In security-typed programming languages, types statically enforce noninterference between potentially conspiring values, such as the arguments and results of functions. But to adopt static security types, like other advanced type disciplines, programmers face a steep wholesale transition, often forcing them to refactor working code just to satisfy their type checker. To provide a gentler path to security typing that supports safe and stylistic but hard-to-verify programming idioms, researchers have designed languages that blend static and dynamic checking of security types. Unfortunately most of the resulting languages only support static, type-based reasoning about noninterference if a program is entirely statically secured. This limitation substantially weakens the benefits that dynamic enforcement brings to static security typing. Additionally, current proposals are focused on languages with explicit casts, and therefore do not fulfill the vision of gradual typing, according to which the boundaries between static and dynamic checking only arise from the (im)precision of type annotations, and are transparently mediated by implicit checks.

In this technical report we present the complete definitions and proofs of GSL_{\text{Ref}}, a gradual security-typed higher-order language with references. As a gradual language, GSL_{\text{Ref}} supports the range of static-to-dynamic security checking exclusively driven by type annotations, without resorting to explicit casts. Additionally, GSL_{\text{Ref}} lets programmers use types to reason statically about termination-insensitive noninterference in all programs, even those that enforce security dynamically. We prove that GSL_{\text{Ref}} satisfies all but one of Siek et al.’s criteria for gradually-typed languages, which ensure that programs can seamlessly transition between simple typing and security typing. A notable exception regards the dynamic gradual guarantee, which some specific programs must violate if they are to satisfy noninterference; it remains an open question whether such a language could fully satisfy the dynamic gradual guarantee. To realize this design, we were led to draw a sharp distinction between syntactic type safety and semantic type soundness, each of which constrains the design of the gradual language.

CCS Concepts: • Security and privacy → Information flow control; • Theory of computation → Type structures; Program semantics;

Additional Key Words and Phrases: Noninterference, language-based security, gradual typing
Contents

1 Overview 3

2 Full definitions for the static and gradual languages 3
2.1 SSLRef: Static semantics 3
2.2 SSLRef: Dynamic semantics 3
2.3 SSLRef: Noninterference definitions 4
2.4 GSLRef: Static semantics 7
2.4.1 Additional Definitions 7
2.5 GSLRef: Static semantics 9
2.6 GSLRef: Dynamic semantics 10
2.7 GSLRef: Translation to GSLRef 11
2.8 Noninterference definitions 13

3 Static Security Typing with References 20
3.1 SSLRef: Static type safety 20
3.2 SSLRef: Noninterference 29
3.3 Definitions 30
3.4 Proof of noninterference 30

4 Gradualizing the Static Semantics 39
4.1 From Gradual Labels to Gradual Types 39
4.2 Static Criteria for Gradual Typing 41

5 Gradualizing the Dynamic Semantics 46
5.1 Precise Evidence for Consistent Security Judgments 46
5.2 Initial evidence 48
5.3 Evolving evidence: Consistent Transitivity 48
5.4 Algorithmic definitions 49
5.4.1 Label Evidences 49
5.4.2 Type Evidences 50
5.4.3 Evidence inversion functions 52
5.5 Proofs 52

6 GSLRef: Dynamic properties 56
6.1 Intrinsic Terms: Static Semantics 56
6.2 Intrinsic Terms: Dynamic Semantics 57
6.3 Relating Intrinsic and Evidence-augmented Terms 58
6.4 Type Safety 62
6.5 Dynamic Gradual Guarantee 71
6.6 Noninterference 78
6.6.1 Definitions 78
6.6.2 Proof of noninterference 80
1 OVERVIEW

In this document we present the complete definitions and proofs of the static language SSL\textsubscript{Ref}, the gradual language GSL\textsubscript{Ref}, and the evidence augmented language GSL\textsubscript{ε}Ref. Section 2 presents the full definitions for the static and gradual languages. Section 3 presents the proof of type safety and noninterference for SSL\textsubscript{Ref}. Section 4 presents the proofs of soundness and optimality of the Galois connection, and the proof of the static gradual guarantee. Section 5 presents the formalization of evidences for GSL\textsubscript{Ref}: structure of evidence along with it corresponding Galois connection, initial evidence, evolving evidence (consistent transitivity), algorithmic definitions and their proofs. Section 6 present dynamic properties of GSL\textsubscript{ε}Ref. The presentation and proofs follows an intrinsic notation rather than evidence augmented notation, as it is more explicit (although more verbose). We present the proofs of type safety and noninterference, along the proof of the dynamic gradual guarantee for a similar gradual language that does not contain the extra dynamic check added in the runtime semantics.

2 FULL DEFINITIONS FOR THE STATIC AND GRADUAL LANGUAGES

In this section we present the full definition of SSL\textsubscript{Ref} (sections 2.1 and 2.2) and the full definition of GSL\textsubscript{Ref} (sections 2.4 and 2.6). Section 2.8 presents the full definitions of noninterference presented in the paper.

2.1 SSL\textsubscript{Ref}: Static semantics

In this section we present the full definition of the static semantics of SSL\textsubscript{Ref}. Figure 1 presents the syntax of SSL\textsubscript{Ref}. Figure 2 presents the complete static semantics of SSL\textsubscript{Ref}, where the join between types and labels is defined as follows

\begin{align*}
\text{valid}(\{\text{bool}_{\ell_i}\}) & \quad \text{valid}(\{\hat{S}_{i1}\}) \\
\text{valid}(\{\hat{S}_{i2}\}) & \quad \text{valid}(\{\Ref_{\ell_i} S_{i}\}) \\
\text{valid}(\{\Unit_{\ell_i}\})
\end{align*}

Figure 3 presents the join and meet type functions.

Definition 2.1 (Valid Type Sets).

2.2 SSL\textsubscript{Ref}: Dynamic semantics

In this section we present in Figure 4 the full definition of the dynamic semantics of SSL\textsubscript{Ref}.
\[ x : S \in \Gamma \]
\[ \Gamma, \Sigma; \ell_c + p_\ell : \text{Bool}_\ell \]
\[ \Gamma; \Sigma; \ell_c + \text{unit}_\ell : \text{Unit}_\ell \]

\[ o : S \in \Sigma \]
\[ \Gamma; \Sigma; \ell_c + o_\ell : \text{Ref}_\ell S \]

\[ \ell ; \Sigma; \ell_c \not\in \ell + t : S \]
\[ \Gamma; \Sigma; \ell_c + \text{prot}_\ell(t) : S \not\ell \]

\[ \ell ; \Sigma; \ell_c + t : \text{Bool}_\ell \]
\[ \Gamma; \Sigma; \ell_c + t : \text{Label}(S) \]

\[ \ell ; \Sigma; \ell_c + \text{ref}^S t : \text{Ref}_\ell S \]

\[ S \ll S \]
\[ \text{Bool} \ll \ell' \]
\[ \text{Unit} \ll \ell' \]

\[ S_1' \ll S_1 \quad S_2' \ll S_2' \quad \ell_1 \ll \ell_1' \quad \ell_2' \ll \ell_2 \]

\[ S_1 \xrightarrow{\ell_1} t_1 S_2 \ll S_1' \xrightarrow{\ell_1'} t_1 S_2' \]

\[ S \ll S \]

\[ \ell \ll \ell' \]

\[ \text{Ref}_\ell S \ll \text{Ref}_\ell \ell' \]

\[ \text{Fig. 2.} \text{SSL}_{\text{Ref}}: \text{Static Semantics} \]

### 2.3 SSL\textsubscript{Ref}: Noninterference definitions

In this section we present definitions and properties of noninterference for SSL\textsubscript{Ref}. Figure 5 presents the full definition of step-indexed logical relations. The proofs can be found in Appendix 3.4.

