Mechanized Logical Relations for Termination-Insensitive Noninterference

Technical Appendix

August 6, 2020

Abstract

This document presents a λsec, a standard ML-like language with higher-order heap equipped with an information-flow control type system featuring subtyping, recursive types, label polymorphism, existential types, and impredicative type polymorphism. We introduce a generalized theory of Modal Weakest Precondition predicates and construct a novel “logical” logical-relations model of the type system in Iris, a state-of-the-art separation logic. Finally, we use the model to prove that the type system guarantees termination-insensitive noninterference.

1 Syntax and Semantics

Definition 1.1 (Syntax and types).

\[ x, y, z \in \text{Var} \]
\[ \iota \in \text{Loc} \]
\[ n \in \mathbb{N} \]
\[ l, \zeta \in \mathcal{L} \]
\[ \ominus ::= + | - | \ast | = | < \]
\[ \ell, \mathcal{L} \in \text{Label}_L \]
\[ \tau \in \text{LType} \]
\[ t \in \text{Type} \]
\[ e \in \text{Expr} \]
\[ v \in \text{Val} \]
\[ K \in \text{ECtx} \]
\[ \sigma \in \text{Loc}^{\text{fin}} \]

In addition to the given constructions we will write let \( x = e_1 \) in \( e_2 \) for the term \( (\lambda x. e_1) e_2 \) and \( e_1; e_2 \) for let \( _= = e_1 \) in \( e_2 \).

The syntax of types is parameterized over a bounded join-semilattice \( \mathcal{L} \) where the induced ordering \( \subseteq \) defines the security policy. \( \forall \ell \alpha. \tau \) denotes the type of label-polymorphic terms (over variable \( \alpha \)) with the corresponding term \( \Lambda e \). \( \forall \ell \kappa. \tau \) denotes the type of type-polymorphic terms (over variable \( \kappa \)) with the corresponding term \( \lambda e \). Both the two polymorphic types and the arrow type are annotated with a label \( \ell \) that in the type system will constitute a lower-bound on side-effects of the term.
Definition 1.2 (Operational semantics).

\[ v \uplus v' \xrightarrow{\text{pure}} v'' \]  
if \( v'' = v \uplus v' \)

if true then \( e_1 \) else \( e_2 \) \( \xrightarrow{\text{pure}} e_1 \)
if false then \( e_1 \) else \( e_2 \) \( \xrightarrow{\text{pure}} e_2 \)

\[ \pi_i (v_1, v_2) \xrightarrow{\text{pure}} v_i \]  
\( i \in \{1, 2\} \)

match inj_1 \( v \) with inj_i \( \Rightarrow e \) end \( \xrightarrow{\text{pure}} e[v/x] \)  
\( i \in \{1, 2\} \)

\[ (\lambda x. e) v \xrightarrow{\text{pure}} e[v/x] \]

\[ (\Lambda e) \xrightarrow{\text{pure}} e \]

unfold (fold \( v \)) \( \xrightarrow{\text{pure}} v \)

unpack (pack \( v \)) as \( x \) in \( e \) \( \xrightarrow{\text{pure}} e[v/x] \)

The operational semantics are mostly standard and defined with a call-by-value, left-to-right evaluation strategy. We first define a head reduction relation, \( (\sigma, e) \rightarrow_h (\sigma', e') \), which relates two pairs of a state and an expression. The head-step relation is lifted to a reduction relation \( (\sigma, e) \rightarrow (\sigma', e') \) using evaluation contexts.

2 Type System

Definition 2.1 (Label-ordering with free variables).

\[
\begin{array}{llll}
F\text{-REFL} & FV(\ell) \subseteq \Psi & \Psi \vdash \ell \subseteq \ell & F\text{-TRANS} & \Psi \vdash \ell_1 \subseteq \ell_2 & \Psi \vdash \ell_2 \subseteq \ell_3 & F\text{-BOTTOM} & FV(\ell) \subseteq \Psi & \Psi \vdash \perp \subseteq \ell & F\text{-LABEL} & \Psi \vdash \ell_1 \subseteq \ell_2 & \Psi \vdash \ell_1 \subseteq \ell_2 \\
F\text{-JOIN} & \Psi \vdash \ell_1 \subseteq \ell_3 & \Psi \vdash \ell_2 \subseteq \ell_3 & F\text{-SUM} & \Psi \vdash \ell_1 \subseteq \ell_2 & \Psi \vdash \ell_2 \subseteq \ell_3 \\
\end{array}
\]

Definition 2.2 (Subtyping).

\[
\begin{array}{llll}
S\text{-REFL} & FV(\ell) \subseteq \Xi & \Xi | \Psi \vdash t <: t & S\text{-TRANS} & \Xi | \Psi \vdash t_1 <: t_2 & \Xi | \Psi \vdash t_2 <: t_3 & \Xi | \Psi \vdash t_1 <: t_3 \\
S\text{-FORALL} & \Psi, \kappa \vdash \ell_2 \subseteq \ell_1 & \Xi | \Psi, \kappa \vdash t_1 <: t_2 & \Xi | \Psi, \kappa \vdash t_1 <: t_2 & \Xi, \alpha, \Psi \vdash t_1 <: t_2 & \Xi | \Psi \vdash \forall_\ell_1, \alpha, t_1 <: \forall_\ell_2, \alpha, t_2 \\
S\text{-PAR} & \Xi | \Psi \vdash t_1 <: t_1' & \Xi | \Psi \vdash t_2 <: t_2' & \Xi | \Psi \vdash t_1 + t_2 <: t_1' + t_2' & \Xi | \Psi \vdash t_1 \times t_2 <: t_1' \times t_2' & \Xi | \Psi \vdash t_1^{\ell_1} <: t_2^{\ell_2} \\
S\text{-PRODUCT} & \Xi | \Psi \vdash \ell_1 \subseteq \ell_2 & \Xi | \Psi \vdash \ell_2 \subseteq \ell_3 & \Xi | \Psi \vdash \ell_1 \subseteq \ell_3 & \Xi | \Psi \vdash \ell_2 \subseteq \ell_3 & \Xi | \Psi \vdash \ell_1 \subseteq \ell_3 \\
S\text{-LABEL} & \Xi | \Psi \vdash \ell_1 \subseteq \ell_2 & \Xi | \Psi \vdash \ell_2 \subseteq \ell_3 & \Xi | \Psi \vdash \ell_1 \subseteq \ell_3 & \Xi | \Psi \vdash \ell_2 \subseteq \ell_3 & \Xi | \Psi \vdash \ell_1 \subseteq \ell_3 \\
\end{array}
\]
Definition 2.3 (Protected-at).

