

# Higher-ranked Exception Types

Ruud Koot

February 23, 2015

## 1 The $\lambda^{\cup}$ -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  $\lambda^\cup$ -calculus has unique normal forms.

**Corollary 3.** Equality of  $\lambda^\cup$ -terms can be decided by normalization.

## 2 Completion

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

$\varphi \in \mathbf{Exn}$	$::=$	$e$	(exception variables)
		$ \ \lambda e : \kappa. \varphi$	(exception abstraction)
$\hat{\tau} \in \mathbf{ExnTy}$	$::=$	$\forall e :: \kappa. \hat{\tau}$	(exception quantification)
		$ \ \widehat{\mathbf{bool}}$	(boolean type)
		$ \ [\hat{\tau}(\varphi)]$	(list type)
		$ \ \hat{\tau}_1(\varphi_1) \rightarrow \hat{\tau}_2(\varphi_2)$	(function type)

The completion procedure as a set of inference rules:

$$\begin{array}{c}
\frac{}{\overline{e_i :: \kappa_i} \vdash \mathbf{bool} : \widehat{\mathbf{bool}} \ \& \ e \ \overline{e_i} \triangleright e :: \overline{\kappa_i} \Rightarrow_{\mathbf{E}}} \text{[C-BOOL]} \\
\\
\frac{\overline{e_i :: \kappa_i} \vdash \tau : \hat{\tau} \ \& \ \varphi \triangleright \overline{e_j :: \kappa_j}}{\overline{e_i :: \kappa_i} \vdash [\tau] : [\hat{\tau}(\varphi)] \ \& \ e \ \overline{e_i} \triangleright e :: \overline{\kappa_i} \Rightarrow_{\mathbf{E}}, \overline{e_j :: \kappa_j}} \text{[C-LIST]} \\
\\
\frac{\vdash \tau_1 : \hat{\tau}_1 \ \& \ \varphi_1 \triangleright \overline{e_j :: \kappa_j} \quad \overline{e_i :: \kappa_i}, \overline{e_j :: \kappa_j} \vdash \tau_2 : \hat{\tau}_2 \ \& \ \varphi_2 \triangleright \overline{e_j :: \kappa_j}}{\overline{e_i :: \kappa_i} \vdash \tau_1 \rightarrow \tau_2 : \forall \overline{e_j :: \kappa_j}. (\hat{\tau}_1(\varphi_1) \rightarrow \hat{\tau}_2(\varphi_2)) \ \& \ e \ \overline{e_i} \triangleright e :: \overline{\kappa_j} \Rightarrow_{\mathbf{E}}, \overline{e_k :: \kappa_k}} \text{[C-ARR]}
\end{array}$$

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

The completion procedure as an algorithm:

```

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

```

### 3 Type system

#### 3.1 Terms

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

#### 3.2 Underlying type system

$$\begin{array}{c}
\frac{}{\Gamma, x : \tau \vdash x : \tau} [\text{T-VAR}] \quad \frac{}{\Gamma \vdash c_\tau : \tau} [\text{T-CON}] \quad \frac{}{\Gamma \vdash \downarrow_\tau^\ell : \tau} [\text{T-CRASH}] \\
\\
\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1. t : \tau_1 \rightarrow \tau_2} [\text{T-ABS}] \quad \frac{\Gamma \vdash t_1 : \tau_2 \rightarrow \tau \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash t_1 t_2 : \tau} [\text{T-APP}] \\
\\
\frac{\Gamma \vdash t : \tau \rightarrow \tau}{\Gamma \vdash \mathbf{fix} \, t : \tau} [\text{T-FIX}] \\
\\
\frac{\Gamma \vdash t_1 : \mathbf{int} \quad \Gamma \vdash t_2 : \mathbf{int}}{\Gamma \vdash t_1 \oplus t_2 : \mathbf{bool}} [\text{T-OP}] \quad \frac{\Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash t_1 \mathbf{seq} \, t_2 : \tau_2} [\text{T-SEQ}] \\
\\
\frac{\Gamma \vdash t_1 : \mathbf{bool} \quad \Gamma \vdash t_2 : \tau \quad \Gamma \vdash t_3 : \tau}{\Gamma \vdash \mathbf{if} \, t_1 \mathbf{then} \, t_2 \mathbf{else} \, t_3 : \tau} [\text{T-IF}] \\
\\
\frac{}{\Gamma \vdash \square_\tau : [\tau]} [\text{T-NIL}] \quad \frac{\Gamma \vdash t_1 : \tau \quad \Gamma \vdash t_2 : [\tau]}{\Gamma \vdash t_1 :: t_2 : [\tau]} [\text{T-CONS}] \\
\\
\frac{\Gamma \vdash t_1 : [\tau_1] \quad \Gamma \vdash t_2 : \tau \quad \Gamma, x_1 : \tau_1, x_2 : [\tau_1] \vdash t_3 : \tau}{\Gamma \vdash \mathbf{case} \, t_1 \mathbf{of} \, \{\square \mapsto t_2; x_1 :: x_2 \mapsto t_3\} : \tau} [\text{T-CASE}]
\end{array}$$