**Definition 2.2.** Let \( \rho \) be a substitution, \( \Gamma \) and \( \Sigma \) a type substitutions. We say that substitution \( \rho \) satisfy environment \( \Gamma \) and \( \Sigma \), written \( \rho \models \Gamma; \Sigma \), if and only if \( \text{dom}(\rho) = \Gamma \) and \( \forall x \in \text{dom}(\Gamma), \forall \ell_c, \Gamma; \Sigma; \ell_c + \rho(x) : S' \), where \( S' \ll \Gamma(x) \).

**Definition 2.3 (Related substitutions).** Tuples \( \langle \ell_1, \rho_1, \mu_1 \rangle \) and \( \langle \ell_2, \rho_2, \mu_2 \rangle \) are related on \( k \) steps, notation \( \Gamma; \Sigma + \langle \ell_1, \rho_1, \mu_1 \rangle \approx_{\ell_\alpha}^k \langle \ell_2, \rho_2, \mu_2 \rangle \), if \( \rho_1 \models \Gamma; \Sigma + \mu_1 \approx_{\ell_\alpha}^k \mu_2 \) and

\[ \forall x \in \Gamma. \Sigma + \langle \ell_1, \rho_1(x), \mu_1 \rangle \approx_{\ell_\alpha}^k \langle \ell_2, \rho_2(x), \mu_2 \rangle : \Gamma(x) \]
\[ S \cup S' \]

\( \forall : \text{TYPE} \times \text{TYPE} \rightarrow \text{TYPE} \)

\( \text{Bool} \cup \text{Bool}' = \text{Bool}(\forall \ell') \)

\( (\ell_1 \vdash \ell_2 \cup \ell_3) \vee (\ell_4 \vdash \ell_5 \cup \ell_6) = (\ell_1 \vdash \ell_2) \cup (\ell_3 \vdash \ell_4) \)

\( \text{Ref} \vdash \text{Ref}' \rightarrow S = \text{Ref}(\forall \ell') \)

\( S \cup S \) undefined otherwise

\( \hat{\vdash} : \text{TYPE} \times \text{TYPE} \rightarrow \text{TYPE} \)

\( \text{Bool} \hat{\vdash} \text{Bool}' = \text{Bool}(\forall \ell') \)

\( (\ell_1 \vdash \ell_2 \hat{\vdash} \ell_3) \vee (\ell_4 \vdash \ell_5 \hat{\vdash} \ell_6) = (\ell_1 \vdash \ell_2) \hat{\vdash} (\ell_3 \vdash \ell_4) \)

\( \text{Ref} \hat{\vdash} \text{Ref}' \rightarrow S = \text{Ref}(\forall \ell') \)

\( S \hat{\vdash} S \) undefined otherwise

---

**Proposition 2.5 (Security Type Soundness).** If \( \Gamma; \Sigma; \ell_c \vdash t : S' \) \( \implies \) \( \forall S, S' : S \leq S' \) \( \implies \) \( \Gamma; \Sigma; \ell_c \vdash t : S' \)
$\Sigma \vdash (\ell_1, v_1, \mu_1) \approx^k_{\ell_o} (\ell_2, v_2, \mu_2) : S \iff \ell_1 \approx^c_{\ell_o} \ell_2 \land \Sigma \vdash \mu_1 \approx^k_{\ell_o} \mu_2 \land \Sigma; \ell_1 \vdash v_1 : S'_1, S'_1 \ll : S,$

$\land \left( \text{obs}_{\ell_o} (\ell_1, S) \implies \text{obsRel}_{\Sigma, S}^{\Sigma, S}(\ell_1, v_1, \mu_1, \ell_2, v_2, \mu_2) \right)$

$\text{obsRel}_{\Sigma, S}^{\Sigma, S}(\ell_1, v_1, \mu_1, \ell_2, v_2, \mu_2) \iff (\text{rval}(v_1) = \text{rval}(v_2)) \quad \text{if } S \in \{\text{Bool}, \text{Unit}, \text{Ref}_g S'\}$

$\text{obsRel}_{\Sigma, S_1}^{\Sigma, S_2}(\ell_1, v_1, \mu_1, \ell_2, v_2, \mu_2) \iff \forall j \leq k. \forall \Sigma \subseteq \Sigma', \Sigma' \vdash (\ell_1, v_1, \mu_1') \approx^j_{\ell_o} (\ell_2, v_2, \mu_2) : S_1,$

$\Sigma' \vdash (\ell_1, v_1, \mu_1') \approx^j_{\ell_o} (\ell_2, v_2, \mu_2) : C(S_2 \sim g)$

$\Sigma \vdash (\ell_1, t_1, \mu_1) \approx^k_{\ell_o} (\ell_2, t_2, \mu_2) : C(S) \iff \ell_1 \approx^c_{\ell_o} \ell_2 \land \Sigma \vdash \mu_1 \approx^k_{\ell_o} \mu_2 \land \Sigma; \ell_1 \vdash t_1 : S'_1, S'_1 \ll : S, \forall j < k$

$(t_1 | \mu_1 \xrightarrow{\ell_i} t'_1 | \mu'_1 \implies \Sigma \subseteq \Sigma', \Sigma' + \mu'_1 \approx^k_{\ell_o} \mu'_2 \land (\text{irred}(t'_1) \implies \Sigma' \vdash (\ell_1, t'_1, \mu'_1) \approx^{k-j}_{\ell_o} (\ell_2, t'_2, \mu'_2) : S))$

$\Sigma \vdash \mu_1 \approx^k_{\ell_o} \mu_2 \iff \Sigma \vdash \mu_1 \land \forall \ell_1, \ell_1 \approx^c_{\ell_o} \ell_2, j < k, \forall o \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$

$\Sigma \vdash (\ell_1, \mu_1(o), \mu_1) \approx^j_{\ell_o} (\ell_2, \mu_2(o), \mu_2) : S(o)$

$\ell_1 \approx^c_{\ell_o} \ell_2 \iff \text{obs}_{\ell_o}(\ell_1) \lor \neg \text{obs}_{\ell_o}(\ell_1)$

$\mu_1 \sim \mu_2 \iff \text{dom}(\mu_1) \subseteq \text{dom}(\mu_2)$

$\text{obs}_{\ell_o}(\ell, S) \iff \text{obs}_{\ell_o}(\ell) \land \text{obs}_{\ell_o}(\text{label}(S))$