\[ t' \setminus \ell \subseteq t \subseteq t' \]

Definition 2.4 (Typing).

| Rule | Premise | Conclusion |
|------|---------|------------|
| **T-Var** | \( x : \tau \in \Gamma \) | \( \Xi | \Psi | \Gamma \vdash_{pc} x : \tau \) |
| **T-Unit** | \( b \in \{\text{true, false} \} \) | \( \Xi | \Psi | \Gamma \vdash_{pc} b : \mathbb{B} \) |
| **T-Bool** | \( n \in \mathbb{N} \) | \( \Xi | \Psi | \Gamma \vdash_{pc} n : \mathbb{N} \) |
| **T-Binop** | | |
| **T-App** | \( \Xi | \Psi | \Gamma \vdash_{pc} e_1 : N^\ell \) \( \Xi | \Psi | \Gamma \vdash_{pc} e_2 : N^{\ell_2} \) \( \otimes : N \times N \Rightarrow t \) \( \Xi | \Psi | \Gamma \vdash_{pc} e_1 \odot e_2 : t' \subseteq \ell_1 \cup \ell_2 \) | \( \Xi | \Psi | \Gamma \vdash_{pc} e_1 : \tau \) \( \Psi \vdash \tau_2 \subseteq \ell \) \( \Psi \vdash pc \cup \ell \subseteq \ell_e \) |
| **T-Lam** | | |
| **T-App** | \( \Xi | \Psi | \Gamma \vdash_{pc} e_1 : (\ell_1 \to \ell_2) \) \( \Xi | \Psi | \Gamma \vdash_{pc} e_2 : \tau_1 \) \( \Psi \vdash \tau_2 \subseteq \ell \) \( \Psi \vdash pc \cup \ell \subseteq \ell_e \) | \( \Xi | \Psi | \Gamma \vdash_{pc} e_1 : \tau_2 \) |
| **T-Lam** | | | |
| **T-If** | \( \Xi | \Psi | \Gamma \vdash_{pc} e : \mathbb{B} \) \( \forall i \in \{1, 2 \} \) \( \Xi | \Psi | \Gamma \vdash_{pc} \ell_i e_i : \tau \) \( \Psi \vdash \tau \subseteq \ell \) | \( \Xi | \Psi | \Gamma \vdash_{pc} \text{if} \ e \text{ then } e_1 \text{ else } e_2 : \tau \) |
| **T-Proj** | \( \Xi | \Psi | \Gamma \vdash_{pc} e : (\tau_1 \times \tau_2) \) \( \Psi \vdash \tau_i \subseteq \ell \) \( i \in \{1, 2 \} \) | \( \Xi | \Psi | \Gamma \vdash_{pc} (e_1, e_2) : (\tau_1 \times \tau_2) \) |
| **T-Match** | \( \Xi | \Psi | \Gamma \vdash_{pc} e : (\tau_1 + \tau_2) \) \( \forall i \in \{1, 2 \} \) | \( \Xi | \Psi | \Gamma \vdash_{pc} e_i : \tau \) \( \Psi \vdash \tau \subseteq \ell \) |
| **T-Fold** | \( \Xi | \Psi | \Gamma \vdash_{pc} e : (\mu \alpha. \tau) \) | \( \Xi | \Psi | \Gamma \vdash_{pc} f o l d \ e : (\mu \alpha. \tau) \) |
| **T-Unfold** | \( \Xi | \Psi | \Gamma \vdash_{pc} e : (\mu \alpha. \tau) \) \( \Psi \vdash \tau (\mu \alpha. \tau) \subseteq \ell \) | \( \Xi | \Psi | \Gamma \vdash_{pc} \text{unfold} \ e : (\mu \alpha. \tau) \) |
| **T-Pack** | \( \Xi | \Psi | \Gamma \vdash_{pc} e : \tau (t/\alpha) \) | \( \Xi | \Psi | \Gamma \vdash_{pc} \text{pack} \ e : (\exists \alpha. \tau) \) |
| **T-unpack** | \( \Xi | \Psi | \Gamma \vdash_{pc} e : \tau \) \( \Xi | \Psi | \Gamma \vdash_{pc} \text{pack} \ e_1 : (\exists \alpha. \tau) \) \( \Xi, \alpha | \Psi, \Gamma, x : \tau' \vdash \text{pack} \ e_2 : \tau \) | \( \Xi | \Psi | \Gamma \vdash_{pc} \text{unpack} \ e_1 \text{ as } x \text{ in } e_2 : \tau \) |
\textbf{3 Modal Weakest Precondition (MWP)}

We refer to the Coq formalization for details not described in this document. Note that the MWP-theory is implicitly parameterized over a suitable language with expressions \( e \in \text{Expr} \), values \( v \in \text{Val} \), a stepping relation \((e, \sigma_1) \rightarrow (e_2, \sigma_2)\), and a state interpretation \( S : \text{State} \rightarrow \text{iProp} \).

