Lambda Calculus com Tipos

Maria João Frade

HASLab - INESC TEC
Departamento de Informática, Universidade do Minho

2019/2020

Lambda calculus com tipos simples - $\lambda_{\rightarrow}$

- O propósito dos sistemas de tipos é a classificação dos termos.
- A relação entre objectos e tipos é capturada por juízos da forma $e : \tau$.
- Os sistemas de tipos foram introduzidos por Bertrand Russell na década de 1900.
- Em 1940 A. Church introduziu o *lambda calculus com tipos simples*.
- Uma versão diferente do lambda calculus tipificado tinha já sido apresentada por H. Curry em 1934, para a lógica combinatorial.
- Na versão de Church os termos têm anotações de tipo, enquanto que na versão de Curry os termos têm a mesma sintaxe abstracta que no lambda calculus puro.

Sintaxe

Tipos

- Seja $\mathcal{G}$ um conjunto não vazio de *tipos de base*.
- Os tipos são definidos pela seguinte sintaxe abstracta

$$\tau, \sigma \ ::= \ T \mid \tau \rightarrow \sigma \qquad \text{onde} \quad T \in \mathcal{G}$$

Termos

- Assume-se um conjunto enumerável de *variáveis*: $x, y, z, \ldots$
- Fixamos um conjunto de termos *constantes* dos diferentes tipos
- Os termos são definidos pela seguinte sintaxe abstracta

$$e, a, b \ ::= \ c \mid x \mid \lambda x\!:\!\tau.\, e \mid a\, b \qquad \text{onde } c \text{ é uma constante}$$

Variáveis livres e ligadas

FV(e) denota o conjunto das *variáveis livres* de uma expressão e

$$\begin{array}{rcl} \mathsf{FV}(c) & = & \{\} \\ \mathsf{FV}(x) & = & \{x\} \\ \mathsf{FV}(\lambda x\!:\!\tau.\, a) & = & \mathsf{FV}(a) \backslash \{x\} \\ \mathsf{FV}(a\, b) & = & \mathsf{FV}(a) \cup \mathsf{FV}(b) \end{array}$$

BV(e) denota o conjunto das *variáveis ligadas* de uma expressão e

$$\begin{array}{rcl} \mathsf{BV}(c) & = & \{\} \\ \mathsf{BV}(x) & = & \{\} \\ \mathsf{BV}(\lambda x\!:\!\tau.\, a) & = & \mathsf{BV}(a) \cup \{x\} \\ \mathsf{BV}(a\, b) & = & \mathsf{BV}(a) \cup \mathsf{BV}(b) \end{array}$$

Uma variável pode ser simultaneamente livre e ligada numa dada expressão.
Por exemplo,

$$(x\, y)\, \lambda z\!:\!\tau.\, \lambda x\!:\!\tau \rightarrow \sigma.\, x\, z$$

Convenções

Para evitar parentesis segue-se a seginte convenção:

- a construção de tipos $\rightarrow$ é associativa à direita;
- a aplicação é associativa à esquerda;
- o âmbito da abstração λ estende-se para a direita o mais possível.

α-conversão

$$\lambda x\!:\!\tau.\, e = \lambda y\!:\!\tau.\, e[y/x] \quad , \text{se } \; y \notin \mathsf{FV}(e)$$

Esta conversão induz uma relação de equivalência nos termos.

Convenção das variáveis

- Identificamos os termos que são iguais a menos de renomeação de variáveis ligadas (α-conversão). Exemplo: $(\lambda x\!:\!\tau.\, y\, x) = (\lambda z\!:\!\tau.\, y\, z)$.
- Todas as variáveis ligadas são escolhidas de forma a serem diferentes das variáveis livres.

Substituição

A convensão das variáveis permite definir a substituição do seguinte modo:

Subtituição

$$\begin{array}{ll}
c[a/x] = c & \\
x[a/x] = a & \\
y[a/x] = y & \text{se } x \neq y \\
(\lambda y\!:\!\tau.\, b)[a/x] = (\lambda y\!:\!\tau.\, b[a/x]) & \\
(e_1\, e_2)[a/x] = (e_1[a/x])\, (e_2[a/x]) &
\end{array}$$

Lema da substituição

Sejam x e y variáveis distintas e $x \notin \mathsf{FV}(e)$, então

$$(a[b/x])[e/y] = (a[e/y])[(b[e/y])/x]$$

β-redução

A redução β indica o efeito de aplicar uma função a um argumento.