$\text{obs}_{\ell_o}(\ell) \iff \ell \ll^c_{\ell_o}$

Fig. 5. Security logical relations
Algorithmically:
Type concretization induces notions of precision and abstraction.

\[ g, g_c, g_r \in \text{GLabel}, \quad U \in \text{GType}, \quad x \in \text{Var}, \quad b \in \text{Bool}, \quad \oplus \in \text{BoolOp} \]
\[ l \in \text{Loc}, \quad t \in \text{GTerm}, \quad r \in \text{RawValue}, \quad v \in \text{Value} \]
\[ \Gamma \in \text{Var} \overset{\text{fin}}{\rightarrow} \text{GType}, \quad \Sigma \in \text{Loc} \overset{\text{fin}}{\rightarrow} \text{GType} \]

\[
\begin{align*}
U & ::= \text{Bool}_g | U \overset{d_e}{\rightarrow} g U | \text{Ref}_g U | \text{Unit}_g & \quad (\text{gradual types}) \\
g & ::= \ell | ? & \quad (\text{gradual labels}) \\
b & ::= \text{true} | \text{false} & \quad (\text{Booleans}) \\
r & ::= b | \lambda \ell x : U.t | \text{unit} | o & \quad (\text{base values}) \\
v & ::= r_g & \quad (\text{values}) \\
t & ::= v | t \cdot t | t \oplus t | \text{if } t \text{ then } t \text{ else } t & \quad (\text{terms}) \\
\oplus & ::= \land | \lor & \quad (\text{operations})
\end{align*}
\]

![Fig. 6. GSLRef: Syntax](image)

### 2.4 GSLRef: Static semantics

In this section we present the syntax and static semantics of GSLRef. The syntax of GSLRef is given in Figure 6 and is otherwise identical to that of SSLRef. Figure 7 presents the type system of GSLRef. Each typing rule is derived from a corresponding SSLRef rule (Figure 2) by lifting labels, types, predicates, and functions to their gradual counterparts. We also present some additional definitions needed in gradualizing SSLRef which are not included in the paper. Finally we present some example typing derivations in Figure 9.

#### 2.4.1 Additional Definitions.

**Definition 2.6 (Type Concretization).** \( \gamma_S : \text{GType} \rightarrow \mathcal{P}(\text{Type}) \)
\[
\gamma_S(\text{Bool}_g) = \{ \text{Bool}_\ell \mid \ell \in \gamma(g) \} \\
\gamma_S(\text{Unit}_g) = \{ \text{Unit}_\ell \mid \ell \in \gamma(g) \} \\
\gamma_S(\text{Ref}_g U) = \{ \text{Ref}_\ell S \mid \ell \in \gamma(g), S \in \gamma_S(U) \}
\]

Type concretization induces notions of precision and abstraction.

**Definition 2.7 (Type Precision).** \( U_1 \sqsubseteq U_2 \), if and only if \( \gamma_S(U_1) \subseteq \gamma_S(U_2) \).

**Definition 2.8 (Type Abstraction).** \( \alpha_S : \mathcal{P}(\text{Type}) \rightarrow \text{GType} \)
\[
\begin{align*}
\alpha_S(\overline{\text{Bool}_\ell_i}) &= \text{Bool}_{\alpha(\overline{\ell_i})} \\
\alpha_S(\overline{\text{Unit}_\ell_i}) &= \text{Unit}_{\alpha(\overline{\ell_i})} \\
\alpha_S(\overline{\text{Ref}_\ell_i S_{i_2}}) &= \text{Ref}_{\alpha(\overline{\ell_i}) \alpha_S(\overline{S_{i_2}})}
\end{align*}
\]

**Proposition 2.9 (\( \alpha_S \) is Sound and Optimal).** Assuming \( \hat{S} \) valid:
(i) \( \hat{S} \subseteq \gamma_S(\alpha_S(\hat{S})) \) (ii) If \( \hat{S} \subseteq \gamma_S(U) \) then \( \alpha_S(\hat{S}) \subseteq U \).

**Definition 2.10 (Gradual label meet).**
\[ g_1 \wedge g_2 = \alpha(\{ \ell_1 \land \ell_2 \mid (\ell_1, \ell_2) \in \gamma(g_1) \times \gamma(g_2) \}) \]

Algorithmically:
\[
\begin{align*}
\bot \wedge ? = ? \bot = \bot \\
g \wedge ? = ? g = ? \quad \text{if } g \neq \bot \\
\ell_1 \wedge \ell_2 = \ell_1 \land \ell_2
\end{align*}
\]
\[
\begin{align*}
(U_x) & \quad x : U \in \Gamma \\
(U_b) & \quad \Gamma ; \Sigma ; g_c \vdash x : U \\
(U_o) & \quad \alpha : U \in \Sigma \\
(U_{\text{prot}}) & \quad \Gamma ; \Sigma ; g_c \dashv g + t : U \\
(U_{\text{app}}) & \quad U_2 \leq U_{11} \quad g \land g_c \leq g_c' \\
(U_{\text{asgn}}) & \quad \Gamma ; \Sigma ; g_c \vdash t : U_1 \\
(U_{\text{ref}}) & \quad \Gamma ; \Sigma ; g_c \vdash \text{ref}^U t : \text{Ref}_U \\
(U_{\text{deref}}) & \quad \Gamma ; \Sigma ; g_c \vdash !t : U \lor g \\
\end{align*}
\]

Fig. 7. GSLRef: Static Semantics

\[
\begin{align*}
\overline{\mathbf{\nu}} : \text{TYPE} \times \text{TYPE} & \rightarrow \text{TYPE} \\
\overline{\text{Bool}_g} : \overline{\mathbf{\nu}} & = \overline{\text{Bool}_g'} \\
(U_1 \overline{g_c} \overline{g} U_{12}) \overline{\mathbf{\nu}} (U_2 \overline{g_c'} \overline{g} U_{22}) & = (U_1 \overline{\overline{\gamma}} U_{21}) \overline{g_c \overline{\gamma} g_c'} (U_2 \overline{\gamma} U_{22}) \\
\overline{\text{Ref}_g} U \overline{\nu} \overline{\text{Ref}_g'} U' & = \overline{\text{Ref}_g (g \overline{\gamma} g') U \cap U'} \\
U \overline{\nu} U & = \text{undefined otherwise} \\
\overline{\overline{\mathbf{\nu}} : \text{TYPE} \times \text{TYPE} & \rightarrow \text{TYPE} \\
\overline{\text{Bool}_g} : \overline{\overline{\mathbf{\nu}}} & = \overline{\text{Bool}_g'} (g \overline{\gamma} g') \\
(U_1 \overline{g_c} \overline{g} U_{12}) \overline{\overline{\gamma}} (U_2 \overline{g_c'} \overline{g} U_{22}) & = (U_1 \overline{\overline{\gamma}} U_{21}) \overline{g_c \overline{\gamma} g_c'} (U_2 \overline{\gamma} U_{22}) \\
\overline{\text{Ref}_g} U \overline{\overline{\nu}} \overline{\text{Ref}_g'} U' & = \overline{\text{Ref}_g (g \overline{\gamma} g') U \cap U'} \\
U \overline{\overline{\nu}} U & = \text{undefined otherwise}
\end{align*}
\]

