Refinement Types for TypeScript

– Supplemental Material –

1. Full System
In this section we present the full type system for the core language of § 3 of the main paper.

1.1 Object Constraint System
Our system leverages the idea introduced in the formal core of X10 [3] to extend a base constraint system \( C \) with a larger constraint system \( O(C) \), built on top of \( C \). The original system \( C \) comprises formulas taken from a decidable SMT logic [2], including, for example, linear arithmetic constraints and uninterpreted predicates. The Object Constraint System \( O(C) \) introduces the constraints:

- \( \text{class } (C) \), which it true for all classes \( C \) defined in the program;
- \( \text{x hasImm } F \), to denote that the immutable field \( F \) is accessible from variable \( x \);
- \( \text{x hasMut } G \), to denote that the mutable field \( G \) is accessible from variable \( x \); and
- \( \text{fields } (x) = \diamond F, G \), to expose all fields available to \( x \).

Figure 1 shows the constraint system as ported from CFG [3]. We refer the reader to that work for details. The main differences are syntactic changes to account for our notion of strengthening. Also the Field rule accounts now for both immutable (as in CFJ) and mutable fields.

1.2 Well-formedness Constraints
The well-formedness rules for predicates, terms, types and heaps can be found in Figure 2. The majority of these rules are routine.

The judgment for term well-formedness assigns a sort to each term \( t \), which can be thought of as a base type. The judgment \( \Gamma \vdash_q t \) is used as a shortcut for any further constraints that the \( f \) operator might impose on its arguments \( t \). For example if \( f \) is the equality operator then the two arguments are required to have types that are related via subtyping, \( i.e. \) if \( t_1 : N_1 \) and \( t_2 : N_2 \), it needs to be the case that \( N_1 \leq N_2 \) or \( N_2 \leq N_1 \).

Type well-formedness is typical among similar refinement types [1].

1.3 Subtyping
Figure 3 presents the full set of subtyping rules, which borrows ideas from similar systems [1, 4].

1.4 Operational Semantics
The reduction rules for language IRSC are shown in Figure 4. These rules are re similar to the respective rules found in FCJ [3]. We use evaluation contexts \( E \), with a left to right evaluation order, defined as:

\[
E ::= \langle \rangle \mid E.f \mid E.m\langle \nu, E, \tau \rangle \mid \text{new } C\langle \nu, E, \tau \rangle \mid E \text{ as } T \mid \text{let } x = E \text{ in } u \mid E.f = u \mid v.f = E \mid \text{if } (E) \text{ then } u \text{ else } u
\]
Well-Formed Predicates

- $\Gamma \vdash p$

Well-Formed Terms

- $\Gamma \vdash t : N$

Well-Formed Types

- $\Gamma ; \Sigma \vdash T$

Well-Formed Heaps

- $\Gamma ; \Sigma \vdash H$

Figure 1: Structural Constraints

Figure 2: Typing Rules
Subtyping

<table>
<thead>
<tr>
<th>Rule</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\leq\text{-REFL})</td>
<td>(\Gamma \vdash T \leq T)</td>
</tr>
<tr>
<td>(\leq\text{-TRANS})</td>
<td>(\Gamma \vdash T_1 \leq T_2) (\Gamma \vdash T_2 \leq T_3) (\Gamma \vdash T_1 \leq T_3)</td>
</tr>
<tr>
<td>(\leq\text{-BASE})</td>
<td>(\Gamma \vdash N \leq N') (\Gamma \vdash (\nu : N \mid p) \leq (\nu : N' \mid p'))</td>
</tr>
<tr>
<td>(\leq\text{-WITNESS})</td>
<td>(\Gamma \vdash u : S) (\Gamma \vdash T \leq [u/x] T')</td>
</tr>
<tr>
<td>(\leq\text{-BIND})</td>
<td>(\Gamma, x : S \vdash T \leq T') (x \notin FV(T')) (\Gamma \vdash \exists x : S . T \leq T')</td>
</tr>
</tbody>
</table>

