

# Higher-ranked Exception Types

Ruud Koot

May 26, 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

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

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

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

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

**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

$\kappa \in \mathbf{Kind}$	$::=$	$\mathbf{EXN}$	(exception)
		$\kappa_1 \Rightarrow \kappa_2$	(exception operator)
$\varphi \in \mathbf{Exn}$	$::=$	$e$	(exception variables)
		$\lambda e : \kappa. \varphi$	(exception abstraction)
$\hat{\tau} \in \mathbf{ExnTy}$	$::=$	$\forall e :: \kappa. \hat{\tau}$	(exception quantification)
		$\mathbf{bool}$	(boolean type)
		$[\hat{\tau}\langle\varphi\rangle]$	(list type)
		$\hat{\tau}_1\langle\varphi_1\rangle \rightarrow \hat{\tau}_2\langle\varphi_2\rangle$	(function type)

The completion procedure as a set of inference rules:

$$\begin{array}{c}
\frac{}{\overline{e_i} :: \kappa_i \vdash \mathbf{bool} : \mathbf{bool} \ \& \ e \ \overline{e_i} \triangleright e :: \kappa_i \Rightarrow \mathbf{EXN}} [\mathbf{C-Bool}] \\
\\
\frac{\overline{e_i} :: \kappa_i \vdash \tau : \hat{\tau} \ \& \ \varphi \triangleright \overline{e_j} :: \kappa_j}{\overline{e_i} :: \kappa_i \vdash [\tau] : [\hat{\tau}\langle\varphi\rangle] \ \& \ e \ \overline{e_i} \triangleright e :: \kappa_i \Rightarrow \mathbf{EXN}, \overline{e_j} :: \kappa_j} [\mathbf{C-List}] \\
\\
\frac{\vdash \tau_1 : \hat{\tau}_1 \ \& \ \varphi_1 \triangleright \overline{e_j} :: \kappa_j \quad \overline{e_i} :: \kappa_i, \overline{e_j} :: \kappa_j \vdash \tau_2 : \hat{\tau}_2 \ \& \ \varphi_2 \triangleright \overline{e_j} :: \kappa_j}{\overline{e_i} :: \kappa_i \vdash \tau_1 \rightarrow \tau_2 : \forall \overline{e_j} :: \kappa_j. (\hat{\tau}_1\langle\varphi_1\rangle \rightarrow \hat{\tau}_2\langle\varphi_2\rangle) \ \& \ e \ \overline{e_i} \triangleright e :: \kappa_i \Rightarrow \mathbf{EXN}, \overline{e_k} :: \kappa_k} [\mathbf{C-Arr}]
\end{array}$$

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

The completion procedure as an algorithm:

```

 $\mathcal{C} :: \mathbf{Env} \times \mathbf{Ty} \rightarrow \mathbf{ExnTy} \times \mathbf{Exn} \times \mathbf{Env}$ 
 $\mathcal{C} \ \overline{e_i} :: \kappa_i \ \mathbf{bool} =$ 
  let  $e$  be fresh
  in  $\langle \mathbf{bool}; e \ \overline{e_i}; e :: \kappa_i \Rightarrow \mathbf{EXN} \rangle$ 

```

## 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)
		$\not\downarrow_\tau^\ell$	(exception constant)
		$t_1 \mathbf{seq} t_2$	(forcing)
		<b>fix</b> $t$	(anonymous fixpoint)
		$[]_\tau$	(nil constructor)
		$t_1 :: t_2$	(cons constructor)
		<b>case</b> $t_1$ <b>of</b> $\{ [] \mapsto t_2; x_1 :: x_2 \mapsto t_3 \}$	(list eliminator)

