

Higher-ranked Exception Types

Ruud Koot

February 12, 2015

1 The λ^{\cup} -calculus

Types

$$\begin{array}{lll} \tau \in \mathbf{Ty} & ::= & \mathcal{P} \quad \text{(base type)} \\ & | & \tau_1 \rightarrow \tau_2 \quad \text{(function type)} \end{array}$$

Terms

$$\begin{array}{lll} t \in \mathbf{Tm} & ::= & x, y, \dots \quad \text{(variable)} \\ & | & \lambda x : \tau. t \quad \text{(abstraction)} \\ & | & t_1 t_2 \quad \text{(application)} \\ & | & \emptyset \quad \text{(empty)} \\ & | & \{c\} \quad \text{(singleton)} \\ & | & t_1 \cup t_2 \quad \text{(union)} \end{array}$$

Values Values v are terms of the form

$$\lambda x_1 : \tau_1. \dots \lambda x_i : \tau_i. \{c_1\} \cup (\dots \cup (\{c_j\} \cup (x_1 v_{11} \dots v_{1m} \cup (\dots \cup x_k v_{k1} \dots v_{kn}))))$$

Environments

$$\Gamma \in \mathbf{Env} ::= \cdot \quad | \quad \Gamma, x : \tau$$

1.1 Typing relation

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} [\mathbf{T-VAR}] \quad \frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1. t : \tau_1 \rightarrow \tau_2} [\mathbf{T-ABS}] \quad \frac{\Gamma \vdash t_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1 t_2 : \tau_2} [\mathbf{T-APP}]$$

$$\frac{}{\Gamma \vdash \emptyset : \mathcal{P}} [\mathbf{T-EMPTY}] \quad \frac{}{\Gamma \vdash \{c\} : \mathcal{P}} [\mathbf{T-CON}] \quad \frac{\Gamma \vdash t_1 : \tau \quad \Gamma \vdash t_2 : \tau}{\Gamma \vdash t_1 \cup t_2 : \tau} [\mathbf{T-UNION}]$$

1.2 Semantics

1.3 Reduction relation

Definition 1. Let \prec be a strict total order on $\mathbf{Con} \cup \mathbf{Var}$, with $c \prec x$ for all $c \in \mathbf{Con}$ and $x \in \mathbf{Var}$.

$$\begin{aligned}
& (\lambda x : \tau. t_1) \ t_2 \longrightarrow [t_2/x] \ t_1 && (\beta\text{-reduction}) \\
& (t_1 \cup t_2) \ t_3 \longrightarrow t_1 \ t_3 \cup t_2 \ t_3 \\
& (\lambda x : \tau. t_1) \cup (\lambda x : \tau. t_2) \longrightarrow \lambda x : \tau. (t_1 \cup t_2) && (\text{congruences}) \\
& x \ t_1 \cdots t_n \cup x' \ t'_1 \cdots t'_n \longrightarrow x \ (t_1 \cup t'_1) \cdots (t_n \cup t'_n) \\
& (t_1 \cup t_2) \cup t_3 \longrightarrow t_1 \cup (t_2 \cup t_3) && (\text{associativity}) \\
& \emptyset \cup t \longrightarrow t \\
& t \cup \emptyset \longrightarrow t && (\text{unit}) \\
& x \cup x \longrightarrow x \\
& x \cup (x \cup t) \longrightarrow x \cup t && (\text{idempotence}) \\
& \{c\} \cup \{c\} \longrightarrow \{c\} \\
& \{c\} \cup (\{c\} \cup t) \longrightarrow \{c\} \cup t \\
& x \ t_1 \cdots t_n \cup \{c\} \longrightarrow \{c\} \cup x \ t_1 \cdots t_n && (1) \\
& x \ t_1 \cdots t_n \cup (\{c\} \cup t) \longrightarrow \{c\} \cup (x \ t_1 \cdots t_n \cup t) && (2) \\
& x \ t_1 \cdots t_n \cup x' \ t'_1 \cdots t'_n \longrightarrow x' \ t'_1 \cdots t'_n \cup x \ t_1 \cdots t_n && \text{if } x' \prec x \quad (3) \\
& x \ t_1 \cdots t_n \cup (x' \ t'_1 \cdots t'_n \cup t) \longrightarrow x' \ t'_1 \cdots t'_n \cup (x \ t_1 \cdots t_n \cup t) && \text{if } x' \prec x \quad (4) \\
& \{c\} \cup \{c'\} \longrightarrow \{c'\} \cup \{c\} && \text{if } c' \prec c \quad (5) \\
& \{c\} \cup (\{c'\} \cup t) \longrightarrow \{c'\} \cup (\{c\} \cup t) && \text{if } c' \prec c \quad (6)
\end{aligned}$$

Conjecture 1. The reduction relation \longrightarrow preserves meaning.