\textbf{Definition 3.1 (MWP).} Let \( \mathcal{M} = (A, B, M, \text{BindCond}) \) where

\[ A, B : \text{Type} \]
\[ M : A \rightarrow \text{Masks} \rightarrow \mathbb{N} \rightarrow (B \rightarrow \text{Prop}) \rightarrow \text{Prop} \]
\[ \text{BindCond} : A \rightarrow A \rightarrow (B \rightarrow A) \rightarrow (B \rightarrow B \rightarrow B) \rightarrow \text{Prop} \]

with \( a \in A \) and \( \mathcal{E} \in \text{Masks} \) then

\[ \text{mwp}_{\mathcal{M},a}^\mathcal{E} \{ \Phi \} \triangleq \forall \sigma_1, \sigma_2, v, n. (e, \sigma_1) \rightarrow^n (v, \sigma_2) \rightarrow S(\sigma_1) \rightarrow M_{\mathcal{E},n}^a(\lambda b. \Phi(v, n, b) * S(\sigma_2)) \].

When omitting the mask \( \mathcal{E} \) we assume it as the largest possible mask \( \top \).

\textbf{Definition 3.2 (MWP validity).} A modality \( \mathcal{M} = (A, B, M, \text{BindCond}) \) is valid if

\[ \forall a, \mathcal{E}', n, \Phi, \mathcal{E} \subseteq \mathcal{E}' \Rightarrow \forall b. \Phi(b) \rightarrow \Psi(b) \vdash M_{\mathcal{E},n}^a(\Phi) \rightarrow M_{\mathcal{E}',n}^a(\Psi) \] (monotone)
\[ \forall a, \mathcal{E}, n, \Phi, M_{\mathcal{E},0,n}^a(\Phi) \vdash M_{\mathcal{E},n}^a(\Phi) \] (introducable)
\[ \forall a', f, g, \mathcal{E}, n, m, \Phi. \text{BindCond}(a, a', f, g) \Rightarrow \]
\[ M_{\mathcal{E},n}^a(\lambda b. M_{\mathcal{E}',m}^b(\lambda b'. \Phi(g(b, b')))) \vdash M_{\mathcal{E},n+m}^a(\Phi) \] (binding)

\textbf{Lemma 3.3 (M validity).} Given a valid modality \( \mathcal{M} = (A, B, M, \text{BindCond}) \) then

\textbf{MWP-INTRO}

\[ \forall v, n. M_{\mathcal{E},n}^a(\lambda b. \Phi(v, n, b)) \quad e \text{ executes purely} \]
\[ \text{mwp}_{\mathcal{M},a}^\mathcal{E} \{ \Phi \} \]

\textbf{MWP-MONO}

\[ \forall v, n, b, \Phi(v, n, b) \rightarrow \Psi(v, n, b) \quad \text{mwp}_{\mathcal{M},a}^\mathcal{E} \{ \Phi \} \]
\[ \text{mwp}_{\mathcal{M},a}^\mathcal{E} \{ \Psi \} \]

\textbf{MWP-BIND}

\[ \text{BindCond}(a, a', f, g) \quad \text{mwp}_{\mathcal{M},a'}^\mathcal{E} \{ v, n, b. \text{mwp}_{\mathcal{M},a}^\mathcal{E} f(b) \} \]
\[ \text{K}[e] \{ \{ w, m, b'. \Phi(w, n + m, g(b, b')) \} \} \]
\[ \text{mwp}_{\mathcal{M},a}^\mathcal{E} \{ \Phi \} \]

\textbf{Definition 3.4 (Atomic shift).} \( \mathcal{M} = (A, B, M, \text{BindCond}) \) supports atomic shifts at \( a \) if

\[ \forall E_1, E_2, n, \Phi, n \leq 1 \Rightarrow E_1 \parallel E_2 M_{\mathcal{E},n}^a(\lambda b. E_2 \parallel E_1 \Phi(b)) \rightarrow M_{\mathcal{E},n}^a(\Phi) \]
Definition 3.5 (Atomic Operation).
atomic(e) ≜ ∀σ, σ', e'. (σ, e) → (σ', e') ⇒ e' ∈ Val

Definition 3.6 (Reducible Operation).
reducible(e, σ) ≜ ∃e', σ'. (σ, e) → (σ', e')

Lemma 3.7 (MWP Atomic Step). Given M that supports atomic shifts at a then
\[
\frac{\text{MWP-Atomic}}{\quad \text{atomic(e)}}
\]

Definition 3.8 (M splitting). Let M₁, M₂ : Masks → iProp → iProp be two modalities indexed by masks. M can be split into (M₁, M₂), written SplitsInto(M; M₁, M₂, a), if
\[
\begin{align*}
&\forall \varepsilon, n, \Phi. \ M_1(\varepsilon) (M_2(\varepsilon) (M_{\varepsilon,n}(\Phi))) \vdash M_{\varepsilon,n+1}(\Phi) \\
&\forall \varepsilon, P, Q. P \Rightarrow Q \vdash M_1(\varepsilon)(P) \Rightarrow M_1(\varepsilon)(Q) \\
&\forall \varepsilon, P, Q. P \Rightarrow Q \vdash M_2(\varepsilon)(P) \Rightarrow M_2(\varepsilon)(Q)
\end{align*}
\]