### 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  $\Delta$ .
- 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

$$\begin{array}{c}
\frac{}{\Gamma, x : \tau \vdash x : \tau} [\text{T-VAR}] \quad \frac{}{\Gamma \vdash c_\tau : \tau} [\text{T-CON}] \quad \frac{}{\Gamma \vdash \not\downarrow_\tau^\ell : \tau} [\text{T-CRASH}] \\
\\
\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1. t : \tau_1 \rightarrow \tau_2} [\text{T-ABS}] \quad \frac{\Gamma \vdash t_1 : \tau_2 \rightarrow \tau \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash t_1 t_2 : \tau} [\text{T-APP}] \\
\\
\frac{\Gamma \vdash t : \tau \rightarrow \tau}{\Gamma \vdash \mathbf{fix} \, t : \tau} [\text{T-FIX}] \\
\\
\frac{\Gamma \vdash t_1 : \mathbf{int} \quad \Gamma \vdash t_2 : \mathbf{int}}{\Gamma \vdash t_1 \oplus t_2 : \mathbf{bool}} [\text{T-OP}] \quad \frac{\Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash t_1 \mathbf{seq} t_2 : \tau_2} [\text{T-SEQ}] \\
\\
\frac{\Gamma \vdash t_1 : \mathbf{bool} \quad \Gamma \vdash t_2 : \tau \quad \Gamma \vdash t_3 : \tau}{\Gamma \vdash \mathbf{if} \, t_1 \mathbf{then} \, t_2 \mathbf{else} \, t_3 : \tau} [\text{T-IF}] \\
\\
\frac{}{\Gamma \vdash []_\tau : [\tau]} [\text{T-NIL}] \quad \frac{\Gamma \vdash t_1 : \tau \quad \Gamma \vdash t_2 : [\tau]}{\Gamma \vdash t_1 :: t_2 : [\tau]} [\text{T-CONS}] \\
\\
\frac{\Gamma \vdash t_1 : [\tau_1] \quad \Gamma \vdash t_2 : \tau \quad \Gamma, x_1 : \tau_1, x_2 : [\tau_1] \vdash t_3 : \tau}{\Gamma \vdash \mathbf{case} \, t_1 \mathbf{of} \, \{ [] \mapsto t_2; x_1 :: x_2 \mapsto t_3 \} : \tau} [\text{T-CASE}]
\end{array}$$

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

- change  $\mathbf{fix} \, t$  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 substituted away by the substitution  $\theta$  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 auxiliary 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 admissible/derivable rule, or should we drop S-BOOL and S-INT?
- Possibly useful lemma:  $\hat{\tau}_1 = \hat{\tau}_2 \iff \hat{\tau}_1 \leq \hat{\tau}_2 \wedge \hat{\tau}_2 \leq \hat{\tau}_1$ .

## 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  $\downarrow_{\tau}^{\ell}$ , or add them to **if then else** and **case of**  $\{\llbracket \cdot \rrbracket \mapsto; :: \mapsto\}$ .
- We do not have a rule E-ANNAPEXN. 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  $\eta$ -reduction.

## 6 Metatheory

### 6.1 Declarative type system

**To do.** Inversion lemmas?

**Lemma 1** (Canonical forms).

1. If  $\hat{v}$  is a possibly exceptional value of type  $\widehat{\mathbf{bool}}$ , then  $\hat{v}$  is either **true**, **false**, or  $\downarrow_{\tau}^{\ell}$ .
2. If  $\hat{v}$  is a possibly exceptional value of type  $\widehat{\mathbf{int}}$ , then  $\hat{v}$  is either some integer  $n$ , or an exceptional value  $\downarrow_{\tau}^{\ell}$ .
3. If  $\hat{v}$  is a possibly exceptional value of type  $[\widehat{\tau}(\varphi)]$ , then  $\hat{v}$  is either  $\llbracket \cdot \rrbracket$ ,  $t :: t'$ , or  $\downarrow_{\tau}^{\ell}$ .
4. If  $\hat{v}$  is a possibly exceptional value of type  $\widehat{\tau}_1(\varphi_1) \rightarrow \widehat{\tau}_2(\varphi_2)$ , then  $\hat{v}$  is either  $\lambda x : \widehat{\tau}_1 \ \& \ \varphi_1.t'$  or  $\downarrow_{\tau}^{\ell}$ .
5. If  $\hat{v}$  is a possibly exceptional value of type  $\forall e : \kappa.\widehat{\tau}$ , then  $\hat{v}$  is  $\Lambda e : \kappa.t$