β-redução

A β-redução, $\rightarrow_\beta$, é definida como o fecho compatível da regra

$$(\lambda x\!:\!\tau.\, a)\; b \;\rightarrow_\beta\; a[b/x]$$

- $\rightarrow_\beta^*$ é fecho reflexivo e transitivo de $\rightarrow_\beta$.
- $=_\beta$ é fecho reflexivo, simétrico e transitivo de $\rightarrow_\beta$.
- um termo $(\lambda x\!:\!\tau.\, a)\, b$ chama-se *β-redex* e a $a[b/x]$ o seu *contractum*

Uma expressão que não contém nenhum β-redex diz-se uma *forma normal*.

Uma expressão e diz-se *fortemente normalizável* se todas as sequências de redução com origem em e terminam.

Sistema de tipos

No lambda calculus com tipos as funções:

- são classificadas com tipos simples que determinam o tipo dos seus argumentos e o tipo dos valores que elas produzem;
- só podem ser aplicadas a argumentos de tipo apropriado.

Para definir a relação de tipificação entre termos e tipos precisamos de introduzir o conceito de *contexto*, para declarar os tipos das variáveis livres.

Sistema de tipos

Contexto

- Um *contexto* Γ é um conjunto finito de assumpções $\{x_1 : \tau_1, \ldots, x_n : \tau_n\}$ onde as variáveis x_i são todas distintas.
- Γ pode ser visto como uma função parcial: $\mathsf{dom}(\Gamma) = \{x_1, \ldots, x_n\}$ e $\Gamma(x_i) = \tau_1$.
- Usualmente escrevemos $x_1 : \tau_1, \ldots, x_n : \tau_n$ em vez de $\{x_1 : \tau_1, \ldots, x_n : \tau_n\}$, e quando escrevemos $\Gamma, x : \tau$ ou Γ, Γ' assumimos implicitamente que $x \notin \mathsf{dom}(\Gamma)$ e $\mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Gamma') = \emptyset$.

Um juízo de tipificação é um triplo da forma

$$\Gamma \vdash e : \sigma$$

onde Γ é um contexto, e um termo e σ um tipo.

Sistema de tipos

Regras de inferência de tipos

(var) $\quad \dfrac{}{\Gamma \vdash x : \sigma} \quad$ se $(x : \sigma) \in \Gamma$

(const) $\quad \dfrac{}{\Gamma \vdash c : T} \quad$ se c tem tipo T

(abs) $\quad \dfrac{\Gamma, x : \tau \vdash e : \sigma}{\Gamma \vdash (\lambda x{:}\tau.\, e) : \tau \to \sigma}$

(app) $\quad \dfrac{\Gamma \vdash a : \tau \to \sigma \qquad \Gamma \vdash b : \tau}{\Gamma \vdash (a\ b) : \sigma}$

Um termo e diz-se *bem tipificado* se $\Gamma \vdash e : \sigma$ para algum Γ e σ.

Sistema de tipos

Árvore de tipificação de $z : \tau \vdash (\lambda y{:}\tau \to \tau.\, yz)(\lambda x{:}\tau.\, x) : \tau$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{z : \tau, y : \tau \to \tau \vdash y : \tau \to \tau}\,(\mathsf{var}) \quad \dfrac{}{z : \tau, x : \tau \to \tau \vdash z : \tau}\,(\mathsf{var})}{z : \tau, y : \tau \to \tau \vdash yz : \tau}\,(\mathsf{app})}{z : \tau \vdash (\lambda y{:}\tau \to \tau.\, yz) : (\tau \to \tau) \to \tau}\,(\mathsf{abs}) \quad \dfrac{\dfrac{}{z : \tau, x : \tau \vdash x : \tau}\,(\mathsf{var})}{z : \tau \vdash (\lambda x{:}\tau.\, x) : \tau \to \tau}\,(\mathsf{abs})}{z : \tau \vdash (\lambda y{:}\tau \to \tau.\, yz)(\lambda x{:}\tau.\, x) : \tau}\,(\mathsf{app})$$

