Higher-ranked Exception Types

Ruud Koot

June 16, 2015

1 Motivation

– Untracked exceptions can break information flow security.

2 The $\lambda^{\cup}$-calculus

Types

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

Terms

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

Values Values v are terms of the form

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

Environments

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

2.1 Typing relation

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}\ [\text{T-Var}] \quad \frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1 . t : \tau_1 \to \tau_2}\ [\text{T-Abs}] \quad \frac{\Gamma \vdash t_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\ t_2 : \tau_2}\ [\text{T-App}]$$

$$\frac{}{\Gamma \vdash \emptyset : \mathcal{P}}\ [\text{T-Empty}] \quad \frac{}{\Gamma \vdash \{c\} : \mathcal{P}}\ [\text{T-Con}] \quad \frac{\Gamma \vdash t_1 : \tau \quad \Gamma \vdash t_2 : \tau}{\Gamma \vdash t_1 \cup t_2 : \tau}\ [\text{T-Union}]$$

2.2 Semantics

2.3 Reduction relation

- To do. Do not match the rules in the prototype (those are sensitive to the order in which they are tried).
- To do. In the second rule only one term is applied; contrast this with the other rules involing applications.
- To do. Should make use of the fact the everything is fully applied (and η-expanded/-long?): all atoms are of the form $k\ \overline{t_i}$, where k is c or x and the number of arguments fixed by the arity of k. Then try to factor out the commutativity rules by taking "sets" of these atoms. That might simplify stuff a whole lot...
- To do. Can we restrict the typing rule T-Union to only allow sets and not functions on both sides? This would remove the 2nd and 3rd rewrite rules and make the system a more traditional higher-order rewrite system: it's "just" higher-order pattern E-unification (decidable), boolean rings are easy to integrate, and higher-ranked dimension types becomes higher-order E-unification (semi-decidable). Open question: how to represent e.g. $U(e_2(e_1), e_1) = [e_2 \mapsto \lambda e_1.e_1]$ without abstractions? (Reinterpret e_1 as $f(e_1)$ with $f = id$?)

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 $x \in$* $\mathbf{Var}$.

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

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. □

Corollary 2. *The $\lambda^{\cup}$-calculus has unique normal forms.*

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

3 Completion

$$
\begin{array}{llll}
\kappa \in \mathbf{Kind} & ::= & \textsc{exn} & \text{(exception)} \\
 & \mid & \kappa_1 \Rightarrow \kappa_2 & \text{(exception operator)} \\[1em]
\varphi \in \mathbf{Exn} & ::= & e & \text{(exception variables)} \\
 & \mid & \lambda e : \kappa.\varphi & \text{(exception abstraction)} \\[1em]
\widehat{\tau} \in \mathbf{ExnTy} & ::= & \forall e :: \kappa.\widehat{\tau} & \text{(exception quantification)} \\
 & \mid & \widehat{\mathrm{bool}} & \text{(boolean type)} \\
 & \mid & [\widehat{\tau}\langle\varphi\rangle] & \text{(list type)} \\
 & \mid & \widehat{\tau}_1\langle\varphi_1\rangle \to \widehat{\tau}_2\langle\varphi_2\rangle & \text{(function type)}
\end{array}
$$

The completion procedure as a set of inference rules:

$$
\frac{}{\overline{e_i :: \kappa_i} \vdash \mathbf{bool} : \widehat{\mathrm{bool}}\ \&\ e\ \overline{e_i} \rhd e :: \overline{\kappa_i \Rightarrow} \textsc{exn}}\ \text{[C-Bool]}
$$

$$
\frac{\overline{e_i :: \kappa_i} \vdash \tau : \widehat{\tau}\ \&\ \varphi \rhd \overline{e_j :: \kappa_j}}{\overline{e_i :: \kappa_i} \vdash [\tau] : [\widehat{\tau}\langle\varphi\rangle]\ \&\ e\ \overline{e_i} \rhd e :: \overline{\kappa_i \Rightarrow} \textsc{exn}, \overline{e_j :: \kappa_j}}\ \text{[C-List]}
$$

$$
\frac{\vdash \tau_1 : \widehat{\tau}_1\ \&\ \varphi_1 \rhd \overline{e_j :: \kappa_j} \qquad \overline{e_i :: \kappa_i}, \overline{e_j :: \kappa_j} \vdash \tau_2 : \widehat{\tau}_2\ \&\ \varphi_2 \rhd \overline{e_j :: \kappa_j}}{\overline{e_i :: \kappa_i} \vdash \tau_1 \to \tau_2 : \forall \overline{e_j :: \kappa_j}.\,(\widehat{\tau}_1\langle\varphi_1\rangle \to \widehat{\tau}_2\langle\varphi_2\rangle)\ \&\ e\ \overline{e_i} \rhd e :: \overline{\kappa_j \Rightarrow} \textsc{exn}, \overline{e_k :: \kappa_k}}\ \text{[C-Arr]}
$$

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

The completion procedure as an algorithm:

$\mathcal{C} :: \mathbf{Env} \times \mathbf{Ty} \to \mathbf{ExnTy} \times \mathbf{Exn} \times \mathbf{Env}$
$\mathcal{C}\ \overline{e_i :: \kappa_i}\ \mathbf{bool} =$
  let *e be fresh*
  in $\langle \widehat{\mathrm{bool}}; e\ \overline{e_i}; e :: \overline{\kappa_i \Rightarrow} \textsc{exn} \rangle$

4 Type system

4.1 Terms

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

4.2 Underlying type system

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}\ [\text{T-Var}] \qquad \frac{}{\Gamma \vdash c_\tau : \tau}\ [\text{T-Con}] \qquad \frac{}{\Gamma \vdash \lightning^{\ell}_{\tau} : \tau}\ [\text{T-Crash}]$$

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1 . t : \tau_1 \to \tau_2}\ [\text{T-Abs}] \qquad \frac{\Gamma \vdash t_1 : \tau_2 \to \tau \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash t_1\ t_2 : \tau}\ [\text{T-App}]$$

$$\frac{\Gamma \vdash t : \tau \to \tau}{\Gamma \vdash \mathbf{fix}\ t : \tau}\ [\text{T-Fix}]$$

$$\frac{\Gamma \vdash t_1 : \mathbf{int} \quad \Gamma \vdash t_2 : \mathbf{int}}{\Gamma \vdash t_1 \oplus t_2 : \mathbf{bool}}\ [\text{T-Op}] \qquad \frac{\Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash t_1\ \mathbf{seq}\ t_2 : \tau_2}\ [\text{T-Seq}]$$

$$\frac{\Gamma \vdash t_1 : \mathbf{bool} \quad \Gamma \vdash t_2 : \tau \quad \Gamma \vdash t_3 : \tau}{\Gamma \vdash \mathbf{if}\ t_1\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3 : \tau}\ [\text{T-If}]$$

$$\frac{}{\Gamma \vdash [\,]_\tau : [\tau]}\ [\text{T-Nil}] \qquad \frac{\Gamma \vdash t_1 : \tau \quad \Gamma \vdash t_2 : [\tau]}{\Gamma \vdash t_1 :: t_2 : [\tau]}\ [\text{T-Cons}]$$

$$\frac{\Gamma \vdash t_1 : [\tau_1] \quad \Gamma \vdash t_2 : \tau \quad \Gamma, x_1 : \tau_1, x_2 : [\tau_1] \vdash t_3 : \tau}{\Gamma \vdash \mathbf{case}\ t_1\ \mathbf{of}\ \{[\,] \mapsto t_2; x_1 :: x_2 \mapsto t_3\} : \tau}\ [\text{T-Case}]$$

