Higher-ranked Exception Types

Ruud Koot

May 26, 2015

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

Types

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

Terms

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

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

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

1.1 Typing relation

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

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

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

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

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.*

2 Completion

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

The completion procedure as a set of inference rules:

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

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

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

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

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 $\langle \widehat{\mathrm{bool}}; e\ \overline{e_i}; e :: \overline{\kappa_i \Rightarrow}\ \textsc{exn} \rangle$

3 Type system

3.1 Terms

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

3.2 Underlying type system

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 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

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

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

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

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

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

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

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

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

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.

3.4 Type elaboration system

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

3.5 Type inference algorithm

- 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.

3.6 Subtyping

- Is S-Refl an admissable/derivable rule, or should we drop S-Bool and S-Int?
- Possibly useful lemma: $\widehat{\tau}_1 = \widehat{\tau}_2 \iff \widehat{\tau}_1 \leqslant \widehat{\tau}_2 \wedge \widehat{\tau}_2 \leqslant \widehat{\tau}_1$.

4 Operational semantics

4.1 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 $\lightning^{\ell}_{\tau}$, 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.

5 Interesting observations

- Exception types are not invariant under η-reduction.

6 Metatheory

6.1 Declarative type system

Lemma 1 (Canonical forms)**.**

1. *If $\widehat{v}$ is a possibly exceptional value of type $\widehat{\mathbf{bool}}$, then $\widehat{v}$ is either* **true**, **false**, *or $\lightning^{\ell}$.*
2. *If $\widehat{v}$ is a possibly exceptional value of type $\widehat{\mathbf{int}}$, 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 $\Gamma; \Delta \vdash t : \widehat{\tau} \mathbin{\&} \varphi$.

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 $\Gamma; \Delta \vdash t_1 : \widehat{\tau}_2\langle\varphi_2\rangle \to \widehat{\tau}\langle\varphi\rangle \mathbin{\&} \varphi$, 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 $t_1 = \lambda x : \widehat{\tau}_2 \mathbin{\&} \varphi_2 . t_1'$ 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* $\Gamma; \Delta, e : \kappa' \vdash t : \widehat{\tau} \mathbin{\&} \varphi$ *and* $\Delta \vdash \varphi' : \kappa'$ *then* $\Gamma; \Delta \vdash t[\varphi'/e] : \widehat{\tau} \mathbin{\&} \varphi$.

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 $\Gamma; \Delta, e : \kappa' \vdash t : \widehat{\tau} \mathbin{\&} \varphi$. 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* $\Gamma, x : \widehat{\tau}' \mathbin{\&} \varphi; \Delta \vdash t : \widehat{\tau} \mathbin{\&} \varphi$ *and* $\Gamma; \Delta \vdash t' : \widehat{\tau}' \mathbin{\&} \varphi'$ *then* $\Gamma; \Delta \vdash t[t'/x] : \widehat{\tau} \mathbin{\&} \varphi$.

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

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

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

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* $\Gamma; \Delta \vdash \lambda x : \widehat{\tau} \ \&\ \varphi.t : \widehat{\tau}_1\langle\varphi_1\rangle \to \widehat{\tau}_2\langle\varphi_2\rangle \ \&\ \varphi_3$*, then*
   - $\Gamma, x : \widehat{\tau} \ \&\ \varphi; \Delta \vdash t : \widehat{\tau}_2' \ \&\ \varphi_2'$,
   - $\vdash \leqslant$
2. To do.

Proof. 1. By induction on the derivation of To do.