Lemma 3.9 (Lifting). Let a ∈ A and M a modality with SplitsInto(M; M₁, M₂, a) then
\[
\frac{\text{MWP-lift-step} \quad e_1 \notin Val \quad \forall \sigma_1, S(\sigma_1) \Rightarrow M_1(\varepsilon) \left(\forall \sigma_2, e_2. (e, \sigma_1) \Rightarrow (e_2, \sigma_2) \Rightarrow M_2(\varepsilon) (S(\sigma_2) \bowtie mwp_{\varepsilon} M_{\varepsilon}^{M,a} e_2 \{v, n, b. \Phi(v, n, 1 + b)\})\right)}{mwp_{\varepsilon} M_{\varepsilon}^{M,a} e_1 \{\Phi\}}
\]

Definition 3.10 (MWP instance: Unary update). Let M₁ [a] ≜ (1, 1, M, BindCond) where
\[
M_{\varepsilon,n}(\Phi) \equiv \Rightarrow_{\varepsilon} \Phi()
\]

Lemma 3.11 (Properties of M₁ [a]).
1. M₁ [a] defines a valid modality.
2. M₁ [a] supports atomic shifts.
3. SplitsInto(M; E₁ [a], E₂ [a], E [a]).

Lemma 3.12 (Unary update MWP always supports atomic shifts).
\[
\frac{\text{E₁ [a]}}{\quad \text{MWP}_{E_1}^{M,a} e \{v, n, b. \Rightarrow_{E_1} \Phi(v, n, b)\} \Rightarrow \text{MWP}_{E_1}^{M,a} e \{\Phi\}}
\]

Definition 3.13 (MWP instance: Unary step-update). Let M₁ [a, f, g] ≜ (1, 1, M, BindCond) where
\[
M_{\varepsilon,n}(\Phi) \equiv \epsilon^{\Rightarrow_0 \bowtie_0 \Phi_0} \Rightarrow_\varepsilon \Phi()
\]

Lemma 3.14 (Properties of M₁ [a, f, g]).
1. M₁ [a, f, g] defines a valid modality.
2. M₁ [a, f, g] supports atomic shifts.
3. SplitsInto(M; E₁ [a, f, g], E₂ [a, f, g]).
Definition 3.15 (MWP instance: Binary update). Let $M \downarrow_{\phi} \triangleq (Expr, Val \times N, M, \text{BindCond})$ where

$$M_{\varepsilon,n}^\phi(\Phi) \triangleq mwp_{\varepsilon}^M \varepsilon \{ w, m, \Phi(w, m) \}$$

$$\text{BindCond}(e_1, e_2, f, g) \triangleq \exists K. e_1 = K[e_2] \land g = \lambda(v_1, n_1), (v_2, n_2). (v_2, n_1 + n_2) \land \forall v, k. f(v, k) = K[v].$$

Lemma 3.16 (Properties of $M \downarrow_{\phi}$).

1. $M \downarrow_{\phi}$ defines a valid modality.
2. $\forall a. \text{SplitsInto}(M; \varepsilon_{\downarrow}^{\varepsilon}, \emptyset \varepsilon_{\downarrow}^{\varepsilon}, a)$.

Fact 3.17 (Unfolding MWP with $M \downarrow_{\phi}$). By unfolding the definition of MWP instantiated with $M \downarrow_{\phi}$ we get:

$$mwp_{\varepsilon}^M \downarrow_{\phi^{v_1}}^e e_1 \{ \Phi \} = \forall \sigma_1, \sigma'_1, v, n, (e_1, \sigma_1) \rightarrow^n (v, \sigma'_1) \Rightarrow S_1(\sigma_1) \Rightarrow$$

$$M_{\varepsilon,n}^\phi(\lambda X. \Phi(v, n, X) \ast S_1(\sigma'_1))$$

$$= \forall \sigma_1, \sigma'_1, v, n, (e_1, \sigma_1) \rightarrow^n (v, \sigma'_1) \Rightarrow S_1(\sigma_1) \Rightarrow$$

$$mwp_{\varepsilon}^M \downarrow_{\phi^{v_1}}^e e_2 \{ w, m, \Phi(v, n, (w, m)) \ast S_1(\sigma'_1) \}$$

$$= \forall \sigma_1, \sigma'_1, v, n, (e_1, \sigma_1) \rightarrow^n (v, \sigma'_1) \Rightarrow S_1(\sigma_1) \Rightarrow$$

$$\forall \sigma_2, \sigma'_2, w, m, (e_2, \sigma_2) \rightarrow^m (w, \sigma'_2) \Rightarrow S_2(\sigma_2) \Rightarrow$$

$$M_{\varepsilon,n}^\phi(\lambda X. \Phi(v, n, (w, m)) \ast S_1(\sigma'_1) \ast S_2(\sigma'_2))$$

$$= \forall \sigma_1, \sigma'_1, v, n, (e_1, \sigma_1) \rightarrow^n (v, \sigma'_1) \Rightarrow S_1(\sigma_1) \Rightarrow$$

$$\forall \sigma_2, \sigma'_2, w, m, (e_2, \sigma_2) \rightarrow^m (w, \sigma'_2) \Rightarrow S_2(\sigma_2) \Rightarrow$$

$$\varepsilon_{\downarrow}^{\varepsilon}(\Phi(v, n, (w, m)) \ast S_1(\sigma'_1) \ast S_2(\sigma'_2))$$

Lemma 3.18 (Unary update MWP implies binary update MWP).

$$mwp_{\varepsilon}^M \downarrow_{\phi^{v_1}}^e e_1 \{ v, n, \text{mwp}_{\varepsilon}^M \downarrow_{\phi^{v_1}}^e e_2 \{ w, m, \Phi(v, n, (w, m)) \} \} \rightarrow mwp_{\varepsilon}^M \downarrow_{\phi^{v_1}}^e e_1 \{ \Phi \}$$

$$mwp_{\varepsilon}^M \downarrow_{\phi^{v_1}}^e e_2 \{ w, m, \text{mwp}_{\varepsilon}^M \downarrow_{\phi^{v_1}}^e e_1 \{ v, n, \Phi(v, n, (w, m)) \} \} \rightarrow mwp_{\varepsilon}^M \downarrow_{\phi^{v_1}}^e e_1 \{ \Phi \}$$