4.3 Declarative exception type system

$$\frac{}{\Gamma, x : \widehat{\tau}\ \&\ \varphi; \Delta \vdash x : \widehat{\tau}\ \&\ \varphi}\ [\text{T-Var}]$$

$$\frac{}{\Gamma; \Delta \vdash c_\tau : \bot_\tau\ \&\ \emptyset}\ [\text{T-Con}] \qquad \frac{}{\Gamma; \Delta \vdash \lightning^{\ell}_{\tau} : \bot_\tau\ \&\ \{\ell\}}\ [\text{T-Crash}]$$

$$\frac{\Gamma, x : \widehat{\tau}_1\ \&\ \varphi_1; \Delta \vdash t : \widehat{\tau}_2\ \&\ \varphi_2}{\Gamma; \Delta \vdash \lambda x : \widehat{\tau}_1\ \&\ \varphi_1.t : \widehat{\tau}_1\langle\varphi_1\rangle \to \widehat{\tau}_2\langle\varphi_2\rangle\ \&\ \emptyset}\ [\text{T-Abs}]$$

$$\frac{\Gamma; \Delta, e : \kappa \vdash t : \widehat{\tau}\ \&\ \varphi \quad e \notin \mathrm{fv}(\varphi)}{\Gamma; \Delta \vdash \Lambda e : \kappa.t : \forall e : \kappa.\widehat{\tau}\ \&\ \varphi}\ [\text{T-AnnAbs}]$$

$$\frac{\Gamma; \Delta \vdash t_1 : \widehat{\tau}_2\langle\varphi_2\rangle \to \widehat{\tau}\langle\varphi\rangle\ \&\ \varphi \quad \Gamma; \Delta \vdash t_2 : \widehat{\tau}_2\ \&\ \varphi_2}{\Gamma; \Delta \vdash t_1\ t_2 : \widehat{\tau}\ \&\ \varphi}\ [\text{T-App}]$$

$$\frac{\Gamma; \Delta \vdash t_1 : \forall e : \kappa.\widehat{\tau}\ \&\ \varphi \quad \Delta \vdash \varphi_2 : \kappa}{\Gamma; \Delta \vdash t_1\ \langle\varphi_2\rangle : \widehat{\tau}[\varphi_2/e]\ \&\ \varphi}\ [\text{T-AnnApp}]$$

$$\frac{\Gamma; \Delta \vdash t : \widehat{\tau}\langle\varphi'\rangle \to \widehat{\tau}\langle\varphi'\rangle\ \&\ \varphi'' \quad \Delta \vdash \varphi' \leqslant \varphi \quad \Delta \vdash \varphi'' \leqslant \varphi}{\Gamma; \Delta \vdash \mathbf{fix}\ t : \widehat{\tau}\ \&\ \varphi}\ [\text{T-Fix}]$$

$$\frac{\Gamma; \Delta \vdash t_1 : \widehat{\mathbf{int}}\ \&\ \varphi \quad \Gamma; \Delta \vdash t_2 : \widehat{\mathbf{int}}\ \&\ \varphi}{\Gamma; \Delta \vdash t_1 \oplus t_2 : \widehat{\mathbf{bool}}\ \&\ \varphi}\ [\text{T-Op}]$$

$$\frac{\Gamma; \Delta \vdash t_1 : \widehat{\tau}_1\ \&\ \varphi \quad \Gamma; \Delta \vdash t_2 : \widehat{\tau}_2\ \&\ \varphi}{\Gamma; \Delta \vdash t_1\ \mathbf{seq}\ t_2 : \widehat{\tau}_2\ \&\ \varphi}\ [\text{T-Seq}]$$

$$\frac{\Gamma; \Delta \vdash t_1 : \widehat{\mathbf{bool}}\ \&\ \varphi \quad \Gamma; \Delta \vdash t_2 : \widehat{\tau}\ \&\ \varphi \quad \Gamma; \Delta \vdash t_3 : \widehat{\tau}\ \&\ \varphi}{\Gamma; \Delta \vdash \mathbf{if}\ t_1\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3 : \widehat{\tau}\ \&\ \varphi}\ [\text{T-If}]$$

$$\frac{}{\Gamma; \Delta \vdash []_\tau : [\bot_\tau\langle\emptyset\rangle]\ \&\ \emptyset}\ [\text{T-Nil}]$$

$$\frac{\Gamma; \Delta \vdash t_1 : \widehat{\tau}\ \&\ \varphi_1 \quad \Gamma; \Delta \vdash t_2 : [\widehat{\tau}\langle\varphi_1\rangle]\ \&\ \varphi_2}{\Gamma; \Delta \vdash t_1 :: t_2 : [\widehat{\tau}\langle\varphi_1\rangle]\ \&\ \varphi_2}\ [\text{T-Cons}]$$

$$\frac{\begin{array}{c}\Gamma; \Delta \vdash t_1 : [\widehat{\tau}_1\langle\varphi_1\rangle]\ \&\ \varphi' \quad \Delta \vdash \varphi' \leqslant \varphi \quad \Gamma; \Delta \vdash t_2 : \widehat{\tau}\ \&\ \varphi \\ \Gamma, x_1 : \widehat{\tau}_1\ \&\ \varphi_1, x_2 : [\widehat{\tau}_1\langle\varphi_1\rangle]\ \&\ \varphi'; \Delta \vdash t_3 : \widehat{\tau}\ \&\ \varphi\end{array}}{\Gamma; \Delta \vdash \mathbf{case}\ t_1\ \mathbf{of}\ \{[] \mapsto t_2; x_1 :: x_2 \mapsto t_3\} : \widehat{\tau}\ \&\ \varphi}\ [\text{T-Case}]$$

$$\frac{\Gamma; \Delta \vdash t : \widehat{\tau}'\ \&\ \varphi' \quad \Delta \vdash \widehat{\tau}' \leqslant \widehat{\tau} \quad \Delta \vdash \varphi' \leqslant \varphi}{\Gamma; \Delta \vdash t : \widehat{\tau}\ \&\ \varphi}\ [\text{T-Sub}]$$

- 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 Δ.

- In T-App, note the double occurence of φ 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, the 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 shape-conformant subtyping, is subeffecting sufficient?

- Do we need additional kinding judgements in some of the rules? Can we merge the kinding judgement with the subtyping and/or -effecting judgement? Kind-preserving substitutions.