2. By induction on the derivation of To do. □

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. □

6.2 Syntax-directed type elaboration

6.3 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. □

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$
\begin{array}{ll}
\mathcal{R} : \mathbf{TyEnv} \times \mathbf{KiEnv} \times \mathbf{Tm} \to \mathbf{ExnTy} \times \mathbf{Exn} & \\
\mathcal{R}\ \Gamma\ \Delta\ x & = \Gamma_x \\
\mathcal{R}\ \Gamma\ \Delta\ c_\tau & = \langle \bot_\tau ; \emptyset \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ \lightning^{\ell}_{\tau} & = \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\ \ \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 \\
 & \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) & = \mathbf{let}\ \langle \widehat{\tau}_1 ; \varphi_1 \rangle = \mathcal{R}\ \Gamma\ \Delta\ t_1 \\
 & \qquad\ \ \langle \widehat{\tau}_2 ; \varphi_2 \rangle = \mathcal{R}\ \Gamma\ \Delta\ t_2 \\
 & \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\ \ \theta = [e'_2 \mapsto \varphi_2] \circ \mathcal{M}\ \emptyset\ \widehat{\tau}_2\ \widehat{\tau}'_2 \\
 & \quad\ \mathbf{in}\ \ \langle \llbracket \theta \widehat{\tau}' \rrbracket_\Delta ; \llbracket \theta \varphi' \cup \varphi_1 \rrbracket_\Delta \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ (\mathbf{fix}\ t) & = \mathbf{let}\ \langle \widehat{\tau} ; \varphi \rangle = \mathcal{R}\ \Gamma\ \Delta\ t \\
 & \qquad\ \ \langle \widehat{\tau}' \langle e' \rangle \to \widehat{\tau}'' \langle \varphi'' \rangle ; \overline{e_i : \kappa_i} \rangle = \mathcal{I}\ \widehat{\tau} \\
 & \quad\ \mathbf{in}\ \ \langle \widehat{\tau}_0 ; \varphi_0 ; i \rangle \leftarrow \langle \bot_{\lfloor \widehat{\tau}' \rfloor} ; \emptyset ; 0 \rangle \\
 & \qquad\ \ \mathbf{do}\ \theta \leftarrow [e' \mapsto \varphi_i] \circ \mathcal{M}\ \emptyset\ \widehat{\tau}_i\ \widehat{\tau}' \\
 & \qquad\qquad \langle \widehat{\tau}_{i+1} ; \varphi_{i+1} ; i \rangle \leftarrow \langle \llbracket \theta \widehat{\tau}'' \rrbracket_\Delta ; \llbracket \theta \varphi'' \rrbracket_\Delta ; i + 1 \rangle \\
 & \qquad\ \ \mathbf{until}\ \langle \widehat{\tau}_i ; \varphi_i \rangle \equiv \langle \widehat{\tau}_{i-1} ; \varphi_{i-1} \rangle \\
 & \qquad\ \ \mathbf{return}\ \langle \widehat{\tau}_i ; \llbracket \varphi \cup \varphi_i \rrbracket_\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\ \ \langle \widehat{\mathbf{int}} ; \varphi_2 \rangle = \mathcal{R}\ \Gamma\ \Delta\ t_2 \\
 & \quad\ \mathbf{in}\ \ \langle \widehat{\mathbf{bool}} ; \llbracket \varphi_1 \cup \varphi_2 \rrbracket_\Delta \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ (t_1\ \mathbf{seq}\ t_2) & \\
 & = \mathbf{let}\ \langle \widehat{\tau}_1 ; \varphi_1 \rangle = \mathcal{R}\ \Gamma\ \Delta\ t_1 \\
 & \qquad\ \ \langle \widehat{\tau}_2 ; \varphi_2 \rangle = \mathcal{R}\ \Gamma\ \Delta\ t_2 \\
 & \quad\ \mathbf{in}\ \ \langle \widehat{\tau}_2 ; \llbracket \varphi_1 \cup \varphi_2 \rrbracket_\Delta \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ (\mathbf{if}\ t_1\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3) & \\
 & = \mathbf{let}\ \langle \widehat{\mathbf{bool}} ; \varphi_1 \rangle = \mathcal{R}\ \Gamma\ \Delta\ t_1 \\
 & \qquad\ \ \langle \widehat{\tau}_2 ; \varphi_2 \rangle \quad = \mathcal{R}\ \Gamma\ \Delta\ t_2 \\
 & \qquad\ \ \langle \widehat{\tau}_3 ; \varphi_3 \rangle \quad = \mathcal{R}\ \Gamma\ \Delta\ t_3 \\
 & \quad\ \mathbf{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) & = \mathbf{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 \\
 & \quad\ \mathbf{in}\ \ \langle \llbracket [(\widehat{\tau}_1 \sqcup \widehat{\tau}_2) \langle \varphi_1 \cup \varphi'_2 \rangle] \rrbracket_\Delta ; \varphi_2 \rangle \\
\mathcal{R}\ \Gamma\ \Delta\ (\mathbf{case}\ t_1\ \mathbf{of}\ \{ [] \mapsto t_2 ; x_1 :: x_2 \mapsto t_3 \}) & \\
 & = \mathbf{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 \\
 & \quad\ \mathbf{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}
$$

Figure 5: Type inference algorithm

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

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

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

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

Figure 6: Subtyping

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Figure 7: 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 8: Annotation substitution

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

Figure 9: Term substitution