Lemma 3.19 (Binary update MWP always supports shifts).

$$\varepsilon_{\downarrow}^{\varepsilon} mwp_{\varepsilon}^M \downarrow_{\phi^{v_1}}^e e_1 \{ v, n, b, \varepsilon_{\downarrow}^{\varepsilon} \Phi(v, n, b) \} \rightarrow mwp_{\varepsilon}^M \downarrow_{\phi^{v_1}}^e e_1 \{ \Phi \}$$

Definition 3.20 (MWP instance: Binary step-update). Let $M_I \triangleq (N, 1, M, \text{BindCond})$ where

$$M_{\varepsilon,n}^\phi(\Phi) \triangleq (\varepsilon_{\downarrow}^{\varepsilon} \Phi(v, n, b))^{n+m} \Phi()$$

$$\text{BindCond}(n, m, f, g) \triangleq m \leq n \land \forall x, f(x) = n - m \land g = id.$$
Fact 3.22 (Unfolding MWP with $\mathcal{M}_{\times \lambda^0}$). By unfolding the definition of MWP instantiated $\mathcal{M}_{\times \lambda^0}$, we get:

$$\text{mwp}_{\mathcal{M}_{\times \lambda^0}^{c_2}} e_1 \{ \Phi \} = \forall \sigma_1, \sigma'_1, v, n, (e_1, \sigma_1) \rightarrow^n (v, \sigma'_1) \Rightarrow S_1(\sigma_1) \Rightarrow$$

$$\text{mwp}_{\mathcal{M}_{\times \lambda^0}^{c_2}} (\lambda X. \Phi(v, n, X) \times S_1(\sigma'_1)) = \forall \sigma_1, \sigma'_1, v, n, (e_1, \sigma_1) \rightarrow^n (v, \sigma'_1) \Rightarrow S_1(\sigma_1) \Rightarrow$$

$$\text{mwp}_{\mathcal{M}_{\times \lambda^0}^{c_2}} e_2 \{ w, m. \Phi(v, n, (w, m)) \times S_1(\sigma'_1) \} = \forall \sigma_1, \sigma'_1, v, n, (e_1, \sigma_1) \rightarrow^n (v, \sigma'_1) \Rightarrow S_1(\sigma_1) \Rightarrow$$

$$\forall \sigma_2, \sigma'_2, w, m. (e_2, \sigma_2) \rightarrow^n (w, \sigma'_2) \Rightarrow S_2(\sigma_2) \Rightarrow$$

$$\text{mwp}_{\mathcal{M}_{\times \lambda^0}^{c_2}} ((\lambda X. \Phi(v, n, (w, m)) \times S_1(\sigma'_1) \times S_2(\sigma'_2))) \Rightarrow$$

$$\forall \sigma_1, \sigma'_1, v, n, (e_1, \sigma_1) \rightarrow^n (v, \sigma'_1) \Rightarrow S_1(\sigma_1) \Rightarrow$$

$$\forall \sigma_2, \sigma'_2, w, m. (e_2, \sigma_2) \rightarrow^n (w, \sigma'_2) \Rightarrow S_2(\sigma_2) \Rightarrow$$

$$\text{mwp}_{\mathcal{M}_{\times \lambda^0}^{c_2}} (\lambda \sigma X. \Phi(v, n, (w, m)) \times S_1(\sigma'_1) 	imes S_2(\sigma'_2))) \Rightarrow$$

$$\text{mwp}_{\mathcal{M}_{\times \lambda^0}^{c_2}} e_1 \{ v, n. \Phi(v, n, (w, m)) \} = \text{mwp}_{\mathcal{M}_{\times \lambda^0}^{c_2}} e_1 \{ \Phi \}$$

Lemma 3.23 (Unary step-update MWP implies binary step-update MWP).

$$\text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_1 \{ v, n, \text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_2 \{ w, m. \Phi(v, n, (w, m)) \} \} \Rightarrow \text{mwp}_{\mathcal{M}_{\times \lambda^0}^{c_2}} e_1 \{ \Phi \}$$

$$\text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_2 \{ w, m. \text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_1 \{ v, n. \Phi(v, n, (w, m)) \} \} \Rightarrow \text{mwp}_{\mathcal{M}_{\times \lambda^0}^{c_2}} e_1 \{ \Phi \}$$

Lemma 3.24 (Double atomicity of binary step-update MWP). If atomic($e_1$) and atomic($e_2$) then

$$\text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_1 \{ v, n, \text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_2 \{ w, m. \text{atomic}(\text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_1 \{ \Phi \}) \} \} \Rightarrow \text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_1 \{ \Phi \}$$

Let $\text{atomic}(e_1)$.

$$\text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_2 \{ w, m. \text{atomic}(\text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_1 \{ v, n. \text{atomic}(\text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_2 \{ \Phi \}) \}) \} \Rightarrow \text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_1 \{ \Phi \}$$

Lemma 3.25 (Binary update MWP implies binary step-update MWP). Let

$$\text{reduces}(e, S, \mathcal{E}) \equiv \forall \sigma. \mathcal{E}(\sigma) \not\equiv^{*} \text{reducible}(e, \sigma).$$

Then

$$\text{reduces}(e_1, S_1, \mathcal{E}_1) \lor \text{reduces}(e_2, S_2, \mathcal{E}_1) \land$$

$$\left( \mathcal{E}_1 \Rightarrow \mathcal{E}_2 \Rightarrow \text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_1 \{ v, n, b. \mathcal{E}_2 \Rightarrow \mathcal{E}_1 \Phi(v, n, (w, m)) \} \right) \Rightarrow \text{mwp}_{\mathcal{M}_{\lambda^0}^{c_2}} e_1 \{ \Phi \}.$$
Lemma 3.28 (Properties of unary step-taking update MWP with $\lambda_{sec}$).