4.4 Type elaboration system

- In T-App and T-Fix, note that there are substitutions in the premises of the rules. Are these inductive? (Probably, as these premises are not "recursive" ones.)

$$\frac{}{\Gamma, x : \widehat{\tau}\ \&\ \varphi; \Delta \vdash x \hookrightarrow x : \widehat{\tau}\ \&\ \varphi}\ [\text{T-Var}]$$

$$\frac{}{\Gamma; \Delta \vdash c_\tau \hookrightarrow c_\tau : \tau\ \&\ \varnothing}\ [\text{T-Con}] \qquad \frac{}{\Gamma; \Delta \vdash \lightning^\ell_\tau \hookrightarrow \lightning^\ell_\tau : \bot_\tau\ \&\ \{\ell\}}\ [\text{T-Crash}]$$

$$\frac{\begin{array}{c}\Delta, \overline{e_i : \kappa_i} \vdash \widehat{\tau}_1 \triangleright \tau_1 \quad \Delta, \overline{e_i : \kappa_i} \vdash \varphi_1 : \textsc{exn} \\ \Gamma, x : \widehat{\tau}_1\ \&\ \varphi_1; \Delta, \overline{e_i : \kappa_i} \vdash t \hookrightarrow t' : \widehat{\tau}_2\ \&\ \varphi_2\end{array}}{\Gamma; \Delta \vdash \lambda x : \tau_1.t \hookrightarrow \Lambda\overline{e_i : \kappa_i}.\lambda x : \widehat{\tau}_1\ \&\ \varphi_1.t' : \forall \overline{e_i : \kappa_i}.\widehat{\tau}_1\langle\varphi_1\rangle \to \widehat{\tau}_2\langle\varphi_2\rangle\ \&\ \varnothing}\ [\text{T-Abs}]$$

$$\frac{\begin{array}{c}\Delta \vdash \widehat{\tau}_2 \leqslant \widehat{\tau}[\overline{\varphi_i}/\overline{e_i}] \quad \Delta \vdash \varphi_2 \leqslant \varphi[\overline{\varphi_i}/\overline{e_i}] \quad \overline{\Delta \vdash \varphi_i : \kappa_i} \\ \Gamma; \Delta \vdash t_1 \hookrightarrow t_1' : \forall \overline{e_i : \kappa_i}.\widehat{\tau}_1\langle\varphi_1\rangle \to \widehat{\tau}\langle\varphi\rangle\ \&\ \varphi' \quad \Gamma; \Delta \vdash t_2 \hookrightarrow t_2' : \widehat{\tau}_2\ \&\ \varphi_2\end{array}}{\Gamma; \Delta \vdash t_1\ t_2 \hookrightarrow t_1'\ \langle\overline{\varphi_i}\rangle\ t_2' : \widehat{\tau}[\overline{\varphi_i}/\overline{e_i}]\ \&\ \varphi[\overline{\varphi_i}/\overline{e_i}] \cup \varphi'}\ [\text{T-App}]$$

$$\frac{\begin{array}{c}\Gamma; \Delta \vdash t \hookrightarrow t' : \forall \overline{e_i : \kappa_i}.\widehat{\tau}\langle\varphi\rangle \to \widehat{\tau}'\langle\varphi'\rangle\ \&\ \varphi'' \\ \Delta \vdash \widehat{\tau}'[\overline{\varphi_i}/\overline{e_i}] \leqslant \widehat{\tau}[\overline{\varphi_i}/\overline{e_i}] \quad \Delta \vdash \varphi'[\overline{\varphi_i}/\overline{e_i}] \leqslant \varphi[\overline{\varphi_i}/\overline{e_i}] \quad \overline{\Delta \vdash \varphi_i : \kappa_i}\end{array}}{\Gamma; \Delta \vdash \mathbf{fix}\ t \hookrightarrow \mathbf{fix}\ t'\ \langle\overline{\varphi_i}\rangle : \widehat{\tau}[\overline{\varphi_i}/\overline{e_i}]\ \&\ \varphi[\overline{\varphi_i}/\overline{e_i}] \cup \varphi''}\ [\text{T-Fix}]$$

$$\frac{\Gamma; \Delta \vdash t_1 \hookrightarrow t_1' : \widehat{\mathbf{int}}\ \&\ \varphi_1 \quad \Gamma; \Delta \vdash t_2 \hookrightarrow t_2' : \widehat{\mathbf{int}}\ \&\ \varphi_2}{\Gamma; \Delta \vdash t_1 \oplus t_2 \hookrightarrow t_1' \oplus t_2' : \widehat{\mathbf{bool}}\ \&\ \varphi_1 \cup \varphi_2}\ [\text{T-Op}]$$

$$\frac{\Gamma; \Delta \vdash t_1 \hookrightarrow t_1' : \widehat{\tau}_1\ \&\ \varphi_1 \quad \Gamma; \Delta \vdash t_2 \hookrightarrow t_2' : \widehat{\tau}_2\ \&\ \varphi_2}{\Gamma; \Delta \vdash t_1\ \mathbf{seq}\ t_2 \hookrightarrow t_1'\ \mathbf{seq}\ t_2' : \widehat{\tau}_2\ \&\ \varphi_1 \cup \varphi_2}\ [\text{T-Seq}]$$

$$\frac{\Gamma; \Delta \vdash t_1 \hookrightarrow t_1' : \widehat{\mathbf{bool}}\ \&\ \varphi_1 \quad \Gamma; \Delta \vdash t_2 \hookrightarrow t_2' : \widehat{\tau}_2\ \&\ \varphi_2 \quad \Gamma; \Delta \vdash t_3 \hookrightarrow t_3' : \widehat{\tau}_3\ \&\ \varphi_3}{\Gamma; \Delta \vdash \mathbf{if}\ t_1\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3 \hookrightarrow \mathbf{if}\ t_1'\ \mathbf{then}\ t_2'\ \mathbf{else}\ t_3' : \widehat{\tau}_2 \sqcup \widehat{\tau}_3\ \&\ \varphi_1 \cup \varphi_2 \cup \varphi_3}\ [\text{T-If}]$$

$$\frac{}{\Gamma; \Delta \vdash [\,]_\tau \hookrightarrow [\,]_\tau : \bot_\tau\ \&\ \varnothing}\ [\text{T-Nil}]$$

$$\frac{\Gamma; \Delta \vdash t_1 \hookrightarrow t_1' : \widehat{\tau}_1\ \&\ \varphi_1 \quad \Gamma; \Delta \vdash t_2 \hookrightarrow t_2' : \left[\widehat{\tau}_1'\langle\varphi_1'\rangle\right]\ \&\ \varphi_2}{\Gamma; \Delta \vdash t_1 :: t_2 \hookrightarrow t_1' :: t_2' : \left[\widehat{\tau}_1 \sqcup \widehat{\tau}_1'\langle\varphi_1 \cup \varphi_1'\rangle\right]\ \&\ \varphi_2}\ [\text{T-Cons}]$$

$$\frac{\begin{array}{c}\Gamma; \Delta \vdash t_1 \hookrightarrow t_1' : [\tau_1\langle\varphi_1\rangle]\ \&\ \varphi_1' \quad \Gamma; \Delta \vdash t_2 \hookrightarrow t_2' : \widehat{\tau}_2\ \&\ \varphi_2 \\ \Gamma, x_1 : \widehat{\tau}_1\ \&\ \varphi_1, x_2 : [\tau_1\langle\varphi_1\rangle]\ \&\ \varphi_1'; \Delta \vdash t_3 \hookrightarrow t_3' : \widehat{\tau}_3\ \&\ \varphi_3\end{array}}{\begin{array}{l}\Gamma; \Delta \vdash \mathbf{case}\ t_1\ \mathbf{of}\ \{[\,] \mapsto t_2; x_1 :: x_2 \mapsto t_3\} \hookrightarrow \\ \qquad\qquad \mathbf{case}\ t_1'\ \mathbf{of}\ \{[\,] \mapsto t_2'; x_1 :: x_2 \mapsto t_3'\} : \widehat{\tau}_2 \sqcup \widehat{\tau}_3\ \&\ \varphi_1' \cup \varphi_2 \cup \varphi_3\end{array}}\ [\text{T-Case}]$$

– For T-Fix: how would a binding fixpoint construct work?