Fig. 8. GSLRef: consistent join and consistent meet

**Definition 2.11** (Gradual label join). \( g_1 \overline{\nu} g_2 = \alpha(\{ \ell_1 \lor \ell_2 \mid (\ell_1, \ell_2) \in \gamma(g_1) \times \gamma(g_2) \}) \)

Algorithmically:
\[
\begin{align*}
\top \overline{\nu} ? = ? \overline{\nu} \top = \top \\
g \overline{\nu} ? = ? \overline{\nu} g = ? \text{ if } g \neq \top \\
\ell_1 \overline{\nu} \ell_2 = \ell_1 \lor \ell_2
\end{align*}
\]
Definition 2.12 (Label Meet). \( g_1 \sqcap g_2 = \alpha(\gamma(g_1) \cap \gamma(g_2)) \).
Algorithmically:
\[
g \sqcap g = g \\
g \sqcap ? = ? \sqcap g = g
\]

Definition 2.13 (Type Meet). \( U_1 \sqcap U_2 = \alpha_S(\gamma_S(U_1) \cap \gamma_S(U_2)) \).
Algorithmically:
\[
g \sqcap g' = g \sqcap g' \\
\text{Bool}_g \sqcap \text{Bool}_{g'} = g \sqcap g' \\
\text{Unit}_g \sqcap \text{Unit}_{g'} = g \sqcap g' \\
\text{Ref}_g U_1 \sqcap \text{Ref}_{g'} U_2 = g \sqcap g' U_1 \sqcap U_2
\]

Also, we introduce a function \( \text{label} \), which yields the security label of a given type:
\[
\text{label} : \text{GType} \rightarrow \text{Label}
\]
\[
\text{label}(\text{Bool}_g) = g \\
\text{label}(\text{Unit}_g) = g \\
\text{label}(U_1 \rightarrow g U_2) = g \\
\text{label}(\text{Ref}_g U) = g
\]

Definition 2.14 (Type Precision (inductive definition)).
\[
g_1 \sqsubseteq g_2 \\
\text{Bool}_{g_1} \sqsubseteq \text{Bool}_{g_2} \\
\text{Unit}_{g_1} \sqsubseteq \text{Unit}_{g_2} \\
U_{11} \sqsubseteq U_{21} \\
U_{12} \sqsubseteq U_{22} \\
g_1 \sqsubseteq g_2 \\
U_{1} \sqsubseteq U_{2} \\
\text{Ref}_{g_1} U_1 \sqsubseteq \text{Ref}_{g_2} U_2
\]

Definition 2.15 (Consistent label ordering (inductive definition)).
\[
? \lessdot g \\
g \lessdot g \\
\ell_1 \lessdot \ell_2 \\
\ell_1 \lessdot \ell_2
\]

Definition 2.16 (Consistent subtyping (inductive definition)).
\[
g \lessdot g' \\
\text{Bool}_g \leq \text{Bool}_{g'} \\
\text{Unit}_g \leq \text{Unit}_{g'} \\
\text{Ref}_g U \leq \text{Ref}_{g'} U \\
U_1' \leq U_1 \\
U_2 \leq U_2 \\
g_1 \lessdot g_1' \\
g_2 \lessdot g_2 \\
U_1 \rightarrow g_1 U_2 \leq U_1' \rightarrow g_1' U_2'
\]

2.5 \( \text{GSL}^{\varepsilon}_{\text{Ref}} \): Static semantics

In this section we present the full definition of the static semantics of \( \text{GSL}^{\varepsilon}_{\text{Ref}} \).

Definition 2.17 (Interval). An interval is a bounded unknown label \([\ell_1, \ell_2]\) where \( \ell_1 \) is the upper bound and \( \ell_2 \) is the lower bound.
\[
i \in \text{LABEL}^2 \\
i ::= [\ell, \ell] \quad \text{(interval)}
\]

Definition 2.18 (Evidence for labels).
\[
\varepsilon ::= (i, i)
\]
In this section we present the full definition of the dynamic semantics of GSL.

Definition 2.20 (Evidence for types).

Definition 2.19 (Type Evidence).

We extend the syntax of GSL with frames defined as follows:

2.6 GSL Ref: Dynamic semantics

In this section we present the full definition of the dynamic semantics of GSL.

We present the syntax of GSL Ref in Figure 10 and the static semantics in Figure 11.
The initial evidence function for consistent label ordering is presented in Figure 18. In this section we present the translation from terms of GSL into terms of GSLRef in Figure 17. The initial evidence function for consistent label ordering is presented in Figure 18. The initial evidence function for consistent subtyping is presented in Figure 19 using the following definition of operation pattern:

We present the complete dynamic semantics in Figure 12, and the evaluation frames and reduction in Figure 13. Auxiliary functions for evidence for labels is presented in Figure 14. Auxiliary functions for evidence for types is shown in Figure 15, and the inversion functions for evidence in Figure 16.

**2.7 GSLRef: Translation to GSLRef**

In this section we present the translation from terms of GSLRef into terms of GSLRef in Figure 17. The initial evidence function for consistent label ordering is presented in Figure 18. The initial evidence function for consistent subtyping is presented in Figure 19 using the following definition of operation pattern:

Every type rule has the extra judgment $\varepsilon \vdash g_c \sim g'_c$. 

Fig. 11. GSLRef: Static Semantics
(r1) $\varepsilon_1(b_1)g_1 \oplus \varepsilon_2(b_2)g_2 \mid \mu \xrightarrow{\varepsilon_{gc}} (\varepsilon_1 \cong \varepsilon_2)(b_1 \oplus b_2)_{(g_1 \searrow g_2)} \mid \mu$  
\[ \rightarrow: \mathbb{C} \times (\mathbb{C} \cup \text{error}) \]

(r2) $\text{prot}_{\varepsilon_1 b_1} \varepsilon_2 g_2 (\varepsilon_3 u) \mid \mu \xrightarrow{\varepsilon_{gc}} (\varepsilon_3 \cong \varepsilon_1)(u \searrow g_1) \mid \mu$

(r3) $\varepsilon_1(\lambda y'.x: U.t) \beta @ \varepsilon_2 u \mid \mu \xrightarrow{\varepsilon_{gc}} \begin{cases} \text{prot}_{iblb(\varepsilon_1)\varepsilon_2}(\varepsilon_1'(\varepsilon_2 x)) \mid \mu & \text{if } \varepsilon_1' \text{ or } \varepsilon_2' \text{ are not defined} \\ \text{error} & \text{if } \varepsilon_1' \text{ or } \varepsilon_2' \text{ are not defined} \end{cases}$
where:
\[ \varepsilon_1' = (\varepsilon \searrow \text{iblb}(\varepsilon_1)) \circ \varepsilon_3 \circ \text{ldom}(\varepsilon_1) \]
\[ \varepsilon_2' = \varepsilon_2 \circ \varepsilon_3 \circ \text{idom}(\varepsilon_1) \]
\[ g_1 = (g_c \searrow g) \]