Conjecture 2. The reduction relation \longrightarrow is strongly normalizing.

Conjecture 3. The reduction relation \longrightarrow is locally confluent.

Corollary 1. The reduction relation \longrightarrow is confluent.

Proof. Follows from SN, LC and Newman's Lemma. \square

Corollary 2. The λ^\cup -calculus has unique normal forms.

Corollary 3. Equality of λ^\cup -terms can be decided by normalization.

2 Completion

$$\begin{aligned}
\kappa \in \mathbf{Kind} & ::= \mathbf{E} && (\text{exception}) \\
& \mid \kappa_1 \Rightarrow \kappa_2 && (\text{exception operator})
\end{aligned}$$

$\chi \in \mathbf{Exn}$	$::= e$	(exception variables)
	$ \lambda e : \kappa. \chi$	(exception abstraction)
$\hat{\tau} \in \mathbf{ExnTy}$	$::= \forall e :: \kappa. \hat{\tau}$	(exception quantification)
	$ \mathbf{bool}$	(boolean type)
	$ [\hat{\tau}\{\chi\}]$	(list type)
	$ \hat{\tau}_1\{\chi_1\} \rightarrow \hat{\tau}_2\{\chi_2\}$	(function type)

The completion procedure as a set of inference rules:

$$\begin{array}{c}
\frac{}{\overline{e_i :: \kappa_i} \vdash \mathbf{bool} : \mathbf{bool} \ \& \ e \ \overline{e_i} \triangleright e :: \overline{\kappa_i} \Rightarrow_{\mathbf{E}}} \text{[C-BOOL]} \\
\\
\frac{\overline{e_i :: \kappa_i} \vdash \tau : \hat{\tau} \ \& \ \chi \triangleright \overline{e_j} :: \overline{\kappa_j}}{\overline{e_i :: \kappa_i} \vdash [\tau] : [\hat{\tau}\{\chi\}] \ \& \ e \ \overline{e_i} \triangleright e :: \overline{\kappa_i} \Rightarrow_{\mathbf{E}}, \overline{e_j} :: \overline{\kappa_j}} \text{[C-LIST]} \\
\\
\frac{\vdash \tau_1 : \hat{\tau}_1 \ \& \ \chi_1 \triangleright \overline{e_j} :: \overline{\kappa_j} \quad \overline{e_i :: \kappa_i}, \overline{e_j :: \kappa_j} \vdash \tau_2 : \hat{\tau}_2 \ \& \ \chi_2 \triangleright \overline{e_j} :: \overline{\kappa_j}}{\overline{e_i :: \kappa_i} \vdash \tau_1 \rightarrow \tau_2 : \forall \overline{e_j} :: \overline{\kappa_j}. (\hat{\tau}_1\{\chi_1\} \rightarrow \hat{\tau}_2\{\chi_2\}) \ \& \ e \ \overline{e_i} \triangleright e :: \overline{\kappa_i} \Rightarrow_{\mathbf{E}}, \overline{e_k} :: \overline{\kappa_k}} \text{[C-ARR]}
\end{array}$$

Figure 1: Type completion ($\Gamma \vdash \tau : \hat{\tau} \ \& \ \chi \triangleright \Gamma'$)

The completion procedure as an algorithm:

```

complete :: Env × Ty → ExnTy × Exn × Env
complete  $\overline{e_i} :: \overline{\kappa_i}$  bool =
  let  $e$  be fresh
  in  $\langle \mathbf{bool}; e \ \overline{e_i}; e :: \overline{\kappa_i} \Rightarrow_{\mathbf{E}} \rangle$ 

```

3 Type system

$t \in \mathbf{Tm}$	$::= x$	(term variable)
	$ c$	(term constant)
	$ \lambda x : \tau. t$	(term abstraction)
	$ t_1 \ t_2$	(term application)
	$ t_1 \oplus t_2$	(operator)
	$ \mathbf{if} \ t_1 \ \mathbf{then} \ t_2 \ \mathbf{else} \ t_3$	(conditional)
	$ \ell^\ell$	(exception constant)
	$ t_1 \ \mathbf{seq} \ t_2$	(forcing)
	$ \mathbf{fix} \ t$	(anonymous fixpoint)
	$ []_\tau$	(nil constructor)
	$ t_1 :: t_2$	(cons constructor)
	$ \mathbf{case} \ t_1 \ \mathbf{of} \ \{ [] \mapsto t_2; x_1 :: x_2 \mapsto t_3 \}$	(list eliminator)