4.5 Type inference algorithm

$$
\begin{array}{l}
\mathcal{R} : \mathbf{TyEnv} \times \mathbf{KiEnv} \times \mathbf{Tm} \to \mathbf{ExnTy} \times \mathbf{Exn} \\
\mathcal{R}\ \Gamma\ \Delta\ x \qquad\qquad = \Gamma_x \\
\mathcal{R}\ \Gamma\ \Delta\ c_\tau \qquad\quad\ = \langle \bot_\tau ; \emptyset \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ \lightning^{\ell}_{\tau} \qquad\quad = \langle \bot_\tau ; \{\ell\} \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ (\lambda x : \tau . t) = \mathbf{let}\ \langle \widehat{\tau}_1 ; e_1 ; \overline{e_i : \kappa_i} \rangle = \mathcal{C}\ \emptyset\ \tau \\
\qquad\qquad\qquad\qquad\qquad\ \ \langle \widehat{\tau}_2 ; \varphi_2 \rangle \qquad\quad = \mathcal{R}\ (\Gamma, x : \widehat{\tau}_1\ \&\ e_1)\ (\Delta, \overline{e_i : \kappa_i})\ t \\
\qquad\qquad\qquad\qquad\quad\ \mathbf{in}\ \ \langle \forall \overline{e_i : \kappa_i} . \widehat{\tau}_1 \langle e_1 \rangle \to \widehat{\tau}_2 \langle \varphi_2 \rangle ; \emptyset \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ (t_1\ t_2) \quad\ = \mathbf{let}\ \langle \widehat{\tau}_1 ; \varphi_1 \rangle \qquad\qquad\qquad\quad = \mathcal{R}\ \Gamma\ \Delta\ t_1 \\
\qquad\qquad\qquad\qquad\qquad\ \ \langle \widehat{\tau}_2 ; \varphi_2 \rangle \qquad\qquad\qquad\quad = \mathcal{R}\ \Gamma\ \Delta\ t_2 \\
\qquad\qquad\qquad\qquad\qquad\ \ \langle \widehat{\tau}_2' \langle e_2' \rangle \to \widehat{\tau}' \langle \varphi' \rangle ; \overline{e_i : \kappa_i} \rangle = \mathcal{I}\ \widehat{\tau}_1 \\
\qquad\qquad\qquad\qquad\qquad\ \ \theta \qquad\qquad\qquad\qquad\quad\ = [e_2' \mapsto \varphi_2] \circ \mathcal{M}\ \emptyset\ \widehat{\tau}_2\ \widehat{\tau}_2' \\
\qquad\qquad\qquad\qquad\quad\ \mathbf{in}\ \ \langle [\![\theta \widehat{\tau}']\!]_\Delta ; [\![\theta \varphi' \cup \varphi_1]\!]_\Delta \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ (\mathbf{fix}\ t) \quad\ = \mathbf{let}\ \langle \widehat{\tau} ; \varphi \rangle \qquad\qquad\qquad\qquad = \mathcal{R}\ \Gamma\ \Delta\ t \\
\qquad\qquad\qquad\qquad\qquad\ \ \langle \widehat{\tau}' \langle e' \rangle \to \widehat{\tau}'' \langle \varphi'' \rangle ; \overline{e_i : \kappa_i} \rangle = \mathcal{I}\ \widehat{\tau} \\
\qquad\qquad\qquad\qquad\quad\ \mathbf{in}\ \ \langle \widehat{\tau}_0 ; \varphi_0 ; i \rangle \leftarrow \langle \bot_{\lfloor \widehat{\tau}' \rfloor} ; \emptyset ; 0 \rangle \\
\qquad\qquad\qquad\qquad\qquad\ \ \mathbf{do}\ \theta \qquad\qquad\quad \leftarrow [e' \mapsto \varphi_i] \circ \mathcal{M}\ \emptyset\ \widehat{\tau}_i\ \widehat{\tau}' \\
\qquad\qquad\qquad\qquad\qquad\qquad \langle \widehat{\tau}_{i+1} ; \varphi_{i+1} ; i \rangle \leftarrow \langle [\![\theta \widehat{\tau}'']\!]_\Delta ; [\![\theta \varphi'']\!]_\Delta ; i + 1 \rangle \\
\qquad\qquad\qquad\qquad\qquad\ \ \mathbf{until}\ \langle \widehat{\tau}_i ; \varphi_i \rangle \equiv \langle \widehat{\tau}_{i-1} ; \varphi_{i-1} \rangle \\
\qquad\qquad\qquad\qquad\qquad\ \ \mathbf{return}\ \langle \widehat{\tau}_i ; [\![\varphi \cup \varphi_i]\!]_\Delta \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ (t_1 \oplus t_2) \ = \mathbf{let}\ \langle \widehat{\mathbf{int}} ; \varphi_1 \rangle = \mathcal{R}\ \Gamma\ \Delta\ t_1 \\
\qquad\qquad\qquad\qquad\qquad\ \ \langle \widehat{\mathbf{int}} ; \varphi_2 \rangle = \mathcal{R}\ \Gamma\ \Delta\ t_2 \\
\qquad\qquad\qquad\qquad\quad\ \mathbf{in}\ \ \langle \widehat{\mathbf{bool}} ; [\![\varphi_1 \cup \varphi_2]\!]_\Delta \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ (t_1\ \mathbf{seq}\ t_2) \\
\qquad\qquad\qquad\qquad\quad = \mathbf{let}\ \langle \widehat{\tau}_1 ; \varphi_1 \rangle = \mathcal{R}\ \Gamma\ \Delta\ t_1 \\
\qquad\qquad\qquad\qquad\qquad\ \ \langle \widehat{\tau}_2 ; \varphi_2 \rangle = \mathcal{R}\ \Gamma\ \Delta\ t_2 \\
\qquad\qquad\qquad\qquad\quad\ \mathbf{in}\ \ \langle \widehat{\tau}_2 ; [\![\varphi_1 \cup \varphi_2]\!]_\Delta \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ (\mathbf{if}\ t_1\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3) \\
\qquad\qquad\qquad\qquad\quad = \mathbf{let}\ \langle \widehat{\mathbf{bool}} ; \varphi_1 \rangle = \mathcal{R}\ \Gamma\ \Delta\ t_1 \\
\qquad\qquad\qquad\qquad\qquad\ \ \langle \widehat{\tau}_2 ; \varphi_2 \rangle \quad\ = \mathcal{R}\ \Gamma\ \Delta\ t_2 \\
\qquad\qquad\qquad\qquad\qquad\ \ \langle \widehat{\tau}_3 ; \varphi_3 \rangle \quad\ = \mathcal{R}\ \Gamma\ \Delta\ t_3
\end{array}
$$