A mesma árvore apresentada de forma tabular:

1.	$z : \tau \vdash (\lambda y{:}\tau \to \tau.\, y\,z)(\lambda x{:}\tau.\, x) : \tau$	(app)
1.1.	$z : \tau \vdash (\lambda y{:}\tau \to \tau.\, yz) : (\tau \to \tau) \to \tau$	(abs)
1.1.1.	$z : \tau, y : \tau \to \tau \vdash yz : \tau$	(app)
1.1.1.1.	$z : \tau, y : \tau \to \tau \vdash y : \tau \to \tau$	(var)
1.1.1.2.	$z : \tau, x : \tau \to \tau \vdash z : \tau$	(var)
1.2.	$z : \tau \vdash (\lambda x{:}\tau.\, x) : \tau \to \tau$	(abs)
1.2.1.	$z : \tau, x : \tau \vdash x : \tau$	(var)

Propriedades

Unicidade de tipos

Se $\Gamma \vdash a : \sigma$ e $\Gamma \vdash a : \tau$, então $\sigma = \tau$.

Inferência de tipos

O problema da inferência de tipos é decidível. Ou seja, é possível deduzir automaticamente o tipo de um termo num dado contexto, caso exista.

Preservação de tipos

Se $\Gamma \vdash a : \sigma$ e $a \to_\beta^* b$, então $\Gamma \vdash b : \sigma$.

Normalização forte

Se $\Gamma \vdash e : \sigma$, então todas as sequências de β-reduções com origem e terminam.

Propriedades

Confluência

Se $a =_\beta b$, então $a \to_\beta^* e$ e $b \to_\beta^* e$, para algum termo e .

Propriedade da substituição

Se $\Gamma, x : \tau \vdash a : \sigma$ e $\Gamma \vdash b : \tau$, então $\Gamma \vdash a[b/x] : \sigma$.

Enfraquecimento

Se $\Gamma \vdash e : \sigma$ e $\Gamma \subseteq \Gamma'$, então $\Gamma' \vdash e : \sigma$.

Fortalecimento

Se $\Gamma, x : \tau \vdash e : \sigma$ e $x \notin \mathsf{FV}(e)$, então $\Gamma \vdash e : \sigma$.

Exercícios

Apresente (se possível) juízos de tipificação para os seguintes termos (omitimos aqui as anotações de tipos nos termos para a simplificar a apresentação).

- $\lambda f.\lambda y.\, f\, y\, y$
- $\lambda g.\lambda x.\lambda y.\lambda z.\, g\,(x\,z)\,(y\,z)$
- $(\lambda x.\, x\, x)(\lambda x.\, x\, x)$
- $(\lambda f.\lambda y.\, f\, y\, y)\,(\lambda f.\lambda y.\, f\, y\, y)$

Indique quais dos seguintes termos são tipificáveis.

$$
\begin{aligned}
t_1 &\equiv (\lambda f : \mathsf{Int} \to \mathsf{Int}.\, \lambda x : \mathsf{Int}.\, f\,(f\,x))\,(\lambda y : \mathsf{Int}.\, h\, y\, 2)\\
t_2 &\equiv (\lambda y : \mathsf{Int} \to \mathsf{Bool}.\, \lambda x : \mathsf{Bool} \to (\mathsf{Int} \to \mathsf{Bool}) \to \mathsf{Int}.\, x\,(y\,a)\,y)\,(\lambda z : \mathsf{Int}.\, f\, z)\\
t_3 &\equiv \lambda z : \mathsf{Int} \to \mathsf{Int} \to \mathsf{Bool}.\, h\,(z\,5\,(h\,(z\ 1)))
\end{aligned}
$$

Apresente uma justificação para a sua resposta.

Exercícios

Escreva as anotações de tipo para os termos

$$
\begin{aligned}
K &\equiv \lambda x.\lambda y.\, x\\
S &\equiv \lambda x.\lambda y.\lambda z.\, x\, z\,(y\, z)
\end{aligned}
$$

de forma a que eles sejam termos bem tipificados, e indique os seu tipos. Apresente as árvores de derivação no sistema de tipos que justificam as suas respostas.