Figure 2: Underlying type system ( $\Gamma \vdash t : \tau$ )

### 3.3 Declarative exception type system

- In T-Abs and T-AnnAbs, should the term-level term-abstraction also have an explicit effect annotation?
- In T-AnnAbs, might need a side condition stating that  $e$  is not free in  $\Delta$ .
- In T-App, note the double occurrence of  $\varphi$  when typing  $t_1$ . Is subeffecting sufficient here? Also note that we do *not* expect an exception variable in the left-hand side annotation of the function space constructor.
- In T-AnnApp, note the substitution. We will need a substitution lemma for annotations.
- In T-Fix, there might be some universal quantifiers in our way. Do annotation applications in  $t$  take care of this, already? Perhaps we do need to change **fix**  $t$  into a binding construct to resolve this? Also, there is some implicit subeffecting going on between the annotations and effect.
- In T-Case, note the use of explicit subeffecting. Can this be done using implicit subeffecting?
- For T-Sub, should we introduce a term-level coercion, as in Dussart–Henglein–Mossin? We now do atomic subtyping, is subeffecting sufficient?

### 3.4 Type elaboration system

### 3.5 Type inference algorithm

$$\begin{array}{c}
\overline{\Gamma, x : \hat{\tau} \& \varphi; \Delta \vdash x : \hat{\tau} \& \varphi} \text{ [T-VAR]} \\
\\
\overline{\Gamma; \Delta \vdash c_\tau : \perp_\tau \& \emptyset} \text{ [T-CON]} \quad \overline{\Gamma; \Delta \vdash \not\downarrow_\tau^\ell : \perp_\tau \& \{\ell\}} \text{ [T-CRASH]} \\
\\
\frac{\Gamma, x : \hat{\tau}_1 \& \varphi_1; \Delta \vdash t : \hat{\tau}_2 \& \varphi_2}{\Gamma; \Delta \vdash \lambda x : \hat{\tau}_1 \& \varphi_1. t : \hat{\tau}_1 \langle \varphi_1 \rangle \rightarrow \hat{\tau}_2 \langle \varphi_2 \rangle \& \emptyset} \text{ [T-ABS]} \\
\\
\frac{\Gamma; \Delta, e : \kappa \vdash t : \hat{\tau} \& \varphi}{\Gamma; \Delta \vdash \Lambda e : \kappa. t : \forall e : \kappa. \hat{\tau} \& \varphi} \text{ [T-ANNAbs]} \\
\\
\frac{\Gamma; \Delta \vdash t_1 : \hat{\tau}_2 \langle \varphi_2 \rangle \rightarrow \hat{\tau} \langle \varphi \rangle \& \varphi \quad \Gamma; \Delta \vdash t_2 : \hat{\tau}_2 \& \varphi_2}{\Gamma; \Delta \vdash t_1 t_2 : \hat{\tau} \& \varphi} \text{ [T-APP]} \\
\\