1. \( \forall \iota. \iota \hookrightarrow v \rightarrow Q \iota \vdash \text{mwp}^E_{\lambda_{sec}} \text{ref}(v) \{ v, Q \} \)
2. \( \iota \hookrightarrow v \leftrightarrow \iota \hookrightarrow (v \rightarrow Q \iota) \vdash \text{mwp}^E_{\lambda_{sec}} \text{!} \{ v, Q \} \)
3. \( \iota \hookrightarrow v \leftrightarrow \iota \hookrightarrow (w \rightarrow Q () ) \vdash \text{mwp}^E_{\lambda_{sec}} \text{!} \leftarrow w \{ v, Q \} \)

4 Logical Relations

The binary value relation is an Iris relation of type \( \text{Rel} \triangleq \text{Val} \times \text{Val} \rightarrow \text{iProp} \). Similarly, the unary value relation is an Iris predicate of type \( \text{Pred} \triangleq \text{Val} \rightarrow \text{iProp} \).

Both the unary and binary logical relation is implicitly quantified over a lattice \( \mathcal{L} \) and an observer/attacker label \( \zeta \). The environment \( \rho : \text{Lvar} \rightarrow \mathcal{L} \) maps label variables to semantic labels from \( \mathcal{L} \) and \( \Theta \) is a semantic type environment for type variables, as is usual for interpretations of languages with parametric polymorphism. However, for every type variable we keep both a binary relation and two unary relations, one for each of the two sides:

\[ \Theta : \text{Tvar} \rightarrow \text{Rel} \times \text{Pred} \times \text{Pred}. \]

We use \( \Theta_L, \Theta_R : \text{Tvar} \rightarrow \text{Pred} \) as shorthand for \( \pi_2 \circ \Theta \) and \( \pi_3 \circ \Theta \), respectively, where \( \pi_i(x) \) denotes the \( i \)th projection of \( x \). We will use

\[ \text{mwp}^E e_1 \sim e_2 \{ v, w, Q \} \]

as shorthand for \( \text{mwp}^E_{\lambda_{sec}} e_1 \{ v, \_, (w, \_). Q \} \).

Definition 4.1 (Label interpretation).

\[
\begin{align*}
[\cdot] : (\text{Lvar} \rightarrow \mathcal{L}) \rightarrow \text{Label}_E \rightarrow \mathcal{L} \\
[k]_\rho & \triangleq \rho(k) \\
[l]_\rho & \triangleq l \\
[l_1 \sqcup l_2]_\rho & \triangleq [l_1]_\rho \sqcup [l_2]_\rho
\end{align*}
\]
Definition 4.2 (Unary value interpretation).

\[
\begin{align*}
[a]_\Delta^\rho & \triangleq \Delta(a) \\
[1]_\Delta^\rho (v) & \triangleq v = () \\
[\mathbb{B}]_\Delta^\rho (v) & \triangleq v \in \{\text{true}, \text{false}\} \\
[N]_\Delta^\rho (v) & \triangleq v \in \mathbb{N} \\
\llbracket \tau_1 \times \tau_2 \rrbracket^\rho_\Delta (v) & \triangleq \exists v_1, v_2. v = (v_1, v_2) \ast \llbracket \tau_1 \rrbracket^\rho_\Delta (v_1) \ast \llbracket \tau_2 \rrbracket^\rho_\Delta (v_2) \\
\llbracket \tau_1 + \tau_2 \rrbracket^\rho_\Delta (v) & \triangleq \bigvee_{i \in \{1, 2\}} \exists w. v = \text{inj}_i w \ast \llbracket \tau_i \rrbracket^\rho_\Delta (w) \\
\llbracket \tau \rightarrow \tau \rrbracket^\rho_\Delta (v) & \triangleq \square (\forall w. \llbracket \tau \rrbracket^\rho_\Delta (w) \rightarrow \mathcal{E}_e [\tau \rrbracket^\rho_\Delta (v \ w)) \\
\llbracket \forall \ell. \alpha. \tau \rrbracket^\rho_\Delta (v) & \triangleq \square (\forall \Phi : \text{Pred.} \mathcal{E}_e [\tau \rrbracket^\rho_\Delta (v \ w) \\
\llbracket \exists a. \tau \rrbracket^\rho_\Delta (v) & \triangleq \square (\exists \Phi : \text{Pred.} \exists w. v = \text{pack} w \ast \llbracket \tau \rrbracket^\rho_\Delta (w)) \\
\llbracket \forall \ell. \kappa. \tau \rrbracket^\rho_\Delta (v) & \triangleq \square (\forall \ell \in L. \mathcal{E}_e [\tau \rrbracket^\rho_\Delta (v \ w) \\
\llbracket \mu \alpha. \tau \rrbracket^\rho_\Delta & \triangleq \mu \Phi : \text{Pred.} \lambda \alpha \exists w. v = \text{fold} w \ast \llbracket \tau \rrbracket^\rho_\Delta (w) \\
\llbracket \text{ref}(t') \rrbracket^\rho_\Delta (v) & \triangleq \exists w, N. v = t \ast \mathcal{R}(\Delta, \rho, \ell, N)
\end{align*}
\]