Figure 3: Subtyping Rules

Operational Semantics

<table>
<thead>
<tr>
<th>Rule</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{RC-ECTX})</td>
<td>(H, u \mapsto H', u')</td>
</tr>
<tr>
<td>(\text{RC-FIELD})</td>
<td>(H[l] = \text{new} C (\nu)) (x : C \vdash \text{fields} (x) = \emptyset : T, \varphi)) (H, l.m (\nu) \mapsto H, [\varphi/x, l/\text{this}] u)</td>
</tr>
<tr>
<td>(\text{RC-INVK})</td>
<td>(H[l] = \text{new} C (\nu)) (x : C \vdash \text{has} (\text{def} m (x, T) {p} : T = u)) (H, l.m (\nu) \mapsto H, [\varphi/x, l/\text{this}] u)</td>
</tr>
<tr>
<td>(\text{RC-NEW})</td>
<td>(H' = l \mapsto \text{new} C (\nu)) (H) (l) fresh (\mapsto H', l)</td>
</tr>
<tr>
<td>(\text{RC-LETIN})</td>
<td>(H, \text{let} x = v) (u \mapsto H, [v/x] u)</td>
</tr>
<tr>
<td>(\text{R-ASGN})</td>
<td>(H[l] = \text{new} C (\nu)) (H' = l \mapsto \text{new} C (\nu)) (v \mapsto H', v)</td>
</tr>
</tbody>
</table>

Figure 4: Reduction Rules
2. Proofs

Lemma 1 (Substitution Lemma). If $\Gamma \vdash \overline{w} : \overline{S}, \Gamma; \overline{x} : \overline{S} \vdash \overline{S}'$, and $\Gamma; \overline{x} : \overline{S}' \vdash u : T$, then $\Gamma \vdash [\overline{w}/\overline{x}] u : R, R \leq T$.

Proof. By induction on the derivation of the statement $\Gamma; \overline{x} : \overline{S} \vdash u : T$.

Lemma 2 (Weakening). If $\Gamma \vdash S \leq T$, then $\Gamma, x : R \vdash S \leq T$.

Proof. Straightforward.

Lemma 3 (Store Typing Weakening). If $\Gamma; \Sigma \vdash u : T$, then for some $\Sigma' \supseteq \Sigma$, it holds that $\Gamma; \Sigma : \overline{S} \vdash u : T$.

Proof. Straightforward.

Lemma 4 (Method Body Type – Lemma A.3 from [3]).

(a) $\Gamma, z : T \vdash z$ has $\text{def} m (\overline{z} : R) \{ p \} : S = u$

(b) $\Gamma, z : T, z : T \vdash T \leq R$

Then for some type $S'$ it is the case that: $\Gamma, z : T, z : T \vdash u : S', S' \leq S$

Proof. Straightforward.

Lemma 5 (Cast). If $\Gamma; \Sigma \vdash H$ and $\Gamma; \Sigma \vdash l : S, S \leq T$, then $\Gamma; \Sigma \vdash H[l] : R, R \leq T$

Proof. Straightforward.

Lemma 6 (Evaluation Context Typing). If $\Gamma \vdash E[u] : T$, then for some type $S$ it holds that $\Gamma \vdash u : S$.

Proof. By induction on the structure of the evaluation context $E$.

Lemma 7 (Evaluation Context Step Typing). If $\Gamma; \Sigma \vdash E[u] : T, u : S$, and for some expression $u'$ and store typing $\Sigma' \supseteq \Sigma$ it holds that $\Gamma; \Sigma' \vdash u' : S', S' \leq S$, then $\Gamma; \Sigma' \vdash E[u'] : T', T' \leq T$

Proof. By induction on the structure of the evaluation context $E$.