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

**To do.:** Say something about T-SUB?



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

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

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  $\Gamma; \Delta \vdash t_1 : \hat{\tau}_2 \langle \varphi_2 \rangle \rightarrow \hat{\tau} \langle \varphi \rangle \& \varphi$ , giving us either a  $t'_1$  such that  $t_1 \longrightarrow t'_1$  or that  $t_1 = \hat{v}$ . In the former case we can make progress using E-APP. In the latter case the canonical forms lemma tells us that either  $t_1 = \lambda x : \hat{\tau}_2 \& \varphi_2. t'_1$  or  $t_1 = \downarrow^\ell$ , in which case we can make progress using E-APPABS or E-APPEXN, respectively.

The remaining cases follow by analogous reasoning.  $\square$

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

*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  $\Delta, e : \kappa' \vdash \{\ell\} : \text{EXN}$  to  $\Delta \vdash \{\ell\} : \text{EXN}$  as  $e \notin \text{fv}(\{\ell\})$ , 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  $\Gamma; \Delta, e : \kappa' \vdash t : \hat{\tau} \& \varphi$ . 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.**

$\square$

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

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

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

Case T-ABS: Our assumptions are

$$\Gamma, x : \hat{\tau}' \& \varphi', y : \hat{\tau}_1 \& \varphi_1; \Delta \vdash t : \hat{\tau}_2 \& \varphi_2 \quad (7)$$

$$\Gamma; \Delta \vdash t' : \hat{\tau}' \& \varphi'. \quad (8)$$

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

$$\Gamma; \Delta \vdash \lambda y : \hat{\tau}_1 \& \varphi_1. t[t'/x] : \hat{\tau}_1 \langle \varphi_1 \rangle \rightarrow \hat{\tau}_2 \langle \varphi_2 \rangle \& \emptyset. \quad (9)$$

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

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

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

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

Case T-APP: Our assumptions are

$$\Gamma, x : \hat{\tau}' \& \varphi'; \Delta \vdash t_1 : \hat{\tau}_2 \langle \varphi_2 \rangle \rightarrow \hat{\tau} \langle \varphi \rangle \& \varphi \quad (11)$$

$$\Gamma, x : \hat{\tau}' \& \varphi'; \Delta \vdash t_2 : \hat{\tau}_2 \& \varphi_2. \quad (12)$$

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

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

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

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

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

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.  $\square$

**Lemma 4** (Inversion).

1. If  $\Gamma; \Delta \vdash \lambda x : \hat{\tau} \ \& \ \varphi. t : \hat{\tau}_1 \langle \varphi_1 \rangle \rightarrow \hat{\tau}_2 \langle \varphi_2 \rangle \ \& \ \varphi_3$ , then

$$- \Gamma, x : \hat{\tau} \ \& \ \varphi; \Delta \vdash t : \hat{\tau}'_2 \ \& \ \varphi'_2,$$

$$- \vdash \leq$$

2. **To do.**

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

2. By induction on the derivation of **To do**.  $\square$

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

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

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.**  $\square$

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

*Proof.* By induction on the term  $t$ .

**To do.**  $\square$

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

*Proof.* By induction on the term  $t$ .