$$
\begin{array}{lll}
 & & \textbf{in } \langle \llbracket \widehat{\tau}_2 \sqcup \widehat{\tau}_3 \rrbracket_\Delta ; \llbracket \varphi_1 \cup \varphi_2 \cup \varphi_3 \rrbracket_\Delta \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ []_\tau & = & \langle [\bot_\tau \langle \emptyset \rangle] ; \emptyset \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ (t_1 :: t_2) & = & \textbf{let } \langle \widehat{\tau}_1 ; \varphi_1 \rangle \qquad\quad = \mathcal{R}\ \Gamma\ \Delta\ t_1 \\
 & & \qquad \langle [\widehat{\tau}_2 \langle \varphi_2' \rangle] ; \varphi_2 \rangle = \mathcal{R}\ \Gamma\ \Delta\ t_2 \\
 & & \textbf{in } \langle \llbracket [(\widehat{\tau}_1 \sqcup \widehat{\tau}_2) \langle \varphi_1 \cup \varphi_2' \rangle] \rrbracket_\Delta ; \varphi_2 \rangle \\
\multicolumn{3}{l}{\mathcal{R}\ \Gamma\ \Delta\ (\textbf{case } t_1 \textbf{ of } \{ [] \mapsto t_2 ; x_1 :: x_2 \mapsto t_3 \})} \\
 & = & \textbf{let } \langle [\widehat{\tau}_1 \langle \varphi_1' \rangle] ; \varphi_1 \rangle \qquad = \mathcal{R}\ \Gamma\ \Delta\ t_1 \\
 & & \qquad \langle \widehat{\tau}_2 ; \varphi_2 \rangle = \mathcal{R}\ (\Gamma, x_1 : \widehat{\tau}_1\ \&\ \varphi_1', x_2 : [\widehat{\tau}_1 \langle \varphi_1' \rangle]\ \&\ \varphi_1)\ \Delta\ t_2 \\
 & & \qquad \langle \widehat{\tau}_3 ; \varphi_3 \rangle = \mathcal{R}\ \Gamma\ \Delta\ t_3 \\
 & & \textbf{in } \langle \llbracket \widehat{\tau}_2 \sqcup \widehat{\tau}_3 \rrbracket_\Delta ; \llbracket \varphi_1 \cup \varphi_2 \cup \varphi_3 \rrbracket_\Delta \rangle
\end{array}
$$

- In R-App and R-Fix: check that the fresh variables generated by $\mathcal{I}$ are subsituted away by the substitution θ created by $\mathcal{M}$. Also, we don't need those variables in the algorithm if we don't generate the elaborated term.
- In R-Fix we could get rid of the auxillary underlying type function if the fixpoint construct was replaced with a binding variant with an explicit type annotation.
- For R-Fix, make sure the way we handle fixpoints of exceptional value in a manner that is sound w.r.t. to the operational semantics we are going to give to this.
- Note that we do not construct the elaborated term, as it is not useful other than for metatheoretic purposes.
- Lemma: The algorithm maintains the invariant that exception types and exceptions are in normal form.

4.6 Subtyping

- Is S-Refl an admissable/derivable rule, or should we drop S-Bool and S-Int?

$$\frac{}{\Delta \vdash \widehat{\tau} \leqslant \widehat{\tau}} \text{ [S-Refl]} \quad \frac{\Delta \vdash \widehat{\tau}_1 \leqslant \widehat{\tau}_2 \quad \Delta \vdash \widehat{\tau}_2 \leqslant \widehat{\tau}_3}{\Delta \vdash \widehat{\tau}_1 \leqslant \widehat{\tau}_3} \text{ [S-Trans]}$$

$$\frac{}{\Delta \vdash \widehat{\mathbf{bool}} \leqslant \widehat{\mathbf{bool}}} \text{ [S-Bool]} \quad \frac{}{\Delta \vdash \widehat{\mathbf{int}} \leqslant \widehat{\mathbf{int}}} \text{ [S-Int]}$$

$$\frac{\Delta \vdash \widehat{\tau}_1' \leqslant \widehat{\tau}_1 \quad \Delta \vdash \varphi_1' \leqslant \varphi_1 \quad \Delta \vdash \widehat{\tau}_2 \leqslant \widehat{\tau}_2' \quad \Delta \vdash \varphi_2 \leqslant \varphi_2'}{\Delta \vdash \widehat{\tau}_1\langle \varphi_1 \rangle \to \widehat{\tau}_2\langle \varphi_2 \rangle \leqslant \widehat{\tau}_1'\langle \varphi_1' \rangle \to \widehat{\tau}_2'\langle \varphi_2' \rangle} \text{ [S-Arr]}$$

$$\frac{\Delta \vdash \widehat{\tau} \leqslant \widehat{\tau}' \quad \Delta \vdash \varphi \leqslant \varphi'}{\Delta \vdash [\widehat{\tau}\langle \varphi \rangle] \leqslant [\widehat{\tau}'\langle \varphi' \rangle]} \text{ [S-List]} \quad \frac{\Delta, e : \kappa \vdash \widehat{\tau}_1 \leqslant \widehat{\tau}_2}{\Delta \vdash \forall e : \kappa . \widehat{\tau}_1 \leqslant \forall e : \kappa . \widehat{\tau}_2} \text{ [S-Forall]}$$

– Possibly useful lemma: $\widehat{\tau}_1 = \widehat{\tau}_2 \iff \widehat{\tau}_1 \leqslant \widehat{\tau}_2 \land \widehat{\tau}_2 \leqslant \widehat{\tau}_1$.

5 Operational semantics

5.1 Evaluation

– The reduction relation is non-deterministic.

– We do not have a Haskell-style imprecise exception semantics (e.g. E-If).

– We either need to omit the type annotations on $\lightning_\tau^\ell$, or add them to **if then else** and **case of** $\{[] \mapsto; :: \mapsto\}$.

– We do not have a rule E-AnnAppExn. Check that the canonical forms lemma gives us that terms of universally quantified type cannot be exceptional values.

6 Interesting observations

– Exception types are not invariant under η-reduction.

7 Metatheory

7.1 Declarative type system

Lemma 1 (Canonical forms)**.**

1. *If $\widehat{v}$ is a possibly exceptional value of type $\widehat{\textbf{bool}}$, then $\widehat{v}$ is either* **true**, **false**, *or $\lightning^{\ell}$.*
2. *If $\widehat{v}$ is a possibly exceptional value of type $\widehat{\textbf{int}}$, then $\widehat{v}$ is either some integer n, or an exceptional value $\lightning^{\ell}$.*
3. *If $\widehat{v}$ is a possibly exceptional value of type $[\widehat{\tau}\langle\varphi\rangle]$, then $\widehat{v}$ is either $[]$, $t :: t'$, or $\lightning^{\ell}$.*
4. *If $\widehat{v}$ is a possibly exceptional value of type $\widehat{\tau}_1\langle\varphi_1\rangle \to \widehat{\tau}_2\langle\varphi_2\rangle$, then $\widehat{v}$ is either $\lambda x : \widehat{\tau}_1 \ \&\ \varphi_1.t'$ or $\lightning^{\ell}$.*
5. *If $\widehat{v}$ is a possibly exceptional value of type $\forall e : \kappa.\widehat{\tau}$, then $\widehat{v}$ is $\Lambda e : \kappa.t$*

Proof. For each part, inspect all forms of $\widehat{v}$ and discard the unwanted cases by inversion of the typing relation. Note that $\bot_\tau$ cannot give us a type of the form $\forall e : \kappa.\widehat{\tau}$. □

To do.: Say something about T-Sub?