Lemma 8 (Selfification). If $\Gamma, x : S \vdash S \leq T$ then $\Gamma, x : S \vdash S \leq \text{self} (T, x)$.

Proof. Straightforward.

Lemma 9 (Existential Weakening). If $\Gamma \vdash R \leq R'$ then $\Gamma \vdash \exists x : R, T \leq \exists x : R', T$.

Proof. Straightforward.

Lemma 10 (Existential Fold). If $\Gamma, z : S, x : T \vdash R \leq R'$, then $\Gamma, x : \exists z : S. T \vdash R \leq R'$, where $z$ does not appear in $R$ and $R'$.

Proof. Straightforward.

Theorem 1 (Subject Reduction).

(a) $\Gamma; \Sigma \vdash u : T$,

(b) $\Gamma; \Sigma \vdash H$,

(c) $H, u \rightarrow H', u'$,

then for some $T'$ and $\Sigma' \supseteq \Sigma$:

(d) $\Gamma; \Sigma' \vdash u' : T'$,

(e) $\Gamma \vdash T' \leq T$,

(f) $\Gamma; \Sigma' \vdash H'$.

Proof. We proceed by induction on the structure of fact (c):

$H, u \rightarrow H', u'$

We have the following cases:
• [RC-ECTX]: Fact (c) has the form:

\[ H, E[u_0] \mapsto H', E[u'_0] \]  \hspace{1cm} (6.1)

From (a):

\[ \Gamma; \Sigma \vdash E[u_0] : T \]  \hspace{1cm} (6.2)

From Lemma 6 on 6.2:

\[ \Sigma; \Gamma \vdash u_0 : T_0 \]  \hspace{1cm} (6.3)

By induction hypothesis, using 6.3, (b) and (c) we get:

\[ \Gamma; \Sigma' \vdash u'_0 : T'_0 \]  \hspace{1cm} (6.4)

\[ \Gamma; \Sigma' \vdash T'_0 \preccurlyeq T_0 \]  \hspace{1cm} (6.5)

\[ \Gamma; \Sigma' \vdash H' \]  \hspace{1cm} (6.6)

\[ \Sigma' \supseteq \Sigma \]  \hspace{1cm} (6.7)

For some type \( T'_0 \) and heap \( H' \). From 6.6 we prove (f).

From Lemma 7 using 6.2, 6.3, 6.4, 6.5 and 6.7:

\[ \Gamma; \Sigma' \vdash u'_0 : T'_0, T' \preccurlyeq T \]  \hspace{1cm} (6.8)

From 6.8 we prove (d) and (e).

• [R-FIELD]: Fact (c) has the form:

\[ H, l.h \mapsto H, v \]  \hspace{1cm} (6.9)

From (a) for \( u \equiv l.h \) we have:

\[ \Gamma; \Sigma \vdash l.h : T \]  \hspace{1cm} (6.10)

By inverting R-FIELD on 6.9:

\[ H[l] = \text{new} \; C \; (\overline{v}) \]  \hspace{1cm} (6.11)

From (b) for \( l \in \text{dom}(H) \), it holds by WH-EXT:

\[ \Gamma; \Sigma \vdash \text{new} \; C \; (\overline{v}) : S' \]  \hspace{1cm} (6.12)

By inverting WH-EXT on (b):

\[ \Sigma[l] = S \]  \hspace{1cm} (6.13)

\[ \Gamma \vdash S' \leq S \]  \hspace{1cm} (6.14)

From T-NEW on 6.12 it holds that:

\[ S' \equiv \exists z_1 : T_1. \{ \nu : C \; | \; \nu, \overline{\nu} = z_1 \land inv(C, \nu) \} \]  \hspace{1cm} (6.15)