**To do.**  $\square$

## **6.2 Syntax-directed type elaboration**

## **6.3 Type inference algorithm**

$$\begin{array}{c}
\frac{}{\Gamma, x : \widehat{\tau} \& \varphi; \Delta \vdash x : \widehat{\tau} \& \varphi} \text{[T-VAR]} \\
\\
\frac{}{\Gamma; \Delta \vdash c_\tau : \perp_\tau \& \emptyset} \text{[T-CON]} \quad \frac{}{\Gamma; \Delta \vdash \not\downarrow_\tau^\ell : \perp_\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 \rightarrow \widehat{\tau}_2 \langle \varphi_2 \rangle \& \emptyset} \text{[T-ABS]} \\
\\
\frac{\Gamma; \Delta, e : \kappa \vdash t : \widehat{\tau} \& \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 \rightarrow \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 \rightarrow \widehat{\tau} \langle \varphi' \rangle \& \varphi'' \quad \Delta \vdash \varphi' \leq \varphi \quad \Delta \vdash \varphi'' \leq \varphi}{\Gamma; \Delta \vdash \mathbf{fix} \, t : \widehat{\tau} \& \varphi} \text{[T-FIX]} \\
\\
\frac{\Gamma; \Delta \vdash t_1 : \mathbf{\hat{int}} \& \varphi \quad \Gamma; \Delta \vdash t_2 : \mathbf{\hat{int}} \& \varphi}{\Gamma; \Delta \vdash t_1 \oplus t_2 : \mathbf{\widehat{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 : \mathbf{\widehat{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 : [\perp_\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{\Gamma; \Delta \vdash t_1 : [\widehat{\tau}_1 \langle \varphi_1 \rangle] \& \varphi' \quad \Delta \vdash \varphi' \leq \varphi \quad \Gamma; \Delta \vdash t_2 : \widehat{\tau} \& \varphi \quad \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}{\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}' \leq \widehat{\tau} \quad \Delta \vdash \varphi' \leq \varphi}{\Gamma; \Delta \vdash t : \widehat{\tau}_1 \& \varphi} \text{[T-SUB]}
\end{array}$$

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

$$\begin{array}{c}
\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 \& \emptyset} \text{[T-CON]} \quad \frac{}{\Gamma; \Delta \vdash \not\hookrightarrow_\tau^\ell \hookrightarrow \not\hookrightarrow_\tau^\ell : \perp_\tau \& \{\ell\}} \text{[T-CRASH]} \\
\\
\frac{\Delta, \overline{e_i} : \overline{\kappa_i} \vdash \widehat{\tau}_1 \triangleright \tau_1 \quad \Delta, \overline{e_i} : \overline{\kappa_i} \vdash \varphi_1 : \text{EXN} \quad \Gamma, x : \widehat{\tau}_1 \& \varphi_1; \Delta, \overline{e_i} : \overline{\kappa_i} \vdash t \hookrightarrow t' : \widehat{\tau}_2 \& \varphi_2}{\Gamma; \Delta \vdash \lambda x : \tau_1. t \hookrightarrow \Lambda \overline{e_i} : \overline{\kappa_i}. \lambda x : \widehat{\tau}_1 \& \varphi_1. t' : \forall \overline{e_i} : \overline{\kappa_i}. \widehat{\tau}_1 \langle \varphi_1 \rangle \rightarrow \widehat{\tau}_2 \langle \varphi_2 \rangle \& \emptyset} \text{[T-ABS]} \\
\\
\frac{\Delta \vdash \widehat{\tau}_2 \leq \widehat{\tau}[\overline{\varphi_i}/\overline{e_i}] \quad \Delta \vdash \varphi_2 \leq \varphi[\overline{\varphi_i}/\overline{e_i}] \quad \Delta \vdash \varphi_i : \overline{\kappa_i} \quad \Gamma; \Delta \vdash t_1 \hookrightarrow t'_1 : \forall \overline{e_i} : \overline{\kappa_i}. \widehat{\tau}_1 \langle \varphi_1 \rangle \rightarrow \widehat{\tau} \langle \varphi \rangle \& \varphi' \quad \Gamma; \Delta \vdash t_2 \hookrightarrow t'_2 : \widehat{\tau}_2 \& \varphi_2}{\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{\Gamma; \Delta \vdash t \hookrightarrow t' : \forall \overline{e_i} : \overline{\kappa_i}. \widehat{\tau} \langle \varphi \rangle \rightarrow \widehat{\tau}' \langle \varphi' \rangle \& \varphi'' \quad \Delta \vdash \widehat{\tau}'[\overline{\varphi_i}/\overline{e_i}] \leq \widehat{\tau}[\overline{\varphi_i}/\overline{e_i}] \quad \Delta \vdash \varphi'[\overline{\varphi_i}/\overline{e_i}] \leq \varphi[\overline{\varphi_i}/\overline{e_i}] \quad \Delta \vdash \varphi_i : \overline{\kappa_i}}{\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 : \mathbf{int} \& \varphi_1 \quad \Gamma; \Delta \vdash t_2 \hookrightarrow t'_2 : \mathbf{int} \& \varphi_2}{\Gamma; \Delta \vdash t_1 \oplus t_2 \hookrightarrow t'_1 \oplus t'_2 : \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 : \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 \square_\tau \hookrightarrow \square_\tau : \perp_\tau \& \emptyset} \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 : [\widehat{\tau}_1 \langle \varphi'_1 \rangle] \& \varphi_2}{\Gamma; \Delta \vdash t_1 :: t_2 \hookrightarrow 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; \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 \quad \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}{\Gamma; \Delta \vdash \mathbf{case} \, t_1 \mathbf{of} \{ \square \mapsto t_2; x_1 :: x_2 \mapsto t_3 \} \hookrightarrow \mathbf{case} \, t'_1 \mathbf{of} \{ \square \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 ( $\Gamma; \Delta \vdash t \hookrightarrow t' : \widehat{\tau} \& \varphi$ )

