightarrow & \varphi^{j-1}_k\ a \quad \text{se } k < i < k+j & (\eta\text{-}\varphi\text{-transition 2}) \\
(\varphi^j_k\ a)\,\eta^i & \longrightarrow & \varphi^j_k\,(a\,\eta^{i-j+1}) \quad \text{se } i > k \text{ e } i \geq k+j & (\eta\text{-}\varphi\text{-transition 3})
\end{array}
$$

(Eta) para λs_e

$$\lambda_A.(a\ \underline{1}\,) \rightarrow N_{R_{\eta s_e}}(a\,\eta^1) \quad \text{se } N_{R_{\eta s_e}}(a\,\eta^1) \text{ é um } \lambda s_e\text{-termo.}$$

(Eta) para λs_e

$$\lambda_A.(a\ \underline{1}\ ) \longrightarrow N_{R_{\eta s_e}}(a\,\eta^1) \quad \text{se } N_{R_{\eta s_e}}(a\,\eta^1) \text{ é um } \lambda s_e\text{-termo.}$$

Teorema

(Redução de Sujeito para (Eta)): *Seja* $\lambda_B.(a\ \underline{1}\ )$ *um* λs_e*-termo, tal que* $\Gamma \vdash \lambda_B.(a\ \underline{1}\ ) : A$*. Se* $\lambda_B.(a\ \underline{1}\ ) \longrightarrow_{Eta} b$*, então* $\Gamma \vdash b : A$

Conclusões

- Tipos essenciais em computação

Conclusões

- Tipos essenciais em computação
- Subsitituição deve ser tratada explicitamente para fechar a brecha entre teoria/implementação

Conclusões

- Tipos essenciais em computação
- Subsitituição deve ser tratada explicitamente para fechar a brecha entre teoria/implementação
- Os sistemas de tipos para cálculos de substituições explícitas devem satisfazer as propriedades relevantes dos sistemas de tipos: redução do sujeito, verificação e inferência de tipos.

Conclusões e Trabalhos Futuros

- Detectou-se que não existia demonstração da propriedade de redução de sujeito para λs_e

Conclusões e Trabalhos Futuros

- Detectou-se que não existia demonstração da propriedade de redução de sujeito para λs_e
- Detectaram-se falhas nos sistemas de tipos do $\lambda\sigma$ e do λs_e que violavam a propriedade de redução do sujeito

Conclusões e Trabalhos Futuros

- Detectou-se que não existia demonstração da propriedade de redução de sujeito para λs_e
- Detectaram-se falhas nos sistemas de tipos do $\lambda\sigma$ e do λs_e que violavam a propriedade de redução do sujeito
- Demonstrou-se redução de sujeito para λs_e

Conclusões e Trabalhos Futuros

- Detectou-se que não existia demonstração da propriedade de redução de sujeito para λs_e
- Detectaram-se falhas nos sistemas de tipos do $\lambda\sigma$ e do λs_e que violavam a propriedade de redução do sujeito
- Demonstrou-se redução de sujeito para λs_e
- Formularam-se novas η-regras que solucionam o problema

Conclusões e Trabalhos Futuros

- Detectou-se que não existia demonstração da propriedade de redução de sujeito para λs_e
- Detectaram-se falhas nos sistemas de tipos do $\lambda\sigma$ e do λs_e que violavam a propriedade de redução do sujeito
- Demonstrou-se redução de sujeito para λs_e
- Formularam-se novas η-regras que solucionam o problema

- Investigar existência de tipos principais nos cálculos de substituição explícita

M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy.
Explicit Substitutions.
J. of Func. Programming, 1(4):375–416, 1991.

M. Ayala-Rincón and F. Kamareddine.
Unification via the λs_e-Style of Explicit Substitution.
The Logical Journal of the Interest Group in Pure and Applied Logics, 9(4):489–523, 2001.

A. Church.
A set of postulates for the foundation of logic,
Annals of Math **33** (1932), no. 2, 346–366.

N.G. de Bruijn.
Lambda-Calculus Notation with Nameless Dummies, a Tool for Automatic Formula Manipulation, with Application to the Church-Rosser Theorem.
Indag. Mat., 34(5):381–392, 1972.

G. Dowek, T. Hardin, and C. Kirchner.
Higher-order unification via explicit substitutions.
Information and Computation, 157:183–235, 2000.

S. Kleene and B. Rosser.
The inconsistency of certain formal logics,
Annals of Math vol. 36(2):630-636, 1935.

F. Kamareddine and A. Ríos.
A λ-calculus à la de Bruijn with explicit substitutions,
volume 982 of Lectures Notes Computer Science, pages 45–62. Springer.

F. Kamareddine and A. Rï¿ $\frac{1}{2}$s.

Generalized β-reduction and explicit substitutions,
volume 1140 of Lectures Notes Computer Science, pages 378–392. Springer.

F. Kamareddine and A. Ríos.
Extending a λ-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms,
Journal of Functional Programming 7, pages 395–420, 1997. Springer.