\[
\mathcal{R}(\Delta, \rho, \ell, N) \triangleq \begin{cases} 
\square \forall \ell. N' \subseteq \mathcal{E} \Rightarrow \begin{cases} 
\begin{aligned}
\varepsilon \in \mathcal{E} \ni \varepsilon \ast \llbracket \exists w. \ell \mapsto i, w * \llbracket \tau \rrbracket_\Delta^\rho (w) * \llbracket \mathcal{N} \rrbracket^\rho_\Delta (w) \ni \mathcal{E} \ni \text{True} \end{aligned}
\end{cases} & \text{if } [\ell]_\rho \subseteq \zeta \\
\square \forall \mathcal{E}, N' \subseteq \mathcal{E} \Rightarrow \begin{cases} 
\begin{aligned}
\varepsilon \in \mathcal{E} \ni \varepsilon \ast \llbracket \exists w. \ell \mapsto i, w * \llbracket \tau \rrbracket_\Delta^\rho (w) * \llbracket \mathcal{N} \rrbracket^\rho_\Delta (w) \ni \mathcal{E} \ni \text{True} \end{aligned}
\end{cases} & \text{if } [\ell]_\rho \not\subseteq \zeta
\end{cases}
\]

\[
[\ell']_\Delta^\rho (v) \triangleq [\ell]_\Delta^\rho (v)
\]

Definition 4.3 (Unary expression interpretation).

\[
\mathcal{E}_e [\tau]_\Delta^\rho (e) \triangleq [pc]_\rho \not\subseteq \zeta \Rightarrow \text{mwp}^\mathcal{M}_\mathcal{E} e \ni \llbracket \tau \rrbracket^\rho_\Delta
\]

Definition 4.4 (Unary environment interpretation).

\[
\mathcal{G}[\llbracket \tau \rrbracket^\rho_\Delta (e) \triangleq \text{True} \\
\mathcal{G}[\Gamma, x : \tau]_\Delta^\rho (\overrightarrow{w}) \triangleq \mathcal{G}[\Gamma]_\Delta^\rho (\overrightarrow{w}) \ast \llbracket \tau \rrbracket^\rho_\Delta (w)
\]

Definition 4.5 (Unary semantic typing).

\[
\Xi \vdash \Psi : | \Gamma \vdash_{pc} e : \tau \triangleq \begin{cases} 
\begin{aligned}
\forall \Delta, \rho, \overrightarrow{w}. \text{dom}(\Xi) \subseteq \text{dom}(\Delta) \ast \text{dom}(\Psi) \subseteq \text{dom}(\rho) \ni \end{aligned} \\
\mathcal{G}[\Gamma]_\Delta^\rho (\overrightarrow{w}) \ni \mathcal{E}_e [\tau \rrbracket^\rho_\Delta (e[\overrightarrow{w} / \overrightarrow{x}])
\end{cases}
\]

Lemma 4.6 (Unary semantic subtyping). If dom(\Xi) \subseteq dom(\Delta) and dom(\Psi) \subseteq dom(\rho) then

\[
\Xi \vdash \tau_1 <: \tau_2 \Rightarrow \llbracket \tau_1 \rrbracket^\rho_\Delta (v) \rightarrow \llbracket \tau_2 \rrbracket^\rho_\Delta (v)
\]

Theorem 4.7 (Unary fundamental theorem).

\[
\Xi \vdash \Psi : | \Gamma \vdash_{pc} e : \tau \Rightarrow \Xi \vdash \Psi : | \Gamma \vdash_{pc} e : \tau
\]
Definition 4.8 (Binary value interpretation).

- $\left[ \alpha \right]_{\Theta}^{\sigma} \triangleq \pi_1 (\Theta (\alpha))$
- $\left[ 1 \right]_{\Theta}^{\sigma} (v, v') \triangleq v = v'$
- $\left[ \mathbb{B} \right]_{\Theta}^{\sigma} (v, v') \triangleq v = v' \in \{ \text{true}, \text{false} \}$
- $\left[ \mathbb{N} \right]_{\Theta}^{\sigma} (v, v') \triangleq v = v' \in \mathbb{N}$
- $\left[ \tau_1 \times \tau_2 \right]_{\Theta}^{\sigma} (v, v') \triangleq \exists v_1, v_2, v_1', v_2' \quad v = (v_1, v_2) \ast v' = (v_1', v_2') \ast \left[ \tau_1 \right]_{\Theta}^{\sigma} (v_1, v_1') \ast \left[ \tau_2 \right]_{\Theta}^{\sigma} (v_2, v_2')$
- $\left[ \tau_1 + \tau_2 \right]_{\Theta}^{\sigma} (v, v') \triangleq \bigvee_{i \in \{1,2\}} \exists w, w'. v = inj_i w \ast v' = inj_i w' \ast \left[ \tau_1 \right]_{\Theta}^{\sigma} (w, w')$
- $\left[ \tau_1 \cdot \tau_2 \right]_{\Theta}^{\sigma} (v, v') \triangleq (\forall w, w'. \left[ \tau_1 \right]_{\Theta}^{\sigma} (w, w') \rightarrow \mathcal{E} [\tau_2]_{\Theta}^{\sigma} (v w, v' w))$
- $\forall \varepsilon, \alpha. \tau \mid_{\Theta}^{\sigma} (v, v') \triangleq (\forall \Phi : \text{Rel}. \forall \Psi, \Phi_R : \text{Pred}$. \(
\square (\forall v, v'. \Phi (v, v') \rightarrow \Phi (v) \ast \Phi_R (v'))) \rightarrow \mathcal{E} [\tau]_{\Theta}^{\sigma} (v w, v' w)$.
- $\exists \alpha. \tau \mid_{\Theta}^{\sigma} (v, v') \triangleq (\exists \Phi : \text{Rel}. \exists \Psi, \Phi_R : \text{Pred}. \square (\forall v, v'. \Phi (v, v') \rightarrow \Phi (v) \ast \Phi_R (v'))) \ast \exists w, w'. v = \text{pack} w \ast v' = \text{pack} w' \ast \left[ \tau \right]_{\Delta, \alpha \rightarrow (\Psi, \Phi, \Phi_R)}^\sigma (w, w')$
- $\forall \varepsilon, \alpha. \tau \mid_{\Theta}^{\sigma} (v, v') \triangleq (\forall \varepsilon \in \mathcal{L}. \mathcal{E} [\tau]_{\Theta}^{\sigma, \text{pack}} (v w, v' w)) \ast \left[ \forall \varepsilon, \alpha. \tau \mid_{\Theta}^{\sigma} (v) \right]_{\Theta}^{\sigma} (v)$
- $\exists \alpha. \tau \mid_{\Theta}^{\sigma} (v, v') \triangleq (\exists \alpha : \text{Rel}. \lambda (v, v'). \exists w, w'. v = \text{fold} w \ast v' = \text{fold} w' \ast \left[ \tau \mid_{\Theta, \alpha \rightarrow (\Psi, \Phi, \Phi_R)}^{\sigma} (w, w') \right]_{\Theta}^{\sigma} (v v', v)$
- $\left[ \text{ref} (\tau) \right]_{\Theta}^{\sigma} (v, v') \triangleq \exists i', v. v = i \ast v' \rightarrow \left[ \exists w, w'. i \rightarrow_{\text{L}} w \ast i' \rightarrow_{\text{R}} w' \ast \left[ \text{ref} (\tau) \right]_{\Theta}^{\sigma} (w, w') \right]_{\Theta}^{\sigma} (v, v')$
- $\left[ v' \right]_{\Theta}^{\sigma} (v, v') \triangleq \left[ \left[ \text{ref} (\tau) \right]_{\Theta}^{\sigma} (v, v') \right]_{\Theta}^{\sigma} (v, v')$