$$\begin{aligned}
\mathcal{R} : \mathbf{TyEnv} \times \mathbf{KiEnv} \times \mathbf{Tm} &\rightarrow \mathbf{ExnTy} \times \mathbf{Exn} \\
\mathcal{R} \Gamma \Delta x &= \Gamma_x \\
\mathcal{R} \Gamma \Delta c_\tau &= \langle \perp_\tau; \emptyset \rangle \\
\mathcal{R} \Gamma \Delta \frac{\ell}{\tau} &= \langle \perp_\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} \oslash \tau \\
&\quad \langle \widehat{\tau}_2; \varphi_2 \rangle = \mathcal{R} (\Gamma, x : \widehat{\tau}_1 \ \& \ e_1) (\Delta, \overline{e_i : \kappa_i}) t \\
&\quad \mathbf{in} \langle \overline{\forall e_i : \kappa_i. \widehat{\tau}_1 \langle e_1 \rangle} \rightarrow \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 \\
&\quad \langle \widehat{\tau}_2; \varphi_2 \rangle = \mathcal{R} \Gamma \Delta t_2 \\
&\quad \langle \widehat{\tau}'_2 \langle e'_2 \rangle \rightarrow \widehat{\tau}' \langle \varphi' \rangle; \overline{e_i : \kappa_i} \rangle = \mathcal{I} \widehat{\tau}_1 \\
&\quad \theta = [e'_2 \mapsto \varphi_2] \circ \mathcal{M} \oslash \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 \\
&\quad \langle \widehat{\tau}' \langle e' \rangle \rightarrow \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 \perp_{[\widehat{\tau}']} \rangle; \emptyset; 0 \rangle \\
&\quad \mathbf{do} \ \theta \leftarrow [e' \mapsto \varphi_i] \circ \mathcal{M} \oslash \widehat{\tau}_i \ \widehat{\tau}' \\
&\quad \quad \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 \\
&\quad \mathbf{until} \ \langle \widehat{\tau}_i; \varphi_i \rangle \equiv \langle \widehat{\tau}_{i-1}; \varphi_{i-1} \rangle \\
&\quad \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 \\
&\quad \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 \\
&\quad \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 \\
&\quad \langle \widehat{\tau}_2; \varphi_2 \rangle = \mathcal{R} \Gamma \Delta t_2 \\
&\quad \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 \\
\mathcal{R} \Gamma \Delta \llbracket \perp_\tau \rrbracket &= \langle \llbracket \perp_\tau \langle \emptyset \rangle \rrbracket; \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 \\
&\quad \langle \llbracket \widehat{\tau}_2 \langle \varphi'_2 \rangle \rrbracket; \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} \ \{ \llbracket \cdot \rrbracket \mapsto t_2; x_1 :: \mathbf{x}\overline{2} \mapsto t_3 \}) &= \mathbf{let} \langle \llbracket \widehat{\tau}_1 \langle \varphi'_1 \rangle \rrbracket; \varphi_1 \rangle = \mathcal{R} \Gamma \Delta t_1 \\
&\quad \langle \widehat{\tau}_2; \varphi_2 \rangle = \mathcal{R} (\Gamma, x_1 : \widehat{\tau}_1 \ \& \ \varphi'_1, x_2 : \llbracket \widehat{\tau}_1 \langle \varphi'_1 \rangle \rrbracket \ \& \ \varphi_1) \Delta t_2 \\
&\quad \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{aligned}$$