(r4) if $\varepsilon_1 b_1$ then $t_2$ else $t_3$ $\mid \mu \xrightarrow{\varepsilon_{gc}} \begin{cases} \text{prot}_{iblb(\varepsilon_1)\varepsilon_2}(\varepsilon_1'(t_2)) \mid \mu & \text{if } b = \text{true} \\ \text{prot}_{iblb(\varepsilon_1)\varepsilon_2}(\varepsilon_1'(t_3)) \mid \mu & \text{if } b = \text{false} \end{cases}$
where:
\[ \varepsilon' = \varepsilon \searrow \text{iblb}(\varepsilon_1) \]
\[ g' = g_c \searrow g_1 \]

(r5) $\text{ref}_{\varepsilon_2} \varepsilon_1 u \mid \mu \xrightarrow{\varepsilon_{gc}} \begin{cases} \emptyset \mid \mu \lceil o \mapsto \varepsilon'(u \searrow g_c) \rceil & \text{if } (\varepsilon \circ \varepsilon_2) \text{ is not defined} \\ \text{error} & \text{if } (\varepsilon \circ \varepsilon_2) \text{ is not defined} \end{cases}$
where:
\[ o \notin \text{dom}(\mu) \]
\[ \varepsilon' = \varepsilon_1 \cong (\varepsilon \circ \varepsilon_2) \]

(r6) $!\varepsilon_1 o g \mid \mu \xrightarrow{\varepsilon_{gc}} \text{prot}_{iblb(\varepsilon_1)\varepsilon_2}(\varepsilon_1'(\text{iref}(\varepsilon_1)\nu))$
where:
\[ \mu(o) = \nu \]
\[ \varepsilon' = \varepsilon \searrow \text{iblb}(\varepsilon_1) \]
\[ g' = g_c \searrow g \]

(r7) $\varepsilon_1 o g := \varepsilon_3 \varepsilon_2 u \mid \mu \xrightarrow{\varepsilon_{gc}} \begin{cases} \text{unit} \mid \mu \lceil o \mapsto \varepsilon'(u \searrow (g_c \searrow g)) \rceil & \text{if } \varepsilon' \text{ is not defined, or } \varepsilon \mid \leq \text{iblb}(\varepsilon'') \text{ does not hold} \\ \text{error} & \text{if } \varepsilon' \text{ is not defined, or } \varepsilon \mid \leq \text{iblb}(\varepsilon'') \text{ does not hold} \end{cases}$
where:
\[ \mu(o) = \varepsilon' \]
\[ \varepsilon' = \varepsilon_2 \circ \varepsilon_3 \circ \text{iref}(\varepsilon_1) \searrow ((\varepsilon \searrow \text{iblb}(\varepsilon_1)) \circ \varepsilon_3 \circ \text{iblb}(\text{iref}(\varepsilon_1))) \]

$\varepsilon_1(\varepsilon_2 u) \rightarrow_{<'}: \begin{cases} (\varepsilon_2 \circ \varepsilon_3 \leq \varepsilon_1)u & \text{error} \\ \text{error} \end{cases}$

Fig. 12. GSRef: Dynamic semantics

**Definition 2.21 (Operation pattern).**

\[ P^T \in \text{GPATTERN}, P^\ell \in \text{LPATTERN} \]

\[ P^T ::= _\bot | P^T \oplus P^T \quad \text{(pattern on types)} \]

\[ \text{op}^T ::= \vee | \wedge | \top \quad \text{(operations on types)} \]

\[ P^\ell ::= _\bot | P^\ell \oplus P^\ell \quad \text{(pattern on labels)} \]

\[ \text{op}^\ell ::= \vee | \wedge | \top \quad \text{(operations on labels)} \]
2.8 Noninterference definitions

The formal definitions of related values and related computations are presented in Figures 20 and 21 respectively.

Definition 2.22 (Related substitutions). Tuples \( \langle g_1, p_1, \mu_1 \rangle \) and \( \langle g_2, p_2, \mu_2 \rangle \) are related on \( k \) steps under \( \Gamma, \Sigma \) and \( g_c \), notation \( \Gamma; \Sigma; g_c \vdash \langle g_1, p_1, \mu_1 \rangle \approx^k_{g_c} \langle g_2, p_2, \mu_2 \rangle \), if \( \rho_1 \models \Gamma, \Sigma \vdash \mu_1 \approx^k_{g_c} \mu_2 \) and

\[
\forall x \in \text{dom}(\Gamma). \Sigma; g_c \vdash \langle \hat{g}, p_1(x), \mu_1 \rangle \approx^k_{g_c} \langle \hat{g}, p_2(x), \mu_2 \rangle : \Gamma(x)
\]

Definition 2.23 (Semantic Security Typing).

\[
\Gamma; \Sigma; \hat{g} \vdash t : U \iff \forall \ell_0 \in \text{LABEL}, k \geq 0, p_1, p_2 \in \text{SUBST} \text{ and } \mu_1, \mu_2 \in \text{STORE}, \forall g_c, \hat{g} = \varepsilon g, \varepsilon \vdash g \not\approx g_c, \text{ such that } \Sigma \vdash \mu_1 \text{ and } \Gamma; \Sigma; g_c \vdash \langle \hat{g}, p_1(t), \mu_1 \rangle \approx^k_{g_c} \langle \hat{g}, p_2(t), \mu_2 \rangle : C(U)
\]

Proposition 2.24 (Security Type Soundness). \( \Gamma; \Sigma; \hat{g} \vdash t : U \implies \Gamma; \Sigma; \hat{g} \vdash t : U \)
\begin{center}
\begin{tabular}{|c|c|}
\hline
 BOOL \cap \text{BOOL}^\prime &= \text{REF} \cap \text{REF}^\prime \\
\hline
(E_{11} \xrightarrow{i_1} (E_{12}) \cap (E_{21} \xrightarrow{i_1'} (E_{22})) = (E_{11} \cap E_{21}) \xrightarrow{i_1 \cap i_1'} (E_{12} \cap E_{22}) & E \cap E' \text{ undefined otherwise} \\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{|c|c|}
\hline
\text{BOOL} \vdash i_2 \quad \text{BOOL} \vdash \neg i_2 &= \text{BOOL} \vdash (i_1 \neg i_2) \\
\hline
E_1 \xrightarrow{i_2} i_2 E_2 \neg i_3 = E_1 \xrightarrow{i_2} (i_1 \neg i_3) E_2 & \text{REF} \vdash E \neg i_2 = \text{REF} \vdash (i_1 \neg i_3) E \\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{|c|c|}
\hline
\text{BOOL} \vdash i_2 \quad \text{BOOL} \vdash \neg i_2 &= \text{BOOL} \vdash (i_1 \neg i_2) \\
\hline
E_1 \xrightarrow{i_2} i_2 E_2 \neg i_3 = E_1 \xrightarrow{i_2} (i_1 \neg i_3) E_2 & \text{REF} \vdash E \neg i_2 = \text{REF} \vdash (i_1 \neg i_3) E \\
\hline
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{|c|c|}
\hline
\text{BOOL} \vdash \neg i_2 \quad \text{BOOL} \vdash \neg \neg i_2 &= \text{BOOL} \vdash (i_1 \neg i_2) \\
\hline
E_1 \xrightarrow{i_2} \neg i_2 E_1 \neg i_2 E_2 \neg i_3 = E_1 \neg i_2 E_1 \neg E_2 \neg i_3 & \text{REF} \vdash E \neg i_2 = \text{REF} \vdash (i_1 \neg i_2) E \\
\hline
\end{tabular}
\end{center}