\frac{\Gamma; \Delta \vdash t_1 : \forall e : \kappa. \hat{\tau} \& \varphi \quad \Delta \vdash \varphi_2 : \kappa}{\Gamma; \Delta \vdash t_1 \langle \varphi_2 \rangle : [\varphi_2/e] \hat{\tau} \& \varphi} \text{ [T-ANNApP]} \\
\\
\frac{\Gamma; \Delta \vdash t : \hat{\tau} \langle \varphi \rangle \rightarrow \hat{\tau} \langle \varphi \rangle \& \varphi}{\Gamma; \Delta \vdash \mathbf{fix} \, t : \hat{\tau} \& \varphi} \text{ [T-FIX]} \\
\\
\frac{\Gamma; \Delta \vdash t_1 : \mathbf{i\hat{n}t} \& \varphi \quad \Gamma; \Delta \vdash t_2 : \mathbf{i\hat{n}t} \& \varphi}{\Gamma; \Delta \vdash t_1 \oplus t_2 : \mathbf{b\hat{o}ol} \& \varphi} \text{ [T-OP]} \\
\\
\frac{\Gamma; \Delta \vdash t_1 : \hat{\tau}_1 \& \varphi \quad \Gamma; \Delta \vdash t_2 : \hat{\tau}_2 \& \varphi}{\Gamma; \Delta \vdash t_1 \mathbf{seq} \, t_2 : \hat{\tau}_2 \& \varphi} \text{ [T-SEQ]} \\
\\
\frac{\Gamma; \Delta \vdash t_1 : \mathbf{b\hat{o}ol} \& \varphi \quad \Gamma; \Delta \vdash t_2 : \hat{\tau} \& \varphi \quad \Gamma; \Delta \vdash t_3 : \hat{\tau} \& \varphi}{\Gamma; \Delta \vdash \mathbf{if} \, t_1 \mathbf{then} \, t_2 \mathbf{else} \, t_3 : \hat{\tau} \& \varphi} \text{ [T-IF]} \\
\\
\overline{\Gamma; \Delta \vdash []_\tau : [\perp_\tau \langle \emptyset \rangle] \& \emptyset} \text{ [T-NIL]} \\
\\
\frac{\Gamma; \Delta \vdash t_1 : \hat{\tau} \& \varphi_1 \quad \Gamma; \Delta \vdash t_2 : [\hat{\tau} \langle \varphi_1 \rangle] \& \varphi_2}{\Gamma; \Delta \vdash t_1 :: t_2 : [\hat{\tau} \langle \varphi_1 \rangle] \& \varphi_2} \text{ [T-CONS]} \\
\\
\frac{\Gamma; \Delta \vdash t_1 : [\hat{\tau}_1 \langle \varphi_1 \rangle] \& \varphi' \quad \Delta \vdash \varphi' \leq \varphi \quad \Gamma; \Delta \vdash t_2 : \hat{\tau} \& \varphi \quad \Gamma, x_1 : \hat{\tau}_1 \& \varphi_1, x_2 : [\hat{\tau}_1 \langle \varphi_1 \rangle] \& \varphi'; \Delta \vdash t_3 : \hat{\tau} \& \varphi}{\Gamma; \Delta \vdash \mathbf{case} \, t_1 \mathbf{of} \, \{ [] \mapsto t_2; x_1 :: x_2 \mapsto t_3 \} : \hat{\tau} \& \varphi} \text{ [T-CASE]} \\
\\
\frac{\Gamma; \Delta \vdash t : \hat{\tau} \& \varphi \quad \Gamma; \Delta \vdash \hat{\tau} \leq \hat{\tau}' \quad \Delta \vdash \varphi \leq \varphi'}{\Gamma; \Delta \vdash t : \hat{\tau}' \& \varphi'} \text{ [T-SUB]}
\end{array}$$

Figure 3: Declarative type system ( $\Gamma; \Delta \vdash t : \hat{\tau} \& \varphi$ )

