Intersection types fit well with resource control

Silvia Ghilezan, Jelena Ivetić, Pierre Lescanne, Silvia Likavec

University of Novi Sad, École normale supérieure de Lyon, Unviersità di Torino

TYPES 2015, 18 May

From practice to theory

With resource control we know when a variable is

- duplicated or
- erased.

For a compiler maker:

- duplication requires pointers and memory management, it should be used only when necessary,
- erasure allows to free memory explicitly.

For a proof theorist:

- duplication corresponds to contraction,
- erasure corresponds to thinning.

Well formed terms

The set $\Lambda_{\circledR}$ of terms are of the following forms:

x	*Variable*	*Axiom*
$\lambda x.M$	*Abstraction*	$\rightarrow$ *Introduction*
$M\,N$	*Application*	$\rightarrow$ *Elimination*
$x <^{x_1}_{x_2} M$	*Duplication*	*Contraction*
$x \odot M$	*Erasure*	*Thinning*

Moreover there are constraints on variables: variables occur once and only once.

A substitution is defined for this calculus.

The reduction rules of $\lambda_{®}$

Rules

$$
\begin{array}{rrcl}
(\beta) & (\lambda x.M)N & \rightarrow & M[N/\!\!/x] \\
(\gamma_1) & x <^{x_1}_{x_2} (\lambda y.M) & \rightarrow & \lambda y.x <^{x_1}_{x_2} M \\
(\gamma_2) & x <^{x_1}_{x_2} (MN) & \rightarrow & (x <^{x_1}_{x_2} M)N, \text{ if } x_1, x_2 \notin Fv(N) \\
(\gamma_3) & x <^{x_1}_{x_2} (MN) & \rightarrow & M(x <^{x_1}_{x_2} N), \text{ if } x_1, x_2 \notin Fv(M) \\
(\omega_1) & \lambda x.(y \odot M) & \rightarrow & y \odot (\lambda x.M), \; x \neq y \\
(\omega_2) & (x \odot M)N & \rightarrow & x \odot (MN) \\
(\omega_3) & M(x \odot N) & \rightarrow & x \odot (MN) \\
(\gamma\omega_1) & x <^{x_1}_{x_2} (y \odot M) & \rightarrow & y \odot (x <^{x_1}_{x_2} M), \; y \neq x_1, x_2 \\
(\gamma\omega_2) & x <^{x_1}_{x_2} (x_1 \odot M) & \rightarrow & M[x/\!\!/x_2]
\end{array}
$$

Structural equivalences

$$
\begin{array}{llll}
(\varepsilon_1) & x \odot (y \odot M) & \equiv_{\lambda_{®}} & y \odot (x \odot M) \\
(\varepsilon_2) & x <^{x_1}_{x_2} M & \equiv_{\lambda_{®}} & x <^{x_2}_{x_1} M \\
(\varepsilon_3) & x <^{y}_{z} (y <^{u}_{v} M) & \equiv_{\lambda_{®}} & x <^{y}_{u} (y <^{z}_{v} M) \\
(\varepsilon_4) & x <^{x_1}_{x_2} (y <^{y_1}_{y_2} M) & \equiv_{\lambda_{®}} & y <^{y_1}_{y_2} (x <^{x_1}_{x_2} M), \quad x \neq y_1, y_2, \ y \neq x_1, x_2
\end{array}
$$

Rules of $\lambda^{\cap}_{®}$

$$\frac{}{x : \sigma \vdash x : \sigma}\ (Ax)$$

$$\frac{\Gamma, x : \alpha \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \alpha \rightarrow \sigma}\ (\rightarrow_I)$$

$$\frac{\Gamma \vdash M : \cap_{i=1}^{n}\tau_i \rightarrow \sigma \qquad \Delta_0 \vdash N : \tau_0 \ \ldots\ \Delta_n \vdash N : \tau_n}{\Gamma, \Delta_0^{\top} \sqcap \Delta_1 \sqcap \ldots \sqcap \Delta_n \vdash MN : \sigma}\ (\rightarrow_E)$$

$$\frac{\Gamma, x : \alpha, y : \beta \vdash M : \sigma}{\Gamma, z : \alpha \cap \beta \vdash z <^{x}_{y} M : \sigma}\ (Cont)$$

$$\frac{\Gamma \vdash M : \sigma}{\Gamma, x : \top \vdash x \odot M : \sigma}\ (Thin)$$

Typing is syntax directed.

Fitness

- $\rightarrow$ is introduced by $(\rightarrow_I)$
- $\cap$ is introduced by $(Cont)$
- $\top$ is introduced by $(Thin)$.

Role of variables

Variable have several roles with a type associated to this role.

Roles

role	**type**
placeholder variables	strict
duplicated variables	intersection
erased variables	$\top$

The role of a variable may change by reduction.

Equivalence between strong normalisation and typeablity

SN $\Rightarrow$ Typeability in $\Lambda_{®}\cap$

Typeability in $\Lambda_{®}\cap \Rightarrow$ SN

The proof uses
the correspondence between

- typing rules and
- roles of variables.

Conclusion

The way variables are introduced $\Longrightarrow$ the way terms are typed.

No irrelevant intersection types are introduced.

The flexibility on the type (as in $\to_E$) comes from the choice in invoking the axiom.