\begin{align*}
\Delta^\prec (t_1, t_2, t_3) &= \langle t'_1, t'_2, t'_3 \rangle \\
\Delta^\prec \text{BOOL}_{i_1, i_2, i_3} &= \langle \text{BOOL}_{i'_1, i'_2} \rangle
\end{align*}

\begin{align*}
\Delta^\prec (E_{31}, E_{21}, E_{11}) &= \langle E'_{31}, E'_{11} \rangle & \Delta^\prec (E_{12}, E_{22}, E_{32}) &= \langle E'_{12}, E'_{32} \rangle \\
\Delta^\prec (t_1, t_2, t_3) &= \langle t'_1, t'_2, t'_3 \rangle & \Delta^\prec (t_{13}, t_{12}, t_{11}) &= \langle t'_{13}, t'_{12}, t'_{11} \rangle
\end{align*}

\begin{align*}
\Delta^\prec (E_{11} \xrightarrow{i_1} (E_{12} \xrightarrow{i_2} (E_{21} \xrightarrow{i_3} E_{32}))) &= \langle E'_{11} \xrightarrow{i'_1} (E'_{12} \xrightarrow{i'_2} (E'_{21} \xrightarrow{i'_3} E'_{32}))) \\
\Delta^\prec (\text{REF} \vdash i_1, \text{REF} \vdash i_2, \text{REF} \vdash i_3) &= \langle \text{REF} \vdash i'_1, \text{REF} \vdash i'_2, \text{REF} \vdash i'_3 \rangle
\end{align*}

\begin{align*}
\langle E_{1}, E_{21} \rangle \circ^\prec \langle E_{22}, E_{3} \rangle &= \Delta^\prec (E_{1}, E_{21} \cap E_{22}, E_{3})
\end{align*}

Fig. 15. GSL\textsuperscript{\text{-}Ref} Auxiliary functions for the dynamic semantics (Types)

**Proof.** Proof in Appendix 6. \hfill $\square$
\[ \text{ilbl}(\langle \text{Bool}_{i_1}, \text{Bool}_{i_2} \rangle) = \langle i_1, i_2 \rangle \]
\[ \text{ilbl}(\langle \text{Unit}_{i_1}, \text{Unit}_{i_2} \rangle) = \langle i_1, i_2 \rangle \]
\[ \text{ilbl}(\langle \text{Ref}_{i_1} U_1, \text{Ref}_{i_2} U_2 \rangle) = \langle i_1, i_2 \rangle \]
\[ \text{ilbl}(\langle E_1 \xrightarrow{i_2} E_2, E'_1 \xrightarrow{i'_2} E'_2 \rangle) = \langle i_1, i'_1 \rangle \]

\[ \text{iref}(\langle \text{Ref}_{i_1} E_1, \text{Ref}_{i_2} E_2 \rangle) = \langle E_1, E_2 \rangle \]
\[ \text{iref}(\langle E_1, E_2 \rangle) = \text{undefined otherwise} \]

\[ \text{idom}(\langle E_1 \xrightarrow{i_2} E_2, E'_1 \xrightarrow{i'_2} E'_2 \rangle) = \langle E'_1, E_1 \rangle \]
\[ \text{idom}(\langle E_1, E_2 \rangle) = \text{undefined otherwise} \]

\[ \text{icod}(\langle E_1 \xrightarrow{i_2} E_2, E'_1 \xrightarrow{i'_2} E'_2 \rangle) = \langle E_2, E'_2 \rangle \]
\[ \text{icod}(\langle E_1, E_2 \rangle) = \text{undefined otherwise} \]

Fig. 16. GSL\textsubscript{Ref}ε: Inversion functions for evidence
Fig. 17. GSLref: translation to GSL$^e_{\text{Ref}}$ terms
\[
\begin{align*}
\text{bounds}(\bot) &= [\bot, \top] \\
\text{bounds}(\ell) &= [\ell, \ell] \\
\text{bounds}(x_1 \lor x_2) &= \text{bounds}(x_1) \lor \text{bounds}(x_2) \\
\text{bounds}(x_1 \land x_2) &= \text{bounds}(x_1) \land \text{bounds}(x_2) \\
\text{bounds}(x_1 \cap x_2) &= \text{bounds}(x_1) \cap \text{bounds}(x_2) \\
\text{bounds}(F_1(\overline{x} \lor F_2(\overline{x})) &= \text{bounds}(F_1(\overline{x})) \lor \text{bounds}(F_2(\overline{x})) \\
\text{bounds}(F_1(\overline{x} \land F_2(\overline{x})) &= \text{bounds}(F_1(\overline{x})) \land \text{bounds}(F_2(\overline{x})) \\
\text{bounds}(F_1(\overline{x}) \cap F_2(\overline{x})) &= \text{bounds}(F_1(\overline{x})) \cap \text{bounds}(F_2(\overline{x})) \\
\end{align*}
\]

\[
\text{bounds}(F_1(\overline{g})) = [\ell_1, \ell_2] \quad \text{bounds}(F_2(\overline{f})) = [\ell'_1, \ell'_2]
\]

\[
\mathcal{G}(F_1(g_1, \ldots, g_n) \preceq F_2(g_{n+1}, \ldots, g_{n+m})) = \langle [\ell_1, \ell_2 \land \ell'_2], [\ell_1 \lor \ell'_1, \ell'_2] \rangle
\]

where \(F_1 : \text{GLabel}^n \rightarrow \text{GLabel}\) and \(F_2 : \text{GLabel}^m \rightarrow \text{GLabel}\).

\[
\mathcal{G}(\overline{g}(g_1, \ldots, g_n)) = \mathcal{G}(\overline{f}(g_1, \ldots, g_n) \preceq (g_1, \ldots, g_n))
\]