By inverting T-NEW on 6.12:

\[ \Gamma; \Sigma \vdash \nu : (\bigcup_{1}, \bigcup_{w}) \]  \hspace{1cm} (6.16)

\[ \vdash \text{class} \; (C) \]  \hspace{1cm} (6.17)

\[ \Gamma, z : C ; \Sigma \vdash \text{fields} \; (z) = \Diamond \overline{T}, \overline{\nu}; \overline{\nu} \]  \hspace{1cm} (6.18)

\[ \Gamma, z : C, z_1 : \text{self} \; (\bigcup_{1}, z, \overline{\nu}) ; \Sigma \vdash \bigcup_{1} \leq \overline{R}, \overline{\bigcup_{w}} \leq \overline{\nu}, inv(C, z) \]  \hspace{1cm} (6.19)

We examine cases on the typing statement 6.10:
[T-FIELD-I]: Field $h$ is an immutable field $f_i$, so fact (a) becomes:

$$\Gamma; \Sigma \vdash \exists z : S. \text{self} \left( R_i, z.f_i \right)$$  \hspace{1cm} (6.20)

By inverting T-FIELD-I on 6.20:

$$\Gamma; \Sigma \vdash l : S$$  \hspace{1cm} (6.21)

$$\Gamma, z : S; \Sigma \vdash z \text{hasImm} f_i : R_i$$  \hspace{1cm} (6.22)

For a fresh $z$.

Keeping only the relevant part of 6.16 and 6.19:

$$\Gamma; \Sigma \vdash v_i : U_i$$  \hspace{1cm} (6.23)

$$\Gamma, z : C, z_i : \text{self} \left( U_i, z.f_i \right) ; \Sigma \vdash U_i \leq R_i$$  \hspace{1cm} (6.24)

By 6.23 we prove (d).

From Lemma 8 and 6.24, picking $z_i$ as the selfification variable:

$$\Gamma, z : C, z_i : \text{self} \left( U_i, z.f_i \right) ; \Sigma \vdash U_i \leq \text{self} \left( R_i, z_i \right)$$  \hspace{1cm} (6.25)

For the above environment it holds that:

$$\left[ \Gamma, z : C, z_i : \text{self} \left( U_i, z.f_i \right) ; \Sigma \right] \Rightarrow z_i = z.f_i$$  \hspace{1cm} (6.26)

By $\leq$-REFL and From Lemma 8 using 6.26:

$$\Gamma, z : C, z_i : \text{self} \left( U_i, z.f_i \right) ; \Sigma \vdash \text{self} \left( R_i, z_i \right) \leq \text{self} \left( \text{self} \left( R_i, z_i \right), z.f_i \right)$$  \hspace{1cm} (6.27)

By simplifying 6.27 using $\leq$-TRANS on 6.25 and 6.27 we get:

$$\Gamma, z : C, z_i : \text{self} \left( U_i, z.f_i \right) ; \Sigma \vdash U_i \leq \text{self} \left( R_i, z.f_i \right)$$  \hspace{1cm} (6.28)

From Lemma 10 using 6.15 and 6.28 we get:

$$\Gamma, z : S' \vdash U_i \leq \text{self} \left( R_i, z.f_i \right)$$  \hspace{1cm} (6.29)

From Rule $\leq$-WITNESS using 6.29:

$$\Gamma \vdash U_i \leq \exists z : S'. \text{self} \left( R_i, z.f_i \right)$$  \hspace{1cm} (6.30)

From Lemma 9 using 6.14 and 6.30:

$$\Gamma \vdash U_i \leq \exists z : S. \text{self} \left( R_i, z.f_i \right)$$  \hspace{1cm} (6.31)

Using 6.20, 6.16 and 6.31 we prove (e).

Heap $H$ does not evolve so (f) holds trivially.

[T-FIELD-M]: Field $h$ is a mutable field $g_i$, so fact (a) becomes:

$$\Gamma; \Sigma \vdash l.g_i : \exists z : S. V_i$$  \hspace{1cm} (6.32)

By inverting T-FIELD-M on 6.32:

$$\Gamma \vdash l : S$$  \hspace{1cm} (6.33)

$$\Gamma, l : S \vdash z \text{hasMut} g_i : V_i$$  \hspace{1cm} (6.34)

For a fresh $z$.