Figure 5: Type inference algorithm

$$\begin{array}{c}
\frac{}{\Delta \vdash \widehat{\tau} \leq \widehat{\tau}} \text{[S-REFL]} \quad \frac{\Delta \vdash \widehat{\tau}_1 \leq \widehat{\tau}_2 \quad \Delta \vdash \widehat{\tau}_2 \leq \widehat{\tau}_3}{\Delta \vdash \widehat{\tau}_1 \leq \widehat{\tau}_3} \text{[S-TRANS]} \\
\\
\frac{}{\Delta \vdash \widehat{\mathbf{bool}} \leq \widehat{\mathbf{bool}}} \text{[S-BOOL]} \quad \frac{}{\Delta \vdash \widehat{\mathbf{int}} \leq \widehat{\mathbf{int}}} \text{[S-INT]} \\
\\
\frac{\Delta \vdash \widehat{\tau}'_1 \leq \widehat{\tau}_1 \quad \Delta \vdash \varphi'_1 \leq \varphi_1 \quad \Delta \vdash \widehat{\tau}_2 \leq \widehat{\tau}'_2 \quad \Delta \vdash \varphi_2 \leq \varphi'_2}{\Delta \vdash \widehat{\tau}_1 \langle \varphi_1 \rangle \rightarrow \widehat{\tau}_2 \langle \varphi_2 \rangle \leq \widehat{\tau}'_1 \langle \varphi'_1 \rangle \rightarrow \widehat{\tau}'_2 \langle \varphi'_2 \rangle} \text{[S-ARR]} \\
\\
\frac{\Delta \vdash \widehat{\tau} \leq \widehat{\tau}' \quad \Delta \vdash \varphi \leq \varphi'}{\Delta \vdash [\widehat{\tau} \langle \varphi \rangle] \leq [\widehat{\tau}' \langle \varphi' \rangle]} \text{[S-LIST]} \quad \frac{\Delta, e : \kappa \vdash \widehat{\tau}_1 \leq \widehat{\tau}_2}{\Delta \vdash \forall e : \kappa. \widehat{\tau}_1 \leq \forall e : \kappa. \widehat{\tau}_2} \text{[S-FORALL]}
\end{array}$$

Figure 6: Subtyping



$$\begin{array}{c}
\frac{t_1 \longrightarrow t'_1}{t_1 \ t_2 \longrightarrow t'_1 \ t_2} [\text{E-APP}] \quad \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}] \quad \frac{}{(\Lambda e : \kappa.t) \ \langle \varphi \rangle \longrightarrow t[\varphi/e]} [\text{E-ANNAbsAbs}] \\
\\
\frac{t \longrightarrow t'}{\mathbf{fix} \ t \longrightarrow \mathbf{fix} \ t'} [\text{E-FIX}] \quad \frac{}{\mathbf{fix} \ (\lambda x : \widehat{\tau} \ \& \ \varphi.t) \longrightarrow t[\mathbf{fix} \ (\lambda x : \widehat{\tau} \ \& \ \varphi.t)/x]} [\text{E-FIXAbs}] \\
\\
\frac{}{\downarrow^\ell t_2 \longrightarrow \downarrow^\ell} [\text{E-APPEXN}] \quad \frac{}{\mathbf{fix} \ \downarrow^\ell \longrightarrow \downarrow^\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] \quad \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 \llbracket v_1 \oplus v_2 \rrbracket} [\text{E-OP}] \\
\\
\frac{}{\downarrow^\ell \oplus t_2 \longrightarrow \downarrow^\ell} [\text{E-OPEXN}_1] \quad \frac{}{t_1 \oplus \downarrow^\ell \longrightarrow \downarrow^\ell} [\text{E-OPEXN}_2] \\
\\
\frac{t_1 \longrightarrow t'_1}{t_1 \ \mathbf{seq} \ t_2 \longrightarrow t'_1 \ \mathbf{seq} \ t_2} [\text{E-SEQ}_1] \quad \frac{}{v_1 \ \mathbf{seq} \ t_2 \longrightarrow t_2} [\text{E-SEQ}_2] \\
\\
\frac{}{\downarrow^\ell \ \mathbf{seq} \ t_2 \longrightarrow \downarrow^\ell} [\text{E-SEQEXN}] \\
\\
\frac{t_1 \longrightarrow t'_1}{\mathbf{if} \ t_1 \ \mathbf{then} \ t_2 \ \mathbf{else} \ t_3 \longrightarrow \mathbf{if} \ t'_1 \ \mathbf{then} \ t_2 \ \mathbf{else} \ t_3} [\text{E-IF}] \\
\\
\frac{}{\mathbf{if} \ \mathbf{true} \ \mathbf{then} \ t_2 \ \mathbf{else} \ t_3 \longrightarrow t_2} [\text{E-IFTRUE}] \\
\\
\frac{}{\mathbf{if} \ \mathbf{false} \ \mathbf{then} \ t_2 \ \mathbf{else} \ t_3 \longrightarrow t_3} [\text{E-IFFALSE}] \\
\\
\frac{}{\mathbf{if} \ \downarrow^\ell \ \mathbf{then} \ t_2 \ \mathbf{else} \ t_3 \longrightarrow \downarrow^\ell} [\text{E-IFEXN}] \\
\\
\frac{t_1 \longrightarrow t'_1}{\mathbf{case} \ t_1 \ \mathbf{of} \ \{\ [] \mapsto t_2; x_1 :: x_2 \mapsto t_3 \} \longrightarrow \mathbf{case} \ t'_1 \ \mathbf{of} \ \{\ [] \mapsto t_2; x_1 :: x_2 \mapsto t_3 \}} [\text{E-CASE}] \\
\\
\frac{}{\mathbf{case} \ [] \ \mathbf{of} \ \{\ [] \mapsto t_2; x_1 :: x_2 \mapsto t_3 \} \longrightarrow t_2} [\text{E-CASENIL}] \\
17 \\
\frac{}{\mathbf{case} \ t_1 :: t'_1 \ \mathbf{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{}{\mathbf{case} \ \downarrow^\ell \ \mathbf{of} \ \{\ [] \mapsto t_2; x_1 :: x_2 \mapsto t_3 \} \longrightarrow \downarrow^\ell} [\text{E-CASEEXN}]
\end{array}$$

Figure 7: Operational semantics ( $t_1 \longrightarrow t_2$ )

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

Figure 8: Annotation substitution

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

Figure 9: Term substitution