$$\begin{array}{c}
\overline{\Gamma, x : \widehat{\tau} \& \varphi \vdash x \rightsquigarrow x : \widehat{\tau} \& \varphi} \text{ [T-VAR]} \\
\\
\overline{\Gamma \vdash c_\tau \rightsquigarrow c_\tau : \tau \& \emptyset} \text{ [T-CON]} \quad \overline{\Gamma \vdash \not\downarrow_\tau^\ell \rightsquigarrow \not\downarrow_\tau^\ell : \perp_\tau \& \{\ell\}} \text{ [T-CRASH]} \\
\\
\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1. t : \tau_1 \rightarrow \tau_2} \text{ [T-ABS (TODO)]} \\
\\
\frac{\Gamma \vdash t_1 : \tau_2 \rightarrow \tau \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash t_1 t_2 : \tau} \text{ [T-APP (TODO)]} \\
\\
\frac{\Gamma \vdash t : \tau \rightarrow \tau}{\Gamma \vdash \mathbf{fix} \ t : \tau} \text{ [T-FIX (TODO)]} \\
\\
\frac{\Gamma \vdash t_1 \rightsquigarrow t'_1 : \mathbf{\hat{int}} \& \varphi_1 \quad \Gamma \vdash t_2 \rightsquigarrow t'_2 : \mathbf{\hat{int}} \& \varphi_2}{\Gamma \vdash t_1 \oplus t_2 \rightsquigarrow t'_1 \oplus t'_2 : \mathbf{\widehat{bool}} \& \varphi_1 \cup \varphi_2} \text{ [T-OP]} \\
\\
\frac{\Gamma \vdash t_1 \rightsquigarrow t'_1 : \widehat{\tau}_1 \& \varphi_1 \quad \Gamma \vdash t_2 \rightsquigarrow t'_2 : \widehat{\tau}_2 \& \varphi_2}{\Gamma \vdash t_1 \mathbf{seq} \ t_2 \rightsquigarrow t'_1 \mathbf{seq} \ t'_2 : \widehat{\tau}_2 \& \varphi_1 \cup \varphi_2} \text{ [T-SEQ]} \\
\\
\frac{\Gamma \vdash t_1 \rightsquigarrow t'_1 : \mathbf{\widehat{bool}} \& \varphi_1 \quad \Gamma \vdash t_2 \rightsquigarrow t'_2 : \widehat{\tau}_2 \& \varphi_2 \quad \Gamma \vdash t_3 \rightsquigarrow t'_3 : \widehat{\tau}_3 \& \varphi_3}{\Gamma \vdash \mathbf{if} \ t_1 \mathbf{then} \ t_2 \mathbf{else} \ t_3 \rightsquigarrow \mathbf{if} \ t'_1 \mathbf{then} \ t'_2 \mathbf{else} \ t'_3 : \widehat{\tau}_2 \sqcup \widehat{\tau}_3 \& \varphi_1 \cup \varphi_2 \cup \varphi_3} \text{ [T-IF]} \\
\\
\overline{\Gamma \vdash []_\tau \rightsquigarrow []_\tau : \perp_\tau \& \emptyset} \text{ [T-NIL]} \\
\\
\frac{\Gamma \vdash t_1 \rightsquigarrow t'_1 : \widehat{\tau}_1 \& \varphi_1 \quad \Gamma \vdash t_2 \rightsquigarrow t'_2 : [\widehat{\tau}'_1 \langle \varphi'_1 \rangle] \& \varphi_2}{\Gamma \vdash t_1 :: t_2 \rightsquigarrow t'_1 :: t'_2 : [\widehat{\tau}_1 \sqcup \widehat{\tau}'_1 \langle \varphi_1 \cup \varphi'_1 \rangle] \& \varphi_2} \text{ [T-CONS]} \\
\\
\frac{\Gamma \vdash t_1 \rightsquigarrow t'_1 : [\tau_1 \langle \varphi_1 \rangle] \& \varphi'_1 \quad \Gamma \vdash t_2 \rightsquigarrow t'_2 : \widehat{\tau}_2 \& \varphi_2 \quad \Gamma, x_1 : \widehat{\tau}_1 \& \varphi_1, x_2 : [\tau_1 \langle \varphi_1 \rangle] \& \varphi'_1 \vdash t_3 \rightsquigarrow t'_3 : \widehat{\tau}_3 \& \varphi_3}{\Gamma \vdash \mathbf{case} \ t_1 \mathbf{of} \ \{\ [] \mapsto t_2; x_1 :: x_2 \mapsto t_3 \} \rightsquigarrow \mathbf{case} \ t'_1 \mathbf{of} \ \{\ [] \mapsto t'_2; x_1 :: x_2 \mapsto t'_3 \} : \widehat{\tau}_2 \sqcup \widehat{\tau}_3 \& \varphi'_1 \cup \varphi_2 \cup \varphi_3} \text{ [T-CASE]}
\end{array}$$

Figure 4: Syntax-directed type elaboration system ( $\Gamma \vdash t \rightsquigarrow t' : \widehat{\tau} \& \varphi$ )