Fig. 18. \(\text{GSLe}^{\text{Ref}}\): Initial evidence for gradual labels
\[ \text{lift}_{P}(...) = \_ \]
\[ \text{lift}_{P}(P_{1} \lor P_{2}) = \text{lift}_{P}(P_{1}) \lor \text{lift}_{P}(P_{2}) \]
\[ \text{lift}_{P}(P_{1} \land P_{2}) = \text{lift}_{P}(P_{1}) \land \text{lift}_{P}(P_{2}) \]
\[ \text{invert}(...) = \_ \]
\[ \text{invert}(P_{1} \lor P_{2}) = \text{invert}(P_{1}) \lor \text{invert}(P_{2}) \]
\[ \text{invert}(P_{1} \land P_{2}) = \text{invert}(P_{1}) \land \text{invert}(P_{2}) \]
\[ \text{tomeet}(...) = \_ \]
\[ \text{tomeet}(P_{1} \lor P_{2}) = \text{tomeet}(P_{1}) \lor \text{tomeet}(P_{2}) \]
\[ \text{tomeet}(P_{1} \land P_{2}) = \text{tomeet}(P_{1}) \land \text{tomeet}(P_{2}) \]

\[ \mathcal{G}[\text{lift}_{P}(G_{1})(\bar{t}_{1}) <: \text{lift}_{P}(G_{2})(\bar{t}_{j})] = \langle t_{1}, t_{2} \rangle \]
\[ \mathcal{G}[G_{1}(\text{Bool}_{g_{1}}) \ll G_{2}(\text{Bool}_{g_{j}})] = \langle \text{Bool}_{t_{1}}, \text{Bool}_{t_{2}} \rangle \]
\[ \mathcal{G}[\text{invert}(G_{2})(\bar{u}_{j1}) <: \text{invert}(G_{1})(\bar{u}_{i1})] = \langle \text{E}_{21}, \text{E}_{11} \rangle \]
\[ \mathcal{G}[\text{lift}_{P}(G_{1})(\bar{u}_{i1}) <: \text{lift}_{P}(G_{2})(\bar{u}_{j1})] = \langle t_{11}, t_{12} \rangle \]
\[ \mathcal{G}[\text{lift}_{P}(\text{invert}(G_{1}))(\bar{t}_{j2}) <: \text{lift}_{P}(\text{invert}(G_{1}))(\bar{t}_{j2})] = \langle t_{22}, t_{21} \rangle \]
\[ \mathcal{G}[G_{1}(U_{1} \rightarrow_{g_{1}} U_{2}) <: G_{2}(U_{1} \rightarrow_{g_{j}} U_{j2})] = \langle E_{11} \rightarrow t_{11}, E_{12} \rightarrow t_{12}, E_{21} \rightarrow t_{12}, E_{22} \rangle \]
\[ \mathcal{G}[\text{tomeet}(G_{1})(\bar{u}_{i1}) <: \text{tomeet}(G_{2})(\bar{u}_{j1})] = \langle E_{1}, E_{2} \rangle \]
\[ \mathcal{G}[\text{tomeet}(G_{2})(\bar{u}_{j1}) <: \text{tomeet}(G_{1})(\bar{u}_{i1})] = \langle E_{2}', E_{1}' \rangle \]
\[ \mathcal{G}[G_{1}(\text{Ref}_{g_{1}})(\bar{u}) <: G_{2}(\text{Ref}_{g_{j}})(\bar{u})] = \langle \text{Ref}_{t_{1}}, E_{1} \cap E_{1}', \text{Ref}_{t_{2}}, E_{2} \cap E_{2}' \rangle \]

where \( G_{1} : \text{GLABEL}^{n} \rightarrow \text{GLABEL} \) and \( G_{2} : \text{GLABEL}^{m} \rightarrow \text{GLABEL} \), and \( G_{1}(x_{1}, \ldots, x_{n}) = P_{1}^{T}(x_{1}, \ldots, x_{n}) \), \( G_{2}(x_{1}, \ldots, x_{n}) = P_{2}^{T}(x_{1}, \ldots, x_{n}) \).

\[ \mathcal{G}_{\text{Ref}}(F(\bar{u}_{1}, ..., \bar{u}_{n})) = \mathcal{G}[F(\bar{u}_{1}, ..., \bar{u}_{n}) <: F(\bar{u}_{1}, ..., \bar{u}_{n})] \]

Fig. 19. GSL\(_{\text{Ref}}^{\ell} \): Initial evidence for gradual types
\[
\Sigma; g_c \vdash (\hat{g}_1, v_1, \mu_1) \approx^{k}_{\ell_o} (\hat{g}_2, v_2, \mu_2) : U \iff g_c \vdash \hat{g}_1 \approx^{\ell_o} \hat{g}_2 \land \Sigma \vdash \mu_1 \approx^{k}_{\ell_o} \mu_2 \land \forall \nu \in \{\text{Bool}_g, \text{Unit}_g, \text{Ref}_g U'\}
\]

\[
\text{obsRel}_{\Sigma; g_c, U} (\hat{g}_1, v_1, \mu_1, \hat{g}_2, v_2, \mu_2) \iff \text{rval}(v_1) = \text{rval}(v_2)
\]

\[
\text{obsRel}_{\Sigma; g_c, U} (\hat{g}_1, v_1, \mu_1, \hat{g}_2, v_2, \mu_2) \iff \forall j \leq k, \forall U' = U_1 \rightarrow^g U_2, U_1', U_1''.
\]

\[
\forall g_c', \forall g_i', \text{where } g_i' \vdash g_i' \not\preceq g_c', \text{ s.t. } \hat{g}_i \leq^{\ell_o} \hat{g}_i', \epsilon_{11} \vdash U_1 \rightarrow g_1 U_2 \leq U', \epsilon_{12} \vdash U_1'' \leq U', \text{ and } \epsilon_{34} \vdash g_c \lor g_3 \leq g_{32}.
\]

\[
\forall v_i', \mu_i', \Sigma', \Sigma' \subseteq \Sigma', \Sigma' ; g_c \vdash (\hat{g}_1, v_1, \mu_1) \approx^{\ell_o} (\hat{g}_2, v_2, \mu_2) : U_1', \text{ dom}(\mu_i) \leq \text{ dom}(\mu'_i), \Sigma'; g_c \vdash (\hat{g}_1, (v_{11} v_1) \ominus_{\epsilon_{31}} v_{12} v_2), \mu_1') \approx^{\ell_o} (\hat{g}_2, (v_{12} v_2) \ominus_{\epsilon_{32}} v_{11} v_1), \mu_2') : C(U'_1 \not\rightarrow g_{31})
\]

Fig. 20. Related values

\[
\Sigma; g_c \vdash (\hat{g}_1, t_1, \mu_1) \approx^{k}_{\ell_o} (\hat{g}_2, t_2, \mu_2) : C(U) \iff g_c \vdash \hat{g}_1 \approx^{\ell_o} \hat{g}_2 \land \Sigma \vdash \mu_1 \approx^{k}_{\ell_o} \mu_2 \land \forall \nu \not\vdash \hat{g}_1 \approx^{\ell_o} \hat{g}_2 \text{ and}
\]

\[
\Sigma; \hat{g}_1' \vdash t_1 : U, \forall j < k, (t_i \mid \mu_1) \rightarrow^{t_i} \downarrow t_i' \mid \mu_i' \iff \exists \Sigma', \Sigma \subseteq \Sigma'.
\]

\[
\Sigma' \vdash \mu_1' \approx^{k-j}_{\ell_o} \mu_2' \land ((\text{irred}(t_i') \land \text{irred}(t_2'))) \iff \Sigma'; g_c \vdash (\hat{g}_1, t_1, \mu_1') \approx^{k-j}_{\ell_o} (\hat{g}_2, t_2, \mu_2') : U)
\]

Fig. 21. Related computations
3 STATIC SECURITY TYPING WITH REFERENCES

In this section we present the proof of type preservation for SSLRef in Sec. 3.1, and the definitions and proof of noninterference for SSLRef in Sec. 3.2.

