Second order typed lambda calculus $\lambda 2$

first order abstraction

first order application

In $\lambda\to$: terms depending on terms

abstraction from terms: $\lambda x:\sigma.M$ ... "the term $\lambda x:\sigma.M$ depends on the term x"

application of terms: MN ... "the term M is applied to the term N"

In $\lambda 2$: terms depending on types

Examples: •) "The" identity function

$\lambda x:\alpha.x$ ... identity function on α, $\lambda x:nat.x$ ... identity function on nat

$\lambda x:(nat\to bool).x$ ... identity function on $nat\to bool$

$\lambda\alpha:*.\lambda x:\alpha.x$ ... "the" identity function

type of all types

second order abstraction

β-reduction: $(\lambda\alpha:*.\lambda x:\alpha.x)\,nat \to_\beta \lambda x:nat.x$

need: second order abstraction and application and β-red. for second order terms

•) Iteration

$\lambda x:\sigma.F(Fx)$ ... iteration of F

Let $D :\equiv \lambda\alpha:*.\lambda f:\alpha\to\alpha.\lambda x:\alpha.f(fx)$

$suc: nat\to nat \Rightarrow D\,nat\,suc \to_\beta \lambda x:nat.suc(suc\,x)$

•) Composition

Let $\circ :\equiv \lambda\alpha:*.\lambda\beta:*.\lambda\gamma:*.\lambda f:\alpha\to\beta.\lambda g:\beta\to\gamma.\lambda x:\alpha.g(fx)$

$F:A\to B,\ G:B\to C \Rightarrow \circ ABCFG \to_\beta \lambda x:A.G(Fx)$

Product types (Π-types)

What is the type of $\lambda\alpha:*.\lambda x:\alpha.x$?

Maybe $\lambda\alpha:*.\lambda x:\alpha.x : *\to(\alpha\to\alpha)$?

Problem: We want to identify $\lambda\alpha:*.\lambda x:\alpha.x$ with $\lambda\beta:*.\lambda x:\beta.x$ (α-conversion),

but then:

$$\begin{array}{ccc} \lambda\alpha:*.\lambda x:\alpha.x & : & *\to(\alpha\to\alpha) \\ \equiv & & \not\equiv \\ \lambda\beta:*.\lambda x:\beta.x & : & *\to(\beta\to\beta) \end{array}$$

Solution: Introduce product types:

$$\begin{array}{ccc} \lambda\alpha:*.\lambda x:\alpha.x & : & \Pi\alpha:*.\alpha\to\alpha \\ \equiv & & \equiv \leftarrow \alpha\text{-conversion} \\ \lambda\beta:*.\lambda x:\beta.x & : & \Pi\beta:*.\beta\to\beta \end{array}$$

new binder in addition to λ, binds type variables

Examples: $\lambda\alpha:*.\lambda f:\alpha\to\alpha.\lambda x:\alpha.f(fx) : \Pi\alpha:*.(\alpha\to\alpha)\to\alpha\to\alpha$

$\lambda\alpha:*.\lambda\beta:*.\lambda\gamma:*.\lambda f:\alpha\to\beta.\lambda g:\beta\to\gamma.\lambda x:\alpha.g(fx) :$

$\Pi\alpha:*.\Pi\beta:*.\Pi\gamma:*.(\alpha\to\beta)\to(\beta\to\gamma)\to\alpha\to\gamma$

The system $\lambda 2$

Abstract syntax for $\lambda 2$-types: $\mathbb{T}_2 = \mathbb{V} \mid (\mathbb{T}_2 \to \mathbb{T}_2) \mid (\Pi \mathbb{V} : * . \mathbb{T}_2)$

$\mathbb{V} = \{\alpha, \beta, \gamma, \dots\}$ ... set of type variables; arrow types; product types

Abstract syntax for $\lambda 2$-terms: $\Lambda_{\mathbb{T}_2} = V \mid (\Lambda_{\mathbb{T}_2} \Lambda_{\mathbb{T}_2}) \mid (\Lambda_{\mathbb{T}_2} \mathbb{T}_2) \mid (\lambda V : \mathbb{T}_2 . \Lambda_{\mathbb{T}_2}) \mid (\lambda \mathbb{V} : * . \Lambda_{\mathbb{T}_2})$

$V = \{a, b, c, \dots\}$ ... set of variables; first order application; second order application; first order abstraction; second order abstraction

Conventions:
- ·) Outermost parentheses may be omitted
- ·) Application is left-associative, λ- and Π-abstraction is right-associative
- ·) Application and $\to$ take precedence over λ- and Π-abstraction
- ·) Successive λ- or Π-abstractions concerning the same type may be combined
- ·) Arrow types are right associative

Example: $\Pi \alpha, \beta : * . \alpha \to \beta \to \alpha \equiv (\Pi \alpha : * . (\Pi \beta : * . (\alpha \to (\beta \to \alpha))))$ — "$\equiv$" "stands for"

Definition: $\boxed{\text{Statement}}$: $M : \sigma$ or $\sigma : *$ (with $M \in \Lambda_{\mathbb{T}_2}$, $\sigma \in \mathbb{T}_2$)

$\boxed{\text{Declaration}}$: $x : \sigma$ or $\alpha : *$ (with $x \in V$, $\sigma \in \mathbb{T}_2$, $\alpha \in \mathbb{V}$)

$\boxed{\text{Context}}$ and $\boxed{\text{domain}}$:
1) $\emptyset$ is a $\lambda 2$-context, $\mathrm{dom}(\emptyset) = ()$ — empty list
2) Γ $\lambda 2$-context, $\alpha \in \mathbb{V}$, $\alpha \notin \mathrm{dom}(\Gamma)$ $\Rightarrow$ $\Gamma, \alpha : *$ $\lambda 2$-context, $\mathrm{dom}(\Gamma, \alpha : *) = (\mathrm{dom}(\Gamma), \alpha)$
3) Γ $\lambda 2$-context, $\sigma \in \mathbb{T}_2$, $\alpha \in \mathrm{dom}(\Gamma)$ for all free $\alpha \in \mathbb{V}$ in σ, $x \notin \mathrm{dom}(\Gamma)$ $\Rightarrow$ $\Gamma, x : \sigma$ $\lambda 2$-context, $\mathrm{dom}(\Gamma, x : \sigma) = (\mathrm{dom}(\Gamma), x)$

Examples: $\emptyset$; $\alpha : *$; $\alpha : *, x : \alpha \to \alpha$; $\alpha : *, x : \alpha \to \alpha, \beta : *$; $\Gamma \equiv \alpha : *, x : \alpha \to \alpha, \beta : *, y : (\alpha \to \alpha) \to \beta$ are $\lambda 2$-contexts, $\mathrm{dom}(\Gamma) = (\alpha, x, \beta, y)$

Derivation rules for $\lambda 2$

(var) (axiom) $\dfrac{}{\Gamma \vdash x : \sigma}$ if Γ is a $\lambda 2$-context and $x : \sigma \in \Gamma$

(appl) $\dfrac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$

(abst) $\dfrac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma . M : \sigma \to \tau}$

(form) (axiom) $\dfrac{}{\Gamma \vdash B : *}$ if Γ is a $\lambda 2$-context, $B \in \mathbb{T}_2$, and all free type-variables in B are declared in Γ — definition analogous to free variables

(appl$_2$) $\dfrac{\Gamma \vdash M : (\Pi \alpha : * . A) \qquad \Gamma \vdash B : *}{\Gamma \vdash MB : A[\alpha := B]}$ — need (form)-rule

(abst$_2$) $\dfrac{\Gamma, \alpha : * \vdash M : A}{\Gamma \vdash \lambda \alpha : * . M : \Pi \alpha : * . A}$

Definition: A $\lambda 2$-term M is called $\boxed{\text{legal}}$ if there exists a $\lambda 2$-context Γ and a $\lambda 2$-type σ s.t. $\Gamma \vdash M : \sigma$

Example of a derivation in $\lambda 2$

Let $M :\equiv \lambda \alpha : *. \lambda f : \alpha \to \alpha . \lambda x : \alpha . f(fx))$, $\Gamma :\equiv \emptyset$ (M has no free term- or type-variables)

Find type of M in context Γ:

(a) $\boxed{\alpha : *}$

⋮

(m) $\lambda f : \alpha \to \alpha . \lambda x : \alpha . f(fx) : ?$

(n) $\lambda \alpha : * . \lambda f : \alpha \to \alpha . \lambda x : \alpha . f(fx)) : \ldots$ (abst$_2$) on (m)

(a) $\boxed{\alpha : *}$

(b) $\boxed{f : \alpha \to \alpha}$

(c) $\boxed{x : \alpha}$

⋮

(k) $f(fx) : ?$ ← typing problem in $\lambda \to$

(l) $\lambda x : \alpha . f(fx)) : \ldots$ (abst) on (k)

(m) $\lambda f : \alpha \to \alpha . \lambda x : \alpha . f(fx) : \ldots$ (abst) on (l)

(n) $\lambda \alpha : * . \lambda f : \alpha \to \alpha . \lambda x : \alpha . f(fx) : \ldots$ (abst$_2$) on (m)

(a) $\boxed{\alpha : *}$

(b) $\boxed{f : \alpha \to \alpha}$

(c) $\boxed{x : \alpha}$

(1) $fx : \alpha$ (appl) on (b) and (c)

(2) $f(fx) : \alpha$ (appl) on (b) and (1)

(3) $\lambda x : \alpha . f(fx) : \alpha \to \alpha$ (abst) on (2)

(4) $\lambda f : \alpha \to \alpha . \lambda x : \alpha . f(fx) : (\alpha \to \alpha) \to \alpha \to \alpha$ (abst) on (3)

(5) $\lambda \alpha : * . \lambda f : \alpha \to \alpha . \lambda x : \alpha . f(fx) : \Pi \alpha : * . (\alpha \to \alpha) \to \alpha \to \alpha$ (abst$_2$) on (4)

So: $\emptyset \vdash \lambda \alpha : * . \lambda f : \alpha \to \alpha . \lambda x : \alpha . f(fx) : \Pi \alpha : * . (\alpha \to \alpha) \to \alpha \to \alpha$

(6) Thinning lemma (later) $\Rightarrow$ $\Gamma \vdash \lambda \alpha : * . \lambda f : \alpha \to \alpha . \lambda x : \alpha . f(fx) : \Pi \alpha : * . (\alpha \to \alpha) \to \alpha \to \alpha$ for every $\lambda 2$ context Γ

(7) $\Gamma \vdash nat : *$ (assumption)

(8) $\Gamma \vdash (\lambda \alpha : * . \lambda f : \alpha \to \alpha . \lambda x : \alpha . f(fx))\, nat : (nat \to nat) \to nat \to nat$ (appl$_2$) on (6), (7)

(9) $\Gamma \vdash suc : nat \to nat$ (assumption)

(10) $\Gamma \vdash (\lambda \alpha : * . \lambda f : \alpha \to \alpha . \lambda x : \alpha . f(fx))\, nat\, suc : nat \to nat$ (appl) on (8), (9)

(11) $\Gamma \vdash two : nat$ (assumption)

(12) $\Gamma \vdash (\lambda \alpha : * . \lambda f : \alpha \to \alpha . \lambda x : \alpha . f(fx))\, nat\, suc\, two : nat$ (appl) on (10), (11)