Definition 4.9 (Binary expression interpretation).

- $\mathcal{E} [\tau]_{\Theta}^{\sigma} (v, v') \triangleq \text{mwp} e_1 \sim e_2 \left\{ \left[ \tau \right]_{\Theta}^{\sigma} \right\}$

Definition 4.10 (Binary environment interpretation).

- $\mathcal{G} [\Gamma]_{\Theta}^{\sigma} (v, v) \triangleq \text{True}$
- $\mathcal{G} [\Gamma, x : \tau]_{\Theta}^{\sigma} (v_1 w_1, v_2 w_2) \triangleq \mathcal{G} [\Gamma]_{\Theta}^{\sigma} (v_1, v_2) \ast \left[ \tau \right]_{\Theta}^{\sigma} (w_1, w_2)$

Definition 4.11 (Binary environment coherence).

- $\text{Coh} (\Theta) \triangleq \bigwedge_{(\Psi, \Phi, \Phi_R) \in \Theta} \square (\forall v_L, v_R. \Phi (v_L, v_R) \rightarrow \Phi (v_L) \ast \Phi_R (v_R))$

Definition 4.12 (Binary semantic typing).

- $\Xi \mid \Psi \mid \Gamma \vdash \varepsilon_1 \approx_{\varepsilon} \varepsilon_R : \tau \triangleq \square \left( \forall \Theta, \rho. \varepsilon_1 \approx \varepsilon_R. \text{dom} (\Xi) \subseteq \text{dom} (\Theta) \ast \text{dom} (\Psi) \subseteq \text{dom} (\rho) \ast \text{Coh} (\Theta) \ast \mathcal{G} [\Gamma]_{\Theta}^{\sigma} (v_1, v_2) \rightarrow \mathcal{E} [\tau]_{\Theta}^{\sigma} (\varepsilon_1 [\varepsilon_2 / \varepsilon], \varepsilon_R [\varepsilon_2 / \varepsilon]) \right)$

Lemma 4.13 (Binary semantic subtyping). If $\text{dom} (\Xi) \subseteq \text{dom} (\Theta)$ and $\text{dom} (\Psi) \subseteq \text{dom} (\rho)$ then

- $\Xi \mid \Psi \vdash \tau_1 \ll \tau_2 \Rightarrow \left[ \tau_1 \right]_{\Theta}^{\sigma} (v_L, v_R) \rightarrow \left[ \tau_2 \right]_{\Theta}^{\sigma} (v_L, v_R)$

Lemma 4.14 (Binary-unary subsumption).

- $\text{Coh} (\Theta) \ast \left[ \tau \right]_{\Theta}^{\sigma} (v_L, v_R) \rightarrow \left[ \tau \right]_{\Theta}^{\sigma} (v_L) \ast \left[ \tau \right]_{\Theta}^{\sigma} (v_R)$
**Theorem 4.15** (Binary fundamental theorem).

\[ \Xi | \Psi | \Gamma \vdash_{pc} e : \tau \Rightarrow \Xi | \Psi | \Gamma \vdash e \approx_{\zeta} e : \tau \]

**Theorem 4.16** (Termination-Insensitive Noninterference). Let \( \top \) and \( \bot \) be labels drawn from a join-semilattice such that \( \bot \subseteq \zeta \) and \( \top \nsubseteq \zeta \). If

\[
\begin{align*}
\cdot | \cdot | x : B^\top \vdash e : B^\bot, \\
\cdot | \cdot | \vdash v_1 : B^\top, \text{ and } \cdot | \cdot | \vdash v_2 : B^\top
\end{align*}
\]

then

\[
(\emptyset, e[v_1/x]) \rightarrow^* (\sigma_1, v'_1) \land (\emptyset, e[v_2/x]) \rightarrow^* (\sigma_2, v'_2) \Rightarrow v'_1 = v'_2.
\]