3.1 SSLRef: Static type safety

In this section we present the proof of type safety for SSLRef.

Definition 3.1 (Well typeness of the store). A store μ is said to be well typed with respect to a typing context Γ and a store typing Σ, written Γ; Σ ⊢ μ, if dom(μ) = dom(Σ) and ∀o ∈ dom(μ), Γ; Σ ⊢ μ(o) : S and S ≺: Σ(o).

Lemma 3.2. If Γ; Σ; ℓc ⊢ t : S then ∀ℓ′ ∈ ℓc, Γ; Σ; ℓ′ ⊢ t : S.

Proof. By induction on the derivation of Γ; Σ; ℓc ⊢ t : S. Noticing that none of the inferred types of the type rules depend on ℓc.

Case (Sx, Sb, Su, Sl). Trivial because neither the premises and the inferred type depend on the security effect.

Case (S⊕). Then t = b₁ℓ₁ ⊕ b₂ℓ₂ and

\[
\begin{align*}
& (Sb) \quad Γ; Σ; ℓ_c ⊢ b₁ℓ₁ : Bool_{ℓ₁} \\
& (Sb) \quad Γ; Σ; ℓ_c ⊢ b₂ℓ₂ : Bool_{ℓ₂} \\
& (S⊕) \quad Γ; Σ; ℓ_c ⊢ b₁ℓ₁ ⊕ b₂ℓ₂ : Bool_{(ℓ₁ ∨ ℓ₂)}
\end{align*}
\]

Suppose ℓ′ ∈ ℓc such that ℓ′ ≺ ℓc, then by induction hypotheses on the premises:

\[
\begin{align*}
& (Sb) \quad Γ; Σ; ℓ′ ⊢ b₁ℓ₁ : Bool_{ℓ₁} \\
& (Sb) \quad Γ; Σ; ℓ′ ⊢ b₂ℓ₂ : Bool_{ℓ₂} \\
& (S⊕) \quad Γ; Σ; ℓ′ ⊢ b₁ℓ₁ ⊕ b₂ℓ₂ : Bool_{(ℓ′ ∨ ℓ₂)}
\end{align*}
\]

where ℓ′₁ = ℓ₁ and ℓ′₂ = ℓ₂ and the result holds.

Case (Sprot). Then t = protℓ(t) and

\[
\begin{align*}
& (Sprot) \quad Γ; Σ; ℓ_c ∨ ℓ ≺ τ : S \\
& \quad Γ; Σ; ℓ_c ⊢ protℓ(t) : S ∨ ℓ
\end{align*}
\]

Suppose ℓ′ ∈ ℓc such that ℓ′ ≺ ℓc. Considering that ℓ′ ≺ ℓ ≺ ℓc ∨ ℓ, then by induction hypotheses on the premise:

\[
\begin{align*}
& (Sprot) \quad Γ; Σ; ℓ′ ⊢ ℓ ≺ τ : S \\
& \quad Γ; Σ; ℓ′ ⊢ protℓ(t) : S ∨ ℓ
\end{align*}
\]

and therefore the result holds.

Case (Sapp). Then t = t₁ t₂ and

\[
\begin{align*}
& (Sa₁) \quad D₁ \\
& \quad Γ; Σ; ℓ_c ⊢ t₁ : S₁₁ → ℓ″₁ \rightarrow ℓ₁ \rightarrow S₁₁ \\
& \quad Γ; Σ; ℓ_c ⊢ t₁ t₂ : S₂ \quad ℓ_c ≺ ℓ′₂ ≺ ℓ″₂ \quad S₂ ≺: S₁₁ \\
& (Sapp) \quad Γ; Σ; ℓ_c ⊢ t₁ t₂ : S₁₂ ∨ ℓ
\end{align*}
\]
Suppose \( \ell' \) such that \( \ell' \preceq \ell \). Then by using induction hypotheses on the premises, considering
\[
S_1 \xrightarrow{\ell''} S_2 < S_1 \xrightarrow{\ell''} S_1 \text{ and } S_2 < S_2.
\]
As \( S_2 < S_1 \) and \( S_1 < S_1' \) then \( S_2' < S_1' \). Also, by definition of the join operator \( \ell' \lor \ell' \preceq \ell \lor \ell \preceq \ell'' \lor \ell'' \), and then:
\[
\begin{align*}
\text{(Sl) } & \frac{D_1}{\Gamma; \Sigma; \ell' + t_1 : S_1} \\
\text{(Sapp) } & \frac{D_2}{\Gamma; \Sigma; t_1 \rightarrow \ell' : S_1', \ell' \lor \ell' \preceq \ell'' \lor \ell''} \\
\end{align*}
\]
Where \( S_1 \lor \ell \preceq S_2 \lor \ell \) and the result holds.

**Case (Sif-true).** Then \( t = \text{true} \) then \( t_1 \) else \( t_2 \) and
\[
\begin{align*}
\text{(Sif) } & \frac{D_0}{\Gamma; \Sigma; t : \text{true} : \text{Bool}_\ell} \\
\text{(Sif) } & \frac{D_1}{\Gamma; \Sigma; \ell \lor t : S_1} \\
\end{align*}
\]
Suppose \( \ell' \) such that \( \ell' \preceq \ell \). As \( \ell' \lor \ell \preceq \ell \lor \ell \), by induction hypotheses in the premises:
\[
\begin{align*}
\text{(Sif) } & \frac{D_0}{\Gamma; \Sigma; t' : \text{true} : \text{Bool}_\ell} \\
\text{(Sif) } & \frac{D_1}{\Gamma; \Sigma; \ell' \lor t' : S_1'} \\
\end{align*}
\]
where \( S_1' = S_1, S_2' = S_2 \). Then \( (S_1' \lor S_2') \lor \ell = (S_1 \lor S_2) \lor \ell \) and therefore the result holds.

**Case (Sif-false).** Analogous to case (if-true).

**Case (Sref).** Then \( t = \text{ref}^S v \) and
\[
\begin{align*}
\text{(Sref) } & \frac{\Gamma; \Sigma; \ell' + v : S' \quad S' < S \quad \ell \preceq \text{label}(S)}{\Gamma; \Sigma; \ell' + \text{ref}^S v : \text{Ref}_\ell^S} \\
\end{align*}
\]
Suppose \( \ell' \) such that \( \ell' \preceq \ell \). By using induction hypotheses in the premise, considering \( \ell' \preceq \ell \preceq \text{label}(S) \):
\[
\begin{align*}
\text{(Sref) } & \frac{\Gