Abstract. This technical report presents the full proofs for the type preservation and effect soundness theorems of the type system presented in the article “Lightweight Polymorphic Effects”.

1 Introduction

We first repeat the lemmas and theorems introduced in the paper [2].

Lemma 1. Monotonicity.
For every effect domain $D$ and every syntactic form $Trm$,
1. if $\forall e_i \in e, d_i \in d . e_i \sqsubseteq d_i$, then $\text{eff}_D(Trm, e) \sqsubseteq \text{eff}_D(Trm, d)$
2. $\text{eff}_D(Trm, e_1, \ldots, e_i \sqcup e_{i+1}, \ldots, e_n) \sqsubseteq \text{eff}_D(Trm, e_1, \ldots, e_{i+1}, \ldots, e_n) \sqcup e_i$

Lemma 2. Consistency.
– Let $S = \text{dynEff}(Trm, S)$ be the set of dynamic effects that occur when evaluating a term $t$ of the form $Trm$. The list $S$ contains an effect set for every subterm of $t$.
– Let $\Gamma; f$ be an environment and $\bar{e}$ be a list of static effects such that every effect set in $S$ is approximated by $S_i \leq e_i \sqcup \text{latent}(\Gamma(f))$.
– Then the static effect $e = \text{eff}(Trm, \bar{e})$ is a conservative approximation of the effects in $S$, i.e., $S \leq e \sqcup \text{latent}(\Gamma(f))$.

As noted in the paper, this lemma has to be verified for the $\text{dynEff}_D$ and $\text{eff}_D$ functions of every effect domain $D$. We show that it holds for the domain of exceptions $E$, and we then show that it holds for the general functions $\text{dynEff}$ and $\text{eff}$ which act on all effect domains.

Lemma 3. Preservation under substitution for monomorphic abstractions.
If $\Gamma, x : T_1; f \vdash t : T ! e_1, f \neq x$ and $\Gamma; g \vdash v : T_2 ! \bot$ with $T_2 <: T_1$, then $\Gamma; f \vdash t[v\backslash x] : T' ! e'_1$ such that $T' <: T$ and $e'_1 \sqsubseteq e_1$.

Lemma 4. Preservation under substitution for polymorphic abstractions.
If $\Gamma, x : T_1; x \vdash t : T ! e_1$ and $\Gamma; g \vdash v : T_2 ! \bot$ with $T_2 <: T_1$, then $\Gamma; e \vdash t[v\backslash x] : T' ! e'_1$ such that $T' <: T$ and $e'_1 \sqsubseteq e_1 \sqcup \text{latent}(T_2)$.

The substitution lemmas are used in the proofs of both the preservation and the soundness theorem.
Theorem 1. Preservation.
If \( \Gamma ; f \vdash t : T ! e \) is a valid typing statement for term \( t \) and the term evaluates as \( t \Downarrow \langle r, S \rangle \), then there is a valid typing statement \( \Gamma ; f \vdash r : T' ! e' \) for \( r \) with \( T' < : T \).

Theorem 2. Effect soundness.
If \( \Gamma ; f \vdash t : T ! e \) and \( t \Downarrow \langle r, S \rangle \), then \( S \preceq e \sqcup \text{latent}(\Gamma(f)) \).

2 Consistency of \( \text{eff} \) and \( \text{dynEff} \), monotonicity of \( \text{eff} \)

2.1 Consistency of \( \text{eff} \) and \( \text{dynEff} \)

Proof (Lemma 2 for the domain of exceptions \( \mathcal{E} \)). By case-analysis on the syntactic terms \( \text{Trm} \):

For every effect \( e_S \in S \), there are two cases:
- case \( \text{THROW}(p) \):
  \( S = \text{dynEff}_E(\text{THROW}(p)) = \text{throws}(p) \)
  \( e = \text{eff}_E(\text{THROW}(p)) = \text{throws}(p) \)
  therefore \( S \preceq e \).
- case \( \text{CATCH} \):
  \( S = \text{dynEff}_E(\text{CATCH}, S_1, S_2) = (S_1 \setminus \{\text{throws}(p_i) \mid p_i \in \mathcal{P}\}) \cup S_2 \)
  \( e = \text{eff}_E(\text{CATCH}, e_1, e_2) = \text{throws}(Q \setminus \mathcal{P} \cup E), \) where \( e_1 = \text{throws}(Q), e_2 = \text{throws}(E) \).
  Given \( \forall e_S, e_S \sqsubseteq \text{throws}(Q \setminus \mathcal{P} \cup E) \lor e_S \sqsubseteq \text{throws}(E) \lor e_S \sqsubseteq \text{latent}(\Gamma(f)) \), we conclude \( e_S \sqsubseteq e_1 \sqcup e_2 \sqcup \text{latent}(\Gamma(f)) \).

Remaining cases (including \( \text{TRY} \)):
  \( S = \text{dynEff}_E(\text{Trm}, S) = \bigcup S_i \)
  \( e = \text{eff}_E(\text{Trm}, \pi) = \bigcup e_i \)
  \( \forall e_{S_i} \in S_i, e_{S_i} \sqsubseteq e_i \sqcup \text{latent}(\Gamma(f)) \) by hypothesis of the lemma. By the lattice properties of \( \sqcup \) and \( \sqsubseteq \), we conclude \( \forall e_S \in S, e_S \sqsubseteq e \sqcup \text{latent}(\Gamma(f)) \).

Proof (Lemma 2 for the multi-domain functions \( \text{dynEff} \) and \( \text{eff} \)). Immediate by using the effect-domain specific versions of Lemma 2 element-wise on the effects in \( S \) and \( e \).

2.2 Monotonicity of \( \text{eff} \)

Proof (Lemma 3, Part 1., for the domain of exceptions \( \mathcal{E} \)). By case-analysis on the syntactic terms \( \text{Trm} \):
– case CATCH:
\[ \text{\text{eff}_{\mathcal{E}}}(\text{CATCH}(\overline{p}), e_1, e_2) = \text{throws}((\overline{q_1} \setminus \overline{p}) \cup \overline{s_c}), \]
where \( e_1 = \text{throws}(\overline{q_1}) \) and \( e_2 = \text{throws}(\overline{s_c}) \)
\[ \text{\text{eff}_{\mathcal{E}}}(\text{CATCH}(\overline{p}), d_1, d_2) = \text{throws}((\overline{q_4} \setminus \overline{p}) \cup \overline{s_d}), \]
where \( d_1 = \text{throws}(\overline{q_4}) \) and \( d_2 = \text{throws}(\overline{s_d}) \).

Since \( e_1 \subseteq d_1, e_2 \subseteq d_2 \), we have \( \overline{q_1} \subseteq \overline{q_4} \) and \( \overline{s_c} \subseteq \overline{s_d} \). By the properties of set operations, \( ((\overline{q_1} \setminus \overline{p}) \cup \overline{s_c}) \subseteq ((\overline{q_4} \setminus \overline{p}) \cup \overline{s_d}) \).
Therefore, \( \text{\text{eff}_{\mathcal{E}}}(\text{CATCH}(\overline{p}), e_1, e_2) \subseteq \text{\text{eff}_{\mathcal{E}}}(\text{CATCH}(\overline{p}), d_1, d_2) \).

– Remaining cases (including THROW(p), TRY):

We have \( \text{\text{eff}_{\mathcal{E}}}(\text{TRM}, \overline{\mathcal{E}}) = \bigcup e_i \) and \( \text{\text{eff}_{\mathcal{E}}}(\text{TRM}, \overline{d}) = \bigcup d_i \). Since \( e_i \subseteq d_i \), the conclusion is immediate.

\[ \square \]

Proof (Lemma 4, Part 2., for the domain of exceptions \( \mathcal{E} \)). By case-analysis on the syntactic terms TRM:

– case CATCH where \( e_1 = e_{11} \sqcup e_{12} \):

\[ \text{\text{eff}_{\mathcal{E}}}(\text{CATCH}(\overline{p}), e_{11} \sqcup e_{12}, e_2) = \text{throws}((\overline{q_{11}} \cup \overline{q_{22}}) \setminus \overline{p}) \cup \overline{s_c} \]
where \( e_{11} = \text{throws}(\overline{q_{11}}) \), \( e_{12} = \text{throws}(\overline{q_{22}}) \) and \( e_2 = \text{throws}(\overline{s_c}) \)
\[ \text{\text{eff}_{\mathcal{E}}}(\text{CATCH}(\overline{p}), e_{11}, e_{12}, d_2) = \text{throws}((\overline{q_{11}} \setminus \overline{p}) \cup \overline{s_c} \cup \overline{q_{22}}) \cup \text{throws}(\overline{q_{22}}). \]
The conclusion is straightforward.

– case CATCH where \( e_2 = e_{21} \sqcup e_{22} \): Similar as above.

– Remaining cases (including THROW(p), TRY):

We have \( \text{\text{eff}_{\mathcal{E}}}(\text{TRM}, \overline{\mathcal{E}}) = \bigcup e_i \). The proof is straightforward by commutativity of \( \sqcup \).

\[ \square \]

3 Canonical forms and value typing environment

We now introduce two additional lemmas that are used in the preservation and soundness proofs.

**Lemma 5.** Canonical forms.

1. If \( \Gamma; f \vdash v : T ! \perp \) and \( T <: T_1 \rightarrow_{\mathcal{E}} T_2 \), then \( v = (x : T'_1) \Rightarrow_{\mathcal{E}} t \).
2. If \( \Gamma; f \vdash v : T ! \perp \) and \( T <: T_1 \rightarrow_{\mathcal{E}} T_2 \), then \( v = (x : T'_1) \rightarrow_{\mathcal{E}} t \).

*Proof (Lemma 3).* There are only two kinds of values in the language: monomorphic and effect-polymorphic function abstractions. In the first case, \( v \) has a monomorphic function type. By inspecting the subtyping rules, it cannot be a subtype of a polymorphic function type, which validates the first case. Similar for the second case.

\[ \square \]

**Lemma 6.** Environment for type-checking values.

1. If \( \Gamma; f \vdash v : T ! \perp \), then \( \Gamma; f' \vdash v : T ! \perp \) for an arbitrary \( f' \).
2. If \( \Gamma; f \vdash x : T ! \perp \) for a parameter \( x \in \Gamma \), then \( \Gamma; f' \vdash x : T ! \perp \) for an arbitrary \( f' \).
3. If $\Gamma; f \vdash \text{throw}(p) : \text{Nothing}! e$, then $\Gamma; f' \vdash \text{throw}(p) : \text{Nothing}! e$ for an arbitrary $f'$.

This lemma states that the polymorphism environment $f$ does not have an impact on type-checking a value, a parameter or an error term.

Proof (Lemma 3). There are two kinds of values: monomorphic and polymorphic function abstractions. Inspecting the corresponding typing rules $T\text{-Abs-Mono}$ and $T\text{-Abs-Poly}$, one can see that both rules do not make use of the effect-polymorphism environment $f$ in any way.

For parameters and error terms, the conclusion also follows immediately from the corresponding typing rules.

4 Substitution Lemmas

We now proof the substitution Lemmas 3 and 4.

As a first step however, we need to introduce a new typing rule for our type system. The reason, as explained in TAPL [1] chapter 16.4, is that the type system has a bottom type ($\text{Nothing}$), but no subsumption rule. The typing rules presented in the main paper are therefore incomplete, they do not cover the case of an application expression when the function is an error.

\[
\frac{\Gamma; f \vdash t_1 : \text{Nothing}! e_1 \\
\Gamma; f \vdash t_2 : T_2! e_2}{\Gamma; f \vdash t_1 t_2 : \text{Nothing}! \text{eff}(\text{App}, e_1, e_2, \bot)} \quad (T\text{-App-E})
\]

4.1 Lemma 3

Proof (Lemma 3). The preconditions of the lemma are:

- $\Gamma, x : T_1; f \vdash t : T! e_l$ with $f \neq x$
- $\Gamma; g \vdash v : T_2! \bot$
- $T_2 <: T_1$

Proof of $\Gamma; f \vdash t[v\backslash x] : T'! e'_l$ with $T' <: T$ and $e'_l \sqsubseteq e_l$ by induction on the typing rules for term $t$.

- Case T-PARAM: we have $t = z$ and $z : T \in \Gamma, x : T_1$. There are two sub-cases:
  - $z = x$, then $z[v\backslash x] = v$ and $T_1 = T$. Since we have $\Gamma; g \vdash v : T_2! \bot$, by Lemma 6 we have $\Gamma; f \vdash v : T_2! \bot$. The required results are immediate.
  - $z \neq x$, then $z[v\backslash x] = z$. We have $\Gamma, x : T_1; f \vdash z : T! e_l$. Since we know that $x$ is not a free variable, $x \notin FV(z)$, we can remove its binding from the typing environment and obtain the result.
Case T-Abs-Mono: \( t = (y : T_a) \Rightarrow t_1 \). From the typing rule, we get \( \Gamma, x : T_1, y : T_a; e \vdash t_1 : T_b ! e \) and \( T = T_a \Rightarrow T_b \). We can assume that \( x \neq y \) and \( y \notin \text{FV}(v) \).

By permutation on the environment: \( \Gamma, y : T_a, x : T_1; e \vdash t_1 : T_b ! e \)

By weakening: \( \Gamma, y : T_a; e \vdash v : T_2 ! \perp \)

Induction hypothesis: \( \Gamma, y : T_a; e \vdash t_1[v/x] : T'_b ! e' \) with \( T'_b \ll T_b \) and \( e' \sqsubseteq e \)

We have \( t[v/x] = (y : T_a) \Rightarrow t_1[v/x] \). By applying T-Abs-MONO, we obtain \( \Gamma; f \vdash t[v/x] : T_a \Rightarrow T'_b \perp \perp \) and verify \( T_a \Rightarrow T'_b \ll T \) and \( \perp \ll e_1 \).

Case T-App-Mono: \( t = t_1 t_2 \). From the typing rule, we have:

\[
\begin{align*}
\Gamma, x : T_1; f \vdash t_1 : T_a \Rightarrow T ! e_1 \\
\Gamma, x : T_1; f \vdash t_2 : T_b ! e_2 \\
T_b \ll T_a \text{ and } e_1 = \text{eff(App, e_1, e_2, e)}. \\
\end{align*}
\]

We apply the induction hypothesis to the subterms \( t_1 \) and \( t_2 \) to obtain:

\[
\begin{align*}
\Gamma; f \vdash t_1[v/x] : T'_a ! e'_1 \text{ with } T'_a <: T_a \Rightarrow T \text{ and } e'_1 \ll e_1 \\
\Gamma; f \vdash t_2[v/x] : T'_b ! e'_2 \text{ with } T'_b \ll T_b \text{ and } e'_2 \ll e_2. \\
\end{align*}
\]

Looking at the subtyping rules, the type \( T_a \) can have two possible forms:

- case \( T_a = \text{Nothing} \): Applying T-App-E yields \( \Gamma; f \vdash t[v/x] : \text{Nothing} ! e'_1 \) where \( e'_1 = \text{eff(App, e_1, e_2, e)} \). We have \( \text{Nothing} \ll T \) by S-Nothing.

- case \( T_a = T'_a \Rightarrow T' \): Using S-Trans we obtain \( T_d <: T'_a \Rightarrow T' \). Applying T-App-Mono yields \( \Gamma; f \vdash t[v/x] : T' ! e'_1 \) where \( e'_1 = \text{eff(App, e_1, e_2, e')} \).

The result \( T' <: T \) is immediate.

In both cases we have \( \text{eff(App, e_1, e_2, e') \subseteq eff(App, e_1, e_2, e) \) for the resulting effect by monotonicity of eff.\)

Case T-Abs-Poly: similar to the case T-Abs-Mono.

Case T-App-Param: \( t = f t_2 \). From the typing rule:

\[
\begin{align*}
f : T_a \Rightarrow T \in \Gamma, x : T_1, \text{ and } f : T_a \Rightarrow T \in \Gamma \text{ since } f \neq x \\
\Gamma, x : T_1; f \vdash t_2 : T_b ! e_2, \text{ and } T_b <: T_a \\
e_1 = \text{eff(App, } \perp, e_2, \perp) \]

By induction hypothesis \( \Gamma; f \vdash t_2[v/x] : T'_b ! e'_2 \text{ with } T'_b <: T_b, e'_2 \ll e_2 \).

Since \( f \neq x \), we have \( f[v/x] = f \). Therefore, by applying T-App-Param, we obtain:

\[
\begin{align*}
\Gamma; f \vdash t[v/x] : T ! e'_1 \text{ with } e'_1 = \text{eff(App, } \perp, e'_2, \perp) \text{. By monotonicity of eff, we obtain } e'_1 \ll e_1. \\
\end{align*}
\]

Case T-App-Poly: similar to the case T-Abs-Mono. We need one additional small lemma: if \( T' \ll T \) then latent\( (T') \subseteq \text{latent}(T) \). The proof is straightforward.

Case T-App-E: similar to T-App-Mono. We make use of the observation that if \( T <: \text{Nothing} \), then \( T = \text{Nothing} \) (by analysis of the subtyping rules).

Case T-Try: straightforward.

Case T-Catch: \( t = \text{try } t_1 \text{ catch } p \text{ t}_2 \). From the typing rule we have:

\[
\begin{align*}
\Gamma, x : T_1; f \vdash t_1 : T_a ! e_1 \text{ and } T_a <: T \\
\Gamma, x : T_1; f \vdash t_2 : T_b ! e_1 \text{ and } T_b <: T \\
e_1 = \text{eff(Catch}\( p)\), eff(Try, e_1), e_2) \\
\end{align*}
\]

By applying the induction hypothesis to \( t_1 \) and \( t_2 \):

\[
\begin{align*}
\Gamma; f \vdash t_1[v/x] : T'_a ! e'_1 \text{ with } T'_a <: T_a \text{ and } e'_1 \ll e_1 \\
\end{align*}
\]
4.2 Lemma

Proof (Lemma 4). The preconditions of the lemma are:

- \( \Gamma, x : T_1; x \vdash t : T ! e_l \)
- \( \Gamma; g \vdash v : T_2 ! \perp \)
- \( T_2 < : T_1 \)

Proof of \( \Gamma; e \vdash t[v\langle x \rangle] : T' ! e'_l \) with \( T' < : T \) and \( e'_l \sqsubseteq e_l \sqcup \text{latent}(T_2) \) by induction on the typing rules for term \( t \).

Case T-Param: Similar to the case in Lemma 3

Case T-Abs-Mono: \( t = (y : T_a) \Rightarrow t_1 \). From the typing rule we get:

\( \Gamma, x : T_1, y : T_a ; e \vdash t_1 : T_b ! e \)
\( T = T_a \Rightarrow T_b \).

We can assume that \( x \neq y \) and \( y \notin \text{FV}(v) \).

By permutation on the environment: \( \Gamma, y : T_a, x : T_1 ; e \vdash t_1 : T_b ! e \)

By weakening: \( \Gamma, y : T_a; g \vdash v : T_2 ! \perp \)

Applying Lemma 3 \( \Gamma, y : T_a; e \vdash t_1[v\langle x \rangle] : T_b' ! e' \) with \( T_b' < : T_b \) and \( e' \sqsubseteq e \).

By applying T-Abs-Mono: \( \Gamma; e \vdash t[v\langle x \rangle] : T_a \Rightarrow T_b' ! \perp \).

Verifying \( T_a \Rightarrow T_b' < : T \) and \( \perp \sqsubseteq e_l \) is straightforward.

Case T-App-Mono: \( t = t_1 t_2 \). From the typing rule we have:

\( \Gamma, x : T_1 ; x \vdash t_1 : T_a \Rightarrow T ! e_1 \)
\( \Gamma, x : T_1 ; x \vdash t_2 : T_b ! e_2 \)
\( T_b < : T_a \) and \( e_l = \text{eff}(\text{App}, e_1, e_2, e) \).

We apply the induction hypothesis to the subterms \( t_1 \) and \( t_2 \) to obtain

\( \Gamma; e \vdash t_1[v\langle x \rangle] : T ! e'_1 \) with \( T_a < : T_a \Rightarrow T \) and \( e'_1 \sqsubseteq e_1 \sqcup \text{latent}(T_2) \)
\( \Gamma; e \vdash t_2[v\langle x \rangle] : T_d ! e'_2 \) with \( T_d < : T_b \) and \( e'_2 \sqsubseteq e_2 \sqcup \text{latent}(T_2) \)

Looking at the subtyping rules, the type \( T_d \) can have two possible forms:

- case \( T_d = \text{Nothing} \): Applying T-App-E yields \( \Gamma; e \vdash t[v\langle x \rangle] : \text{Nothing} ! e'_1 \)
  where \( e'_1 = \text{eff}(\text{App}, e'_1, e'_2, \perp) \). We have \( \text{Nothing} < : T \) by S-Nothing.

- case \( T_d = T'_a \Rightarrow T' \): Using S-Trans we obtain \( T_d < : T'_a \Rightarrow T' \). Applying T-App-Mono yields \( \Gamma; e \vdash t[v\langle x \rangle] : T' ! e'_1 \) where \( e'_1 = \text{eff}(\text{App}, e'_1, e'_2, e') \).
  The result \( T' < : T \) is immediate.

We need to verify \( \text{eff}(\text{App}, e'_1, e'_2, e') \sqsubseteq \text{eff}(\text{App}, e_1, e_2, e) \sqcup \text{latent}(T_2) \).

Monotonicity of \( \text{eff} \) gives
\( \text{eff}(\text{App}, e'_1, e'_2, e') \sqsubseteq \text{eff}(\text{App}, e_1 \sqcup \text{latent}(T_2), e_2 \sqcup \text{latent}(T_2), e) \)

Using the second property of the monotonicity Lemma 1 we conclude:
\( \ldots \sqsubseteq \text{eff}(\text{App}, e_1, e_2, e) \sqcup \text{latent}(T_2) \)

Case T-Abs-Poly: similar to the case T-Abs-Mono.
Case T-App-Param: $t = x \ t_2$. From the typing rule:

$x : T_a \Rightarrow T \in \Gamma, \ x : T_1$, therefore $T_1 = T_a \Rightarrow T$

$\Gamma, x : T_1; x \vdash t_2 : T_b \ ! e_2$, and $T_b < : T_a$

$e_i = \text{eff(App, } \bot, e_2, \bot)$

By the induction hypothesis $\Gamma; e \vdash t_2[v \backslash x] : T'_b \ ! e'_2$ with $T'_b < : T_b$ and $e'_2 \subseteq e_2 \sqcup \text{latent}(T_2)$.

We have $t[v \backslash x] = v \ t_2[v \backslash x]$. By Lemma 6 we have $\Gamma; e \vdash v : T_2 ! \bot$. Since $T_2 < : T_1 = T_a \Rightarrow T$, there are two possible shapes for $T_2$:

- case $T_2 = \text{Nothing}$: Applying T-App-E yields $\Gamma; e \vdash t[v \backslash x] : \text{Nothing} ! e'_1$ where $e'_1 = \text{eff(App, } \bot, e_2, \bot)$. We have $\text{Nothing} < : T$ by S-NOTHING.

- case $T_2 = T'_a \Rightarrow T'$: Using S-TRANS we obtain $T_b' < : T'_a$. Applying T-App-Mono yields $\Gamma; e \vdash t[v \backslash x] : T' ! e'_1$ where $e'_1 = \text{eff(App, } \bot, e_2, e')$.

The result $T' < : T$ is immediate.

We need to verify $\text{eff(App, } \bot, e_2, e') \subseteq \text{eff(App, } \bot, e_2, \bot) \sqcup \text{latent}(T_2)$. Monotonicity of \text{eff} gives

$$\text{eff(App, } \bot, e_2, e') \subseteq \text{eff(App, } \bot, e_2 \sqcup \text{latent}(T_2), e')$$

Since $e' \subseteq \bot \sqcup e'$, monotonicity gives

$$\ldots \subseteq \text{eff(App, } \bot, e_2 \sqcup \text{latent}(T_2), \bot \sqcup e')$$

Using the second property of the monotonicity Lemma 1

$$\ldots \subseteq \text{eff(App, } \bot, e_2, \bot) \sqcup e' \sqcup \text{latent}(T_2)$$

And finally, since $e' = \text{latent}(T_2)$, we obtain

$$\ldots \subseteq \text{eff(App, } \bot, e_2, \bot) \sqcup \text{latent}(T_2)$$

Case T-App-Poly: similar to the case T-App-Mono. We again use the property that if $T' < : T$, then $\text{latent}(T') \subseteq \text{latent}(T)$.

Case T-App-E: similar to T-App-Mono. We need to make use of the observation that if $T < : \text{Nothing}$, then $T = \text{Nothing}$ (by analysis of the subtyping rules).

Case T-Throw: straightforward.

Case T-Try: $t = \text{try } t_1 \text{ catch}(\overline{p}) \ t_2$. By the typing rules:

$\Gamma, x : T_1; x \vdash t_1 : T_a \ ! e_1$ and $T_a < : T$

$\Gamma, x : T_1; x \vdash t_2 : T_b \ ! e_1$ and $T_b < : T$

$e_i = \text{eff(Catch(\overline{p}), eff(Try, e_1), e_2)}$

By applying the induction hypothesis to $t_1$ and $t_2$:

$\Gamma; e \vdash t_1[v \backslash x] : T'_a \ ! e'_1$ with $T'_a < : T_a$ and $e'_1 \subseteq e_1 \sqcup \text{latent}(T_2)$

$\Gamma; e \vdash t_2[v \backslash x] : T'_b \ ! e'_2$ with $T'_b < : T_b$ and $e'_2 \subseteq e_2 \sqcup \text{latent}(T_2)$

Applying T-Try, we obtain $\Gamma; e \vdash t[v \backslash x] : T ! e'_1$ with $e'_1 = \text{eff(Catch(\overline{p}), eff(Try, e'_1), e'_2)}$.

By the monotonicity Lemma 1 we obtain $e'_1 \subseteq e_i$.

□
5 Soundness theorems

5.1 Preservation (Theorem \[\ref{th:pres}])

Proof (Theorem \[\ref{th:pres}]). The preconditions are

- \(\Gamma, f \vdash t : T ! e\)
- \(t \downarrow \langle r, S \rangle\)

Proof of \(\Gamma, f \vdash r : T^! : e'\) with \(T^! <: T\) by induction on the evaluation rules for term \(t\).

\[\downarrow\] Case E-App-E1: \(t = t_1 t_2\). We have
\[
t_1 \downarrow \langle \text{throw}(p), S_1 \rangle \quad t_2 \downarrow \langle \text{throw}(p), S \rangle,
\]
where \(S = \text{dynEff}(S_1, \emptyset, \emptyset)\).

By typing rule T-Throw, we obtain \(\Gamma; f \vdash r : \text{Nothing} ! e'\text{eff(\text{THROW}(p))}\).

The result \(\text{Nothing} <: T\) is immediate by S-Nothing.

\[\downarrow\] Case E-App-E2: similar.

\[\downarrow\] Case E-App: \(t = t_1 t_2\). We have:
\[
t_1 \downarrow \langle (x : T_1) \mapsto t_r, S_1 \rangle\]
\[
t_2 \downarrow \langle v_2, S_2 \rangle\]
\[
t_e[v_2 \langle x \rangle] \downarrow \langle r, S \rangle\]

We distinguish sub-cases for the typing rule of the application expression \(t_1 t_2\):

- case T-App-E: \(\Gamma; f \vdash t_1 : \text{Nothing} ! e_1\). By induction hypothesis, we have \(\Gamma; f \vdash (x : T_1) \mapsto t_r : T_1' ! e_1'\) such that \(T_1' <: \text{Nothing}\). Since the type of a function abstraction cannot be \(\text{Nothing}\), this case is impossible.

- case T-App-Param: \(t_1 = p\) for some parameter \(p\). This is impossible, since we have \(t_1 \downarrow \langle (x : T_1) \mapsto t_r, S_1 \rangle\), but there is no evaluation rule for parameters.

- case T-App-Mono: we have
\[
\Gamma; f \vdash t_1 : T_a \equiv T ! e_1\]
\[
\Gamma; f \vdash t_2 : T_b ! e_2\]

Applying the induction hypothesis to \(t_1\) and \(t_2\):
\[
\Gamma; f \vdash (x : T_1) \mapsto t_r : T_c ! e_1'\]
\[
\Gamma; f \vdash v_2 : T_b' ! e_2'\]

According to the subtyping rules, \(T_c\) can either be \(\text{Nothing}\) or a monomorphic function type. But since \(T_c\) is the type of a function abstraction, it cannot be \(\text{Nothing}\); therefore we have \(T_c = T_d \equiv T_e\). By the canonical forms Lemma \[\ref{lem:canform}\] the corresponding function abstraction is a monomorphic one. We obtain:
\[
\Gamma; f \vdash (x : T_1) \mapsto t_r : T_1' \equiv T_e ! e_1',\]
with \(T_d = T_1\)

Therefore:
\[
\Gamma; x : T_1; e \vdash t_r : T_c ! e_1'\]

By transitivity of subtyping, we have \(T_b' <: T_b <: T_a <: T_1\) and we can apply substitution Lemma \[\ref{lem:substitution}\] to obtain
\[
\Gamma; \epsilon \vdash t_r[v_2 \langle x \rangle] : T_c' ! e_1''\]
with \(T_c' <: T_c\).
Now we apply the induction hypothesis on \( t_r[v_2/x] \) to obtain
\[
\Gamma; e \vdash r : T_r ! e_r \text{ with } T_r <: T_r'.
\]
By Lemma 6 we get \( \Gamma; f \vdash r : T_r ! e_r \) and by transitivity of subtyping
\( T_r <: T_r' <: T_s <: T \).
• case T-APP-MONO: similar.

\[\begin{align*}
\text{Case E-Throw: } t &= \text{throw}(p). \text{ By T-Throw, we have } \\
\Gamma; f &\vdash \text{throw}(p) : \text{Nothing} ! \text{eff(Throw(p))}.
\end{align*}\]
The same typing rule is also applied to the result \( r \), and we verify \( \text{Nothing} <: \text{Nothing} \) by S-REFL.

\[\begin{align*}
\text{Case E-Try-E: } t &= \text{try} \ t_1 \ \text{catch}(\overline{p}) \ t_2. \text{ We have } \\
t_1 &\downarrow (\text{throw}(p), S_1) \\
t_2 &\downarrow (r_2, S_2)
\end{align*}\]
From the typing rule T-Try:
\[\begin{align*}
\Gamma; f &\vdash t_1 : T_1 ! e_1 \text{ with } T_1 <: T \\
\Gamma; f &\vdash t_2 : T_2 ! e_2 \text{ with } T_2 <: T
\end{align*}\]
Applying the induction hypothesis to \( t_1 \), we obtain \( \Gamma; f \vdash r_2 : T_2' ! e_2' \) with \( T_2' <: T_2 \). Since \( r = r_2 \), in remains to verify using S-TRANS that \( T_2' <: T_2 <: T \).

\[\begin{align*}
\text{Case E-Try: } \text{similar.}
\end{align*}\]

\[ \square \]

5.2 Effect soundness (Theorem 2)

**Proof (Theorem 2).** The preconditions are
\[\begin{align*}
- \Gamma, f &\vdash t : T ! e \\
- t &\downarrow (r, S)
\end{align*}\]
Proof of \( S \preceq e \sqcup \text{latent}(\Gamma(f)) \) by induction on the evaluation rules for term \( t \).

\[\begin{align*}
\text{Case T-App-E1: } t &= t_1 \ t_2. \text{ We have } \\
t_1 &\downarrow (\text{throw}(p), S_1) \\
t_2 &\downarrow (\text{throw}(p), \text{dynEff(App, S_1, \emptyset, \emptyset)})
\end{align*}\]
We look at the sub-cases corresponding to the typing rules for applications:
• case T-App-E: \( \Gamma; f \vdash t_1 : \text{Nothing} ! e_1 \) and \( \Gamma; f \vdash t_2 : T_2 ! e_2 \) and \( e = \text{eff(App, e_1, e_2, \bot)} \).
By induction hypothesis, we have \( S_1 \preceq e_1 \sqcup \text{latent}(\Gamma(f)) \). Note that trivially, \( \emptyset \preceq e_x \) for any \( e_x \). Therefore we can apply the consistency Lemma 2 to obtain \( \text{dynEff(App, S_1, \emptyset, \emptyset) \preceq eff(App, e_1, e_2, \bot \sqcup \text{latent}(\Gamma(f)))} \).
• case T-App-Param: \( t_1 = p \) for some parameter \( p \). This case is not possible because there is no evaluation rule for parameters.
• case T-App-Mono:
\[\begin{align*}
\Gamma; f &\vdash t_1 : T_1 \Downarrow T ! e_1 \\
\Gamma; f &\vdash t_2 : T_2 ! e_2 \text{ with } T_2 <: T_1 \\
e &= \text{eff(App, e_1, e_2, e_1)}
\end{align*}\]
The induction hypothesis on \( t_1 \) gives \( S_1 \preceq e_1 \sqcup \text{latent}(\Gamma(f)) \).
Since $\emptyset \leq e_x$ for any $e_x$, we can apply Lemma 2 to obtain
\[\text{dynEff}(\text{App}, S_1, \emptyset, \emptyset) \leq \text{eff}(\text{App}, e_1, e_2, e_l) \sqcup \text{latent}(\Gamma(f))\].

- case T-App-Poly: similar.

\[
\begin{align*}
\text{(1)} & : ((x : T_a) \mapsto t_r, S_1) \\
\text{(2)} & : \langle v_2, S_2 \rangle \\
\text{(3)} & : \langle v_2 \setminus x \rangle \mapsto \langle r, S_1 \rangle \\
\end{align*}
\]

Since $\emptyset \leq e_x$ for any $e_x$, we can apply Lemma 2 to obtain
\[\text{dynEff}(\text{App}, S_1, \emptyset, \emptyset) \leq \text{eff}(\text{App}, e_1, e_2, e_l) \sqcup \text{latent}(\Gamma(f))\].

- case E-App-E2: similar.

\[
\begin{align*}
\text{(4)} & : t_1 = t_2. \text{ We have the following preconditions} \\
& : t_1 \Downarrow \langle (x : T_a) \mapsto t_r, S_1 \rangle \\
& : t_2 \Downarrow \langle v_2, S_2 \rangle \\
& : t_r[v_2 \setminus x] \Downarrow \langle r, S_1 \rangle \\
\end{align*}
\]

Therefore, we have Lemma 5, the term is a monomorphic function abstraction:
\[e \sqsubseteq \text{eff}(\text{App}, e_1, e_2, e_l)\]. By the preservation theorem, we have
\[\Gamma; f \vdash (x : T_a) \mapsto t_r : T'_1 ! e'_1\] with \(T'_1 \lessdot \text{Nothing}\), which cannot be derived with any typing rule. Therefore, this case is impossible.

- case T-App-Param: \(t_1 = p\) for some parameter \(p\). This case is not possible because there is no evaluation rule for parameters.

- case T-App-Mono:

\[
\begin{align*}
\Gamma; f \vdash t_1 : T_1 \Downarrow T ! e_1 \\
\Gamma; f \vdash t_2 : T_2 ! e_2 \text{ with } T_2 < : T_1 \\
\text{By induction hypothesis on } t_1 \text{ and } t_2, \text{ we obtain} \\
S_1 \leq e_1 \sqcup \text{latent}(\Gamma(f)) \\
S_2 \leq e_2 \sqcup \text{latent}(\Gamma(f)) \\
\end{align*}
\]

By the preservation theorem, we have \(\Gamma; f \vdash (x : T_a) \mapsto t_r : T'_1 ! \bot\) with \(T'_1 \lessdot : T_1 \Downarrow T\). Since the term is a function abstraction, \(T'_1 = \text{Nothing}\) is not possible, which implies \(T'_1 = T_a \Downarrow T'\). By the canonical forms Lemma 3, the term is a monomorphic function abstraction:
\[\Gamma; f \vdash (x : T_a) \mapsto t_r : T_a \Downarrow T! \bot\]

Therefore, we have \(\Gamma; x : T_a; e \vdash t_r : T' ! e'_1\). Applying preservation on \(t_2\) gives us \(\Gamma; f \vdash v_2 : T_2 ! \bot\) with \(T_2 < : T_2 < : T_1 < : T_a\). We can apply substitution Lemma 2 to obtain
\[\Gamma; e \vdash t_r[v_2 \setminus x] : T'' ! e''_1\] with \(T'' < : T'\) and \(e''_1 \subseteq e'_1\).

The induction hypothesis on \(t[v_2 \setminus x]\) gives \(S_1 \leq e''_1 \sqcup \text{latent}(\Gamma(e))\). Since \(\text{latent}(\Gamma(e)) = \bot\), and by \(e''_1 \subseteq e'_1 \subseteq e_1\), we have \(S_1 \leq e_1\). Together with the induction hypotheses, we apply the consistency Lemma 2 to obtain
\[S \leq \text{eff}(\text{App}, e_1, e_2, e_l) \sqcup \text{latent}(\Gamma(f))\]

- case T-App-Poly: similar (see main paper).

\[
\begin{align*}
\text{(5)} & : t = \text{throw}(p) \text{ and } t \Downarrow \langle \text{throw}(p), \text{dynEff}(\text{THROW}(p)) \rangle \\
\end{align*}
\]

By typing rule T-Throw, we obtain
\[\Gamma; f \vdash \text{throw}(p) : \text{Nothing} \sqsubseteq \text{eff}(\text{THROW}(p))\]. By Lemma 2 we conclude
\[\text{dynEff}(\text{THROW}(p)) \leq \text{eff}(\text{THROW}(p)) \sqcup \text{latent}(\Gamma(f))\].

\[
\begin{align*}
\text{(6)} & : t = \text{try} t_1 \text{ catch}(\overline{p}) t_2. \text{ We have} \\
& : t_1 \Downarrow \langle \text{throw}(p), S_1 \rangle \\
& : t_2 \Downarrow \langle r_2, S_2 \rangle \\
\end{align*}
\]
\[ S_t = \text{dynEff}(\text{Try}, S_1) \quad S = \text{dynEff}(\text{Catch}(\overline{p}), S_1, S_2) \]

From the typing rule T-Try:
\[
\begin{align*}
\Gamma; f &\vdash t_1 : T_1 ! e_1 \text{ with } T_1 <: T \\
\Gamma; f &\vdash t_2 : T_2 ! e_2 \text{ with } T_2 <: T \\
e_t &= \text{eff}(\text{Try}, e_1) \\
e &= \text{eff}(\text{Catch}(\overline{p}), e_t, e_2)
\end{align*}
\]

The induction hypotheses are
\[
\begin{align*}
S_1 &\preceq e_1 \sqcup \text{latent}(\Gamma(f)) \\
S_2 &\preceq e_2 \sqcup \text{latent}(\Gamma(f))
\end{align*}
\]

Applying Lemma 2 to \( S_t \) and \( e_t \) yields \( S_t \preceq e_t \sqcup \text{latent}(\Gamma(f)) \). Now we can apply the same lemma to \( S \) and \( e \) and conclude \( S \preceq e \sqcup \text{latent}(\Gamma(f)) \).

\[ \quad \square \]

References