Theorem 1 (Progress)**.** *If $\Gamma; \Delta \vdash t : \widehat{\tau} \ \&\ \varphi$ with t a closed term, then t is either a possibly exceptional value $\widehat{v}$ or there is a closed term t' such that $t \longrightarrow t'$.*

Proof. By induction on the typing derivation $\Gamma; \Delta \vdash t : \widehat{\tau} \ \&\ \varphi$.

The case T-Var can be discarded, as a variable is not a closed term. The cases T-Con, T-Crash, T-Abs, T-AnnAbs, T-Nil and T-Cons are immediate as they are values.

Case T-App: We can immediately apply the induction hypothesis to $\Gamma; \Delta \vdash t_1 : \widehat{\tau}_2\langle\varphi_2\rangle \to \widehat{\tau}\langle\varphi\rangle \ \&\ \varphi$, giving us either a t_1' such that $t_1 \longrightarrow t_1'$ or that $t_1 = \widehat{v}$. In the former case we can make progress using E-App. In the latter case the canonical forms lemma tells us that either $t_1 = \lambda x : \widehat{\tau}_2 \ \&\ \varphi_2.t_1'$ or $t_1 = \lightning^{\ell}$, in which case we can make progress using E-AppAbs or E-AppExn, respectively.

The remaining cases follow by analogous reasoning. □

Lemma 2 (Annotation substitution)**.**

1. *If* $\Delta, e : \kappa' \vdash \varphi : \kappa$ *and* $\Delta \vdash \varphi' : \kappa'$ *then* $\Delta \vdash \varphi[\varphi'/e] : \kappa$.
2. *If* $\Delta, e : \kappa' \vdash \varphi_1 \leqslant \varphi_2$ *and* $\Delta \vdash \varphi' : \kappa'$ *then* $\Delta \vdash \varphi_1[\varphi'/e] \leqslant \varphi_2[\varphi'/e]$.
3. *If* $\Delta, e : \kappa' \vdash \widehat{\tau}_1 \leqslant \widehat{\tau}_2$ *and* $\Delta \vdash \varphi' : \kappa'$ *then* $\Delta \vdash \widehat{\tau}_1[\varphi'/e] \leqslant \widehat{\tau}_2[\varphi'/e]$.
4. *If* $\Gamma; \Delta, e : \kappa' \vdash t : \widehat{\tau} \,\&\, \varphi$ *and* $\Delta \vdash \varphi' : \kappa'$ *then* $\Gamma; \Delta \vdash t[\varphi'/e] : \widehat{\tau}[\varphi'/e] \,\&\, \varphi$.

To do.: In part 4, either we need the assumption $e \notin \mathrm{fv}(\varphi)$ (which seems to be satisfied everywhere we want to apply this lemma), or we also need to apply the substitution to φ (is this expected or not in a type-and-effect system)? T-Fix seems to be to only rule where an exception variable can flow from $\widehat{\tau}$ to φ....

Proof. 1. By induction on the derivation of $\Delta, e : \kappa' \vdash \varphi : \kappa$. The cases A-Var, A-Abs and A-App are analogous to the respective cases in the proof of term substitution below. In the case A-Con one can strengthen the assumption $\Delta, e : \kappa' \vdash \{\ell\} : \textsc{exn}$ to $\Delta \vdash \{\ell\} : \textsc{exn}$ as $e \notin \mathrm{fv}(\{\ell\})$, the result is then immediate; similarly for A-Empty. The case A-Union goes analogous to A-App.

2. To do.

3. To do.

4. By induction on the derivation of $\Gamma; \Delta, e : \kappa' \vdash t : \widehat{\tau} \,\&\, \varphi$. Most cases can be discarded by a straightforward application of the induction hypothesis; we show only the interesting case.

Case T-AnnApp: To do.

To do.

□

Lemma 3 (Term substitution)**.** *If* $\Gamma, x : \widehat{\tau}' \,\&\, \varphi'; \Delta \vdash t : \widehat{\tau} \,\&\, \varphi$ *and* $\Gamma; \Delta \vdash t' : \widehat{\tau}' \,\&\, \varphi'$ *then* $\Gamma; \Delta \vdash t[t'/x] : \widehat{\tau} \,\&\, \varphi$.

Proof. By induction on the derivation of $\Gamma, x : \widehat{\tau}' \,\&\, \varphi; \Delta \vdash t : \widehat{\tau} \,\&\, \varphi$.

Case T-Var: We either have $t = x$ or $t = x'$ with $x \neq x'$. In the first case we need to show that $\Gamma; \Delta \vdash x[t'/x] : \widehat{\tau} \,\&\, \varphi$, which by definition of substitution is equal to $\Gamma; \Delta \vdash x : \widehat{\tau} \,\&\, \varphi$, but this is one of our assumptions. In the second case we need to show that $\Gamma, x' : \widehat{\tau} \,\&\, \varphi; \Delta \vdash x'[t/x] : \widehat{\tau} \,\&\, \varphi$, which by definition of substitution is equal to $\Gamma, x' : \widehat{\tau} \,\&\, \varphi; \Delta \vdash x' : \widehat{\tau} \,\&\, \varphi$. This follows immediately from T-Var.

Case T-ABS: Our assumptions are

$$\Gamma, x : \widehat{\tau}' \ \& \ \varphi', y : \widehat{\tau}_1 \ \& \ \varphi_1; \Delta \vdash t : \widehat{\tau}_2 \ \& \ \varphi_2 \tag{7}$$

$$\Gamma; \Delta \vdash t' : \widehat{\tau}' \ \& \ \varphi'. \tag{8}$$

By the Barendregt convention we may assume that $y \neq x$ and $y \notin \mathrm{fv}(t')$. We need to show that $\Gamma; \Delta \vdash (\lambda y : \widehat{\tau}_1 \ \& \ \varphi_1.t)[t'/x] : \widehat{\tau}_1\langle\varphi_1\rangle \to \widehat{\tau}_2\langle\varphi_2\rangle \ \& \ \emptyset$, which by definition of substitution is equal to

$$\Gamma; \Delta \vdash \lambda y : \widehat{\tau}_1 \ \& \ \varphi_1.t[t'/x] : \widehat{\tau}_1\langle\varphi_1\rangle \to \widehat{\tau}_2\langle\varphi_2\rangle \ \& \ \emptyset. \tag{9}$$

We weaken (8) to $\Gamma, y : \widehat{\tau}_1 \ \& \ \varphi_1; \Delta \vdash t' : \widehat{\tau}' \ \& \ \varphi'$ and apply the induction hypothesis on this and (7) to obtain

$$\Gamma, y : \widehat{\tau}_1 \ \& \ \varphi_1; \Delta \vdash t[t'/x] : \widehat{\tau}_2 \ \& \ \varphi_2. \tag{10}$$

The desired result (9) can be constructed from (10) using T-ABS.

Case T-ANNABS: Our assumptions are $\Gamma, x : \widehat{\tau}' \ \& \ \varphi'; \Delta, e : \kappa \vdash t : \widehat{\tau} \ \& \ \varphi$ and $\Gamma; \Delta \vdash t' : \widehat{\tau}' \ \& \ \varphi'$. By the Barendregt convention we may assume that $e \notin \mathrm{fv}(t')$. We need to show that $\Gamma; \Delta \vdash (\Lambda e : \kappa.t)\,[t'/x] : \widehat{\tau} \ \& \ \varphi$, which is equal to $\Gamma; \Delta \vdash \Lambda e : \kappa.t[t'/\kappa] : \widehat{\tau} \ \& \ \varphi$ by definition of substitution. By applying the induction hypothesis we obtain $\Gamma; \Delta, e : \kappa \vdash t[t'/x] : \widehat{\tau} \ \& \ \varphi$. The desired result can be constructed using T-ANNABS.

Case T-APP: Our assumptions are

$$\Gamma, x : \widehat{\tau}' \ \& \ \varphi'; \Delta \vdash t_1 : \widehat{\tau}_2\langle\varphi_2\rangle \to \widehat{\tau}\langle\varphi\rangle \ \& \ \varphi \tag{11}$$

$$\Gamma, x : \widehat{\tau}' \ \& \ \varphi'; \Delta \vdash t_2 : \widehat{\tau}_2 \ \& \ \varphi_2. \tag{12}$$

We need to show that $\Gamma; \Delta \vdash (t_1 \ t_2)[t'/x] : \widehat{\tau} \ \& \ \varphi$, which by definition of substitution is equal to

$$\Gamma; \Delta \vdash \big(t_1[t'/x]\big) \ \big(t_2[t'/x]\big) : \widehat{\tau} \ \& \ \varphi. \tag{13}$$

By applying the induction hypothesis to (11) respectively (12) we obtain

$$\Gamma; \Delta \vdash t_1[t'/x] : \widehat{\tau}_2\langle\varphi_2\rangle \to \widehat{\tau}\langle\varphi\rangle \ \& \ \varphi \tag{14}$$

$$\Gamma; \Delta \vdash t_2[t'/x] : \widehat{\tau}_2 \ \& \ \varphi_2. \tag{15}$$

The desired result (13) can be constructed by applying T-APP to (14) and (15).

All other cases are either immediate or analogous to the case of T-APP. □

Lemma 4 (Inversion)**.**

1. *If* $\Gamma; \Delta \vdash \lambda x : \widehat{\tau} \mathbin{\&} \varphi.t : \widehat{\tau}_1\langle\varphi_1\rangle \to \widehat{\tau}_2\langle\varphi_2\rangle \mathbin{\&} \varphi_3$*, then*
   - $\Gamma, x : \widehat{\tau} \mathbin{\&} \varphi; \Delta \vdash t : \widehat{\tau}' \mathbin{\&} \varphi'$,
   - $\Delta \vdash \widehat{\tau}_1 \leqslant \widehat{\tau}$ *and* $\Delta \vdash \varphi_1 \leqslant \varphi$,
   - $\Delta \vdash \widehat{\tau}' \leqslant \widehat{\tau}_2$ *and* $\Delta \vdash \varphi' \leqslant \varphi_2$.
2. *If* $\Gamma; \Delta \vdash \Lambda e : \kappa.t : \forall e : \kappa.\widehat{\tau} \mathbin{\&} \varphi$*, then*
   - $\Gamma; \Delta, e : \kappa \vdash t : \widehat{\tau}' \mathbin{\&} \varphi'$,
   - $\Delta, e : \kappa \vdash \widehat{\tau}' \leqslant \widehat{\tau}$,
   - $\Delta \vdash \varphi' \leqslant \varphi$.
   - To do. $e \notin \mathrm{fv}(\varphi)$ *and/or* $e \notin \mathrm{fv}(\varphi')$.

Proof. 1. By induction on the typing derivation.

Case T-Abs: We have $\widehat{\tau} = \widehat{\tau}_1$, $\varphi = \varphi_1$ and take $\widehat{\tau}' = \widehat{\tau}_2$, $\varphi' = \varphi_2$, the result then follows immediately from the assumption $\Gamma, x : \widehat{\tau} \mathbin{\&} \varphi; \Delta \vdash t : \widehat{\tau}_2 \mathbin{\&} \varphi_2$ and reflexivity of the subtyping and subeffecting relations.

Case T-Sub: We are given the additional assumptions

$$\Gamma; \Delta \vdash \lambda x : \widehat{\tau} \mathbin{\&} \varphi.t : \widehat{\tau}_1'\langle\varphi_1'\rangle \to \widehat{\tau}_2'\langle\varphi_2'\rangle \mathbin{\&} \varphi_3', \tag{16}$$
$$\Delta \vdash \widehat{\tau}_1'\langle\varphi_1'\rangle \to \widehat{\tau}_2'\langle\varphi_2'\rangle \leqslant \widehat{\tau}_1\langle\varphi_1\rangle \to \widehat{\tau}_2\langle\varphi_2\rangle, \tag{17}$$
$$\Delta \vdash \varphi_3' \leqslant \varphi_3. \tag{18}$$

Applying the induction hypothesis to (16) gives us

$$\Gamma, x : \widehat{\tau} \mathbin{\&} \varphi; \Delta \vdash t : \widehat{\tau}_2'' \mathbin{\&} \varphi_2'', \tag{19}$$
$$\Delta \vdash \widehat{\tau}_1' \leqslant \widehat{\tau}, \quad \Delta \vdash \varphi_1' \leqslant \varphi, \tag{20}$$
$$\Delta \vdash \widehat{\tau}_2'' \leqslant \widehat{\tau}_2', \quad \Delta \vdash \varphi_2'' \leqslant \varphi_2'. \tag{21}$$

Inversion of the subtyping relation on (17) gives us

$$\Delta \vdash \widehat{\tau}_1' \leqslant \widehat{\tau}, \quad \Delta \vdash \varphi_1' \leqslant \varphi, \tag{22}$$
$$\Delta \vdash \widehat{\tau}_2'' \leqslant \widehat{\tau}_2', \quad \Delta \vdash \varphi_2'' \leqslant \varphi_2'. \tag{23}$$

The result follows from (19) and combining (22) with (20) and (21) with (23) using the transitivity of the subtyping and subeffecting relations.

2. By induction on the typing derivation.

Case T-AnnAbs: We need to show that $\Gamma; \Delta, e : \kappa \vdash t : \widehat{\tau} \ \& \ \varphi$, which is one of our assumptions, and that $\Delta, e : \kappa \vdash \widehat{\tau} \leqslant \widehat{\tau}$ and $\Delta \vdash \varphi \leqslant \varphi$; this follows from the reflexivity of the subtyping, respectively subeffecting, relation (noting that $e \notin \mathrm{fv}(\varphi)$).

Case T-Sub: Similar to the case T-Sub in part 1.

∎

Theorem 2 (Preservation). *If $\Gamma; \Delta \vdash t : \widehat{\tau} \ \& \ \varphi$ and $t \longrightarrow t'$, then $\Gamma; \Delta \vdash t' : \widehat{\tau} \ \& \ \varphi$.*

Proof. By induction on the typing derivation $\Gamma; \Delta \vdash t : \widehat{\tau} \ \& \ \varphi$.

The cases for T-Var, T-Con, T-Crash, T-Abs, T-AnnAbs, T-Nil, and T-Cons can be discarded immediately, as they have no applicable evaluation rules.

To do. ∎

7.2 Syntax-directed type elaboration

7.3 Type inference algorithm

Theorem 3 (Syntactic soundness). *If $\mathcal{R}\ \Gamma\ \Delta\ t = \langle \widehat{\tau}; \varphi \rangle$, then $\Gamma; \Delta \vdash t : \widehat{\tau} \ \& \ \varphi$.*

Proof. By induction on the term t.

To do. ∎

Theorem 4 (Termination). *$\mathcal{R}\ \Gamma\ \Delta\ t$ terminates.*

Proof. By induction on the term t.

To do. ∎

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ [\text{E-App}] \qquad \frac{}{(\lambda x : \widehat{\tau}\ \&\ \varphi.t)\ t_2 \longrightarrow t_1[t_2/x]}\ [\text{E-AppAbs}]$$

$$\frac{t \longrightarrow t'}{t\ \langle\varphi\rangle \longrightarrow t'\ \langle\varphi\rangle}\ [\text{E-AnnApp}] \qquad \frac{}{(\Lambda e : \kappa.t)\ \langle\varphi\rangle \longrightarrow t[\varphi/e]}\ [\text{E-AnnAbsAbs}]$$

$$\frac{t \longrightarrow t'}{\textbf{fix}\ t \longrightarrow \textbf{fix}\ t'}\ [\text{E-Fix}] \qquad \frac{}{\textbf{fix}\ (\lambda x : \widehat{\tau}\ \&\ \varphi.t) \longrightarrow t[\textbf{fix}\ (\lambda x : \widehat{\tau}\ \&\ \varphi.t)/x]}\ [\text{E-FixAbs}]$$

$$\frac{}{\lightning^{\ell}\ t_2 \longrightarrow \lightning^{\ell}}\ [\text{E-AppExn}] \qquad \frac{}{\textbf{fix}\ \lightning^{\ell} \longrightarrow \lightning^{\ell}}\ [\text{E-FixExn}]$$

$$\frac{t_1 \longrightarrow t_1'}{t_1 \oplus t_2 \longrightarrow t_1' \oplus t_2}\ [\text{E-Op}_1] \qquad \frac{t_2 \longrightarrow t_2'}{t_1 \oplus t_2 \longrightarrow t_1 \oplus t_2'}\ [\text{E-Op}_2]$$

$$\frac{}{v_1 \oplus v_2 \longrightarrow [\![v_1 \oplus v_2]\!]}\ [\text{E-Op}]$$

$$\frac{}{\lightning^{\ell} \oplus t_2 \longrightarrow \lightning^{\ell}}\ [\text{E-OpExn}_1] \qquad \frac{}{t_1 \oplus \lightning^{\ell} \longrightarrow \lightning^{\ell}}\ [\text{E-OpExn}_2]$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ \textbf{seq}\ t_2 \longrightarrow t_1'\ \textbf{seq}\ t_2}\ [\text{E-Seq}_1] \qquad \frac{}{v_1\ \textbf{seq}\ t_2 \longrightarrow t_2}\ [\text{E-Seq}_2]$$

$$\frac{}{\lightning^{\ell}\ \textbf{seq}\ t_2 \longrightarrow \lightning^{\ell}}\ [\text{E-SeqExn}]$$

$$\frac{t_1 \longrightarrow t_1'}{\textbf{if}\ t_1\ \textbf{then}\ t_2\ \textbf{else}\ t_3 \longrightarrow \textbf{if}\ t_1'\ \textbf{then}\ t_2\ \textbf{else}\ t_3}\ [\text{E-If}]$$

$$\frac{}{\textbf{if}\ \textbf{true}\ \textbf{then}\ t_2\ \textbf{else}\ t_3 \longrightarrow t_2}\ [\text{E-IfTrue}]$$

$$\frac{}{\textbf{if}\ \textbf{false}\ \textbf{then}\ t_2\ \textbf{else}\ t_3 \longrightarrow t_3}\ [\text{E-IfFalse}]$$

$$\frac{}{\textbf{if}\ \lightning^{\ell}\ \textbf{then}\ t_2\ \textbf{else}\ t_3 \longrightarrow \lightning^{\ell}}\ [\text{E-IfExn}]$$

$$\frac{t_1 \longrightarrow t_1'}{\textbf{case}\ t_1\ \textbf{of}\ \{[] \mapsto t_2; x_1 :: x_2 \mapsto t_3\} \longrightarrow \textbf{case}\ t_1'\ \textbf{of}\ \{[] \mapsto t_2; x_1 :: x_2 \mapsto t_3\}}\ [\text{E-Case}]$$

$$\frac{}{\textbf{case}\ []\ \textbf{of}\ \{[] \mapsto t_2; x_1 :: x_2 \mapsto t_3\} \longrightarrow t_2}\ [\text{E-CaseNil}]$$

$$\frac{}{\textbf{case}\ t_1 :: t_1'\ \textbf{of}\ \{[] \mapsto t_2; x_1 :: x_2 \mapsto t_3\} \longrightarrow t_3[t_1; t_1'/x_1; x_2]}\ [\text{E-CaseNil}]$$

$$\frac{}{\textbf{case}\ \lightning^{\ell}\ \textbf{of}\ \{[] \mapsto t_2; x_1 :: x_2 \mapsto t_3\} \longrightarrow \lightning^{\ell}}\ [\text{E-CaseExn}]$$

Figure 2: Operational semantics ($t_1 \longrightarrow t_2$)

$$
\begin{aligned}
e[\varphi/e] &\equiv \varphi \\
e'[\varphi/e] &\equiv e' && \text{if } e \neq e' \\
\{\ell\}[\varphi/e] &\equiv \{\ell\} \\
\emptyset[\varphi/e] &\equiv \emptyset \\
\left(\lambda e' : \kappa.\varphi'\right)[\varphi/e] &\equiv \lambda e' : \kappa.\varphi'[\varphi/e] && \text{if } e \neq e' \text{ and } e' \notin \mathrm{fv}(\varphi) \\
(e_1\ e_2)[\varphi/e] &\equiv (e_1[\varphi/e])\ \ (e_2[\varphi/e]) \\
(e_1 \cup e_2)[\varphi/e] &\equiv e_1[\varphi/e] \cup e_2[\varphi/e]
\end{aligned}
$$

Figure 3: Annotation substitution

$$
\begin{aligned}
x[t/x] &\equiv t \\
x'[t/x] &\equiv x' && \text{if } x \neq x' \\
c_\tau[t/x] &\equiv c_\tau \\
\left(\lambda x' : \widehat{\tau}.t'\right)[t/x] &\equiv \lambda x' : \widehat{\tau}.t'[t/x] && \text{if } x \neq x' \text{ and } x' \notin \mathrm{fv}(t) \\
&\dots
\end{aligned}
$$

Figure 4: Term substitution