Keeping only the relevant parts of 6.16 and 6.19:

$$\Gamma \vdash v_i : U_i$$  \hspace{1cm} (6.35)

$$\Gamma, z : C, z_i : \text{self} \left( U_i, z.f_i \right) ; \Sigma \vdash U_i \leq V_i$$  \hspace{1cm} (6.36)
By 6.35 we prove (d).
From Lemma 10 using 6.15 and 6.36 we get:

\[ \Gamma, z : S' \vdash U_i \leq V_i \]  \hspace{1cm} (6.37)

From Rule \(\leq\text{-Witness}\) using 6.37:

\[ \Gamma \vdash U_i \leq \exists z : S'. V_i \]  \hspace{1cm} (6.38)

From Lemma 9 using 6.14 and 6.38:

\[ \Gamma \vdash U_i \leq \exists z : S.V_i \]  \hspace{1cm} (6.39)

Using 6.32, 6.16 and 6.39 we prove (e).
Heap \(H\) does not evolve so (f) holds trivially.

* [R-INVK]: Fact (c) has the form:

\[ H, l.m (\forall) \rightarrow H, [\forall/\exists, l/\text{this}] u' \]  \hspace{1cm} (6.40)

From (a) for \(u \equiv l.m (\forall)\) we have:

\[ \Gamma ; \Sigma \vdash l.m (\forall) : \exists z : T, \exists z : T. S \]  \hspace{1cm} (6.41)

By inverting T-INV on 6.41:

\[ \Gamma ; \Sigma ; l : T, \forall : \overline{T} \]  \hspace{1cm} (6.42)

\[ \Gamma, z : T, z : T \vdash z \text{ has } (\text{def } m (z, R) \{p\} : S = u') \]  \hspace{1cm} (6.43)

\[ \Gamma, z : T, z : \overline{T} \vdash \overline{T} \leq R \]  \hspace{1cm} (6.44)

\[ \Gamma, z : T, z : \overline{T} \vdash p \]  \hspace{1cm} (6.45)

With fresh \(z\) and \(z\).
By inverting R-INVK on 6.40:

\[ H[l] = \textbf{new } C (\ldots) \]  \hspace{1cm} (6.46)

\[ z : C \vdash z \text{ has } (\text{def } m (z, R) \{p\} : S = u') \]  \hspace{1cm} (6.47)

Note that this has already been substituted by \(z\) in \(S\).
By inverting WH-EXT on (c) using 6.46:

\[ \Sigma[l] = T \]  \hspace{1cm} (6.48)

\[ \Gamma ; \Sigma \vdash H[l] : T_0, T_0 \leq T \]  \hspace{1cm} (6.49)

From Lemma 4 using 6.43 and 6.44:

\[ \Gamma, z : T, z : \overline{T} \vdash u' : S', S' \leq S \]  \hspace{1cm} (6.50)

From 6.50 we prove (d).
From Rule \(\leq\text{-Witness}\) using 6.50:

\[ \Gamma \vdash S' \leq \exists z : T, \exists z : \overline{T}. S \]  \hspace{1cm} (6.51)

From Lemma 1 using 6.42, 6.44 and 6.50:

\[ \Gamma \vdash [\forall/\exists, l/\text{this}] u' : U, U \leq S' \]  \hspace{1cm} (6.52)

By Rule \(\leq\text{-Trans}\) on 6.50 and 6.52:

\[ \Gamma \vdash U \leq \exists z : T, \exists z : \overline{T}. S \]  \hspace{1cm} (6.53)

From 6.53 we prove (e).
Heap \(H\) does not evolve so (f) holds trivially.
• [R-Cast]: Fact (c) has the form:

\[ H, l \text{ as } T \mapsto H, l \]

From (a) for \( u \equiv l \text{ as } T \) we have:

\[ \Gamma; \Sigma \vdash l \text{ as } T : T \] (6.54)

By inverting T-Cast on 6.54:

\[ \Gamma; \Sigma \vdash S \] (6.55)
\[ \Gamma \vdash T \] (6.56)
\[ \Gamma \vdash S \preceq T \] (6.57)

From 6.55 and 6.57 we get (d) and (e), respectively.

\( H \) does not evolve, which proves (f), given (b)

• [R-New]: Fact (c) has the form:

\[ H, \text{new } C (\nu) \mapsto H', l \]

Where \( l \) is a fresh location and:

\( H' \equiv l \mapsto \text{new } C (\nu), H \)

From (a) for \( u \equiv \text{new } C (\nu) \) we have:

\[ \Gamma; \Sigma \vdash \text{new } C (\nu) : R_0 \] (6.58)

Where:

\[ R_0 \equiv \exists z_1: T_1. \{ \nu: C | \nu.\bar{f} = z_1 \land \text{inv}(C, \nu) \} \] (6.59)

By inverting T-New on 6.58:

\[ \Gamma \vdash \nu: (T_1, T_H) \] (6.60)
\[ \vdash \text{class}(C) \] (6.61)
\[ \Gamma, z: C \vdash \text{fields}(z) = \emptyset \bar{f}; \bar{R}, \bar{g}; \bar{U} \] (6.62)
\[ \Gamma, z: C, \bar{z}, \bar{t} = z_1 \vdash T_1 \leq \bar{R}, \bar{T_H} \leq \bar{U}, \\text{inv}(C, z) \] (6.63)

For fresh \( z \) and \( \bar{z} \).

We choose a store typing \( \Sigma' \), such that:

\[ \Sigma' = l \mapsto R_0, \Sigma \]

Hence:

\[ \Sigma'[l] = R_0 \] (6.64)

By applying rule T-LOC using the latest equation:

\[ \Gamma; \Sigma' \vdash l : R_0 \]

By \( \leq \text{-ID} \) we trivially get:

\[ \Gamma \vdash R_0 \leq R_0 \] (6.65)

Which prove (d) and (e).

By applying Lemma 3 on 6.58:

\[ \Gamma; \Sigma' \vdash \text{new } C (\nu) : R_0 \] (6.66)

Using 6.64, 6.65, 6.66 and (b), on rule WH-Ext:

\[ \Gamma; \Sigma' \vdash H' \]

Which proves (f).
[R-LETIN] Similar approach to case R-INVK.

[R-ASGN]: Fact (c) has the form:
\[ H, l.g_i = v' \rightarrow H', v' \]  
(6.67)

By inverting R-ASGN on 6.67:
\[ H[l] = \text{new } C \langle \emptyset \rangle \]  
(6.68)

\[ H' = l \rightarrow \text{new } C \langle \ldots, v_{i-1}, v', v_{i+1}, \ldots \rangle, H \]  
(6.69)

From (a) for \( u \equiv l.g_i = v' \):
\[ \Gamma; \Sigma \vdash l.g_i = v' : T' \]  
(6.70)

By inverting T-ASGN on 6.70:
\[ \Gamma \vdash l : T_i, v' : T' \]  
(6.71)

\[ \Gamma; z : [T_i]; \Sigma \vdash z \text{ hasMut } g_i : U_i, T' \leq U_i \]  
(6.72)

With fresh \( z \).
With 6.71 and \( \leq \text{-REFL} \) we prove (d) and (e).

By inverting T-LOC on 6.71:
\[ \Sigma[l] = T_i \]  
(6.73)

By inverting WH-EXT on (c) for location \( l \), that holds an object \( o \equiv H[l] \), and using 6.73:
\[ \Gamma; \Sigma \vdash o : S, S \leq T_i \]  
(6.74)

\[ \Gamma; \Sigma \vdash H \]  
(6.75)

By 6.68 and 6.74 we get:
\[ \Gamma; \Sigma \vdash \text{new } C \langle \emptyset \rangle : S \]  
(6.76)

By T-NEW, \( S \) is of the form:
\[ S \equiv \exists z_i : T_i. \{ \nu : C | \nu.t = z_i \wedge \text{inv}(C, \nu) \} \]  
(6.77)

By inverting T-NEW on 6.76:
\[ \Gamma \vdash \nu : T \]  
(6.78)

\[ \vdash \text{class } (C) \]  
(6.79)

\[ \Gamma; z : C \vdash \text{fields } (z) = \Diamond \bar{\nu}; \bar{R}, \bar{g} : \bar{U} \]  
(6.80)

\[ \Gamma; z : C, z_i : \text{self } (T_i, z, \bar{r}) \vdash \bar{T}_i \leq \bar{R}, \bar{T}_H \leq \bar{U}, \text{inv } (C, \nu) \]  
(6.81)

Where \( z \) and \( z_i \) are fresh and \( T \equiv (T_i, T_H) \).

By 6.74 it holds that:
\[ \Gamma \vdash [S] \leq [T_i] \]  
(6.82)

By 6.82 and 6.77:
\[ \Gamma \vdash C \leq [T_i] \]  
(6.83)

From Lemma A.6 in [3] using 6.72 and 6.83:
\[ \Gamma; z : C \vdash T' \leq U_i \]  
(6.84)
From Lemma 2 on 6.84:

\[ \Gamma, z : C, \bar{z}_1 : \text{self} (\bar{t}_1, z, \bar{t}) \vdash T' \leq U_i \] (6.85)

Let \( z_{n_i-1} \) and \( z_{n_i+1}, \ldots \), such that:

\[ \bar{z}_n = z_{n_i-1}, z_{n_i+1}, \ldots \]

and

\[ \bar{z}'_n = z_{n_i-1}, z_{n_i+1}, \ldots \]

Also if:

\[ \bar{v} = \ldots, v_{i-1}, v, v_{i+1}, \ldots \] and \( \bar{T} = \ldots, T_{i-1}, T, T_{i+1}, \ldots \)

Then:

\[ \bar{v}' = \ldots, v_{i-1}, v', v_{i+1}, \ldots \] and \( \bar{T}' = \ldots, T_{i-1}, T', T_{i+2}, \ldots \)

Combining 6.81 and 6.85:

\[ \Gamma, z : C, \bar{z}_1 : \text{self} (\bar{t}_1, z, \bar{t}) \vdash T' \leq (R, U), \text{inv} (C, z) \] (6.86)

Also from 6.71 and 6.78:

\[ \Gamma \vdash \bar{v}' : \bar{T}' \] (6.87)

By applying rule T-NEW using 6.87, 6.79, 6.80 and 6.86:

\[ \Gamma ; \Sigma \vdash \text{new } C (\bar{v}') : S' \] (6.88)

Where:

\[ S' \equiv \exists \bar{z}_1 : \bar{t}_1. \{ \nu : C \mid \nu. t = \bar{z}_1 \land \text{inv} (C, \nu) \} \] (6.89)

From 6.77 and 6.89:

\[ S = S' \]

Also by 6.74 for \( o' = \text{new } C (\bar{v}') \):

\[ \Gamma ; \Sigma \vdash o' : S', S' \leq T \] (6.90)

By applying rule WH-EXT on 6.73 6.90 and 6.75:

\[ \Gamma ; \Sigma \vdash \lambda \mapsto o', H \]

Which proves (f).

- [R-ITE-T] Similar approach to case RC-ECTX.
- [R-ITE-F] Similar approach to case RC-ECTX.

**Theorem 2** (Progress). If

(a) \( \Gamma ; \Sigma \vdash u : \bar{T} \)

(b) \( \Gamma ; \Sigma \vdash H \)

then one of the following holds:

- \( u \) is a value,
- there exist \( u', H' \) and \( \Sigma' \) s.t. \( \Gamma ; \Sigma \vdash H' \) and \( H, u \mapsto H', u' \).
\[\text{Proof. We proceed by induction on the structure of the derivation: } \Gamma; \Sigma \vdash u : T.\]

\* [T-FIELD-I]

\[
\Gamma; \Sigma \vdash u_0.f_i : \exists z. T_0. \text{self } (T, z.f_i) \tag{2.1}
\]

By inverting T-FIELD-I on 2.1:

\[
\begin{align*}
\Gamma; \Sigma & \vdash u_0 : T_0 \\
\Gamma; z : T_0; \Sigma & \vdash z \text{ hasImm } f_i : T
\end{align*}
\tag{2.2}
\tag{2.3}
\]

By i.h. using 2.2 and (b) there are two possible cases on \(u_0\):

\* \([u_0 \equiv l_0]\) Statement 2.2 becomes:

\[
\Gamma; \Sigma \vdash l_0 : T_0 \tag{2.4}
\]

From (b) for location \(l_0\):

\[
\Gamma; \Sigma \vdash l_0 \mapsto o, H \tag{2.5}
\]

Where:

\[
o \equiv \text{new } C (\forall) \tag{2.6}
\]

By inverting WH-EXT on 2.5:

\[
\Sigma[l_0] = T_0 \tag{2.7}
\]

\[
\Gamma; \Sigma \vdash o : S_0, S_0 \leq T_0 \tag{2.8}
\]

\[
\Gamma; \Sigma \vdash H \tag{2.9}
\]

From Lemma 5 using (b) and 2.8:

\[
\Gamma; \Sigma \vdash o : S_0, S_0 \leq T_0 \tag{2.10}
\]

From Lemma A.6 in [3] using 2.3 and 2.10:

\[
\Gamma; z : S_0; \Sigma \vdash z \text{ hasImm } f_i : T \tag{2.11}
\]

From R-FIELD using 2.5, 2.6 and 2.11:

\[
H, l_0.f_i \mapsto H, v_i
\]

\* \([\exists u' \text{ s.t. } H, u_0 \mapsto H', u'_0] \text{ By rule RC-CTX:}\]

\[
H, u_0.f_i \mapsto H', u'_0.f_i
\]

\* [T-FIELD-M] Similar to previous case.

\* [T-INV], [T-NEW] Similar to the respective case of CFJ [3].

\* [T-CAST]:

\[
\Gamma; \Sigma \vdash u_0 \text{ as } T : T \tag{2.12}
\]

By inverting T-CAST on 2.12:

\[
\begin{align*}
\Gamma & \vdash u_0 : S_0 \\
\Gamma; \Sigma & \vdash T \\
\Gamma; \Sigma & \vdash S_0 \preceq T
\end{align*}
\tag{2.13}
\tag{2.14}
\tag{2.15}
\]

By i.h. using 2.13 and (b) there are two possible cases on \(u_0\):
Statement 2.13 becomes:

\[ \Gamma; \Sigma \vdash l_0 : S_0 \]  

(2.16)

From Lemma 5 using (b) and 2.15:

\[ \Gamma; \Sigma \vdash H[l_0]; R_0, R_0 \leq T \]  

(2.17)

From R-Cast using 2.17:

\[ H, l_0 \text{ as } T \mapsto H, l_0 \]

- [\exists u'_0 \text{ s.t. } H, u_0 \mapsto H', u'_0] \text{ By rule RC-CTX:}

\[ H, u_0 \text{ as } T \mapsto H', u'_0 \text{ as } T \]

- [T-LET], [T-ASGN], [T-IF] These cases are handled in a similar manner.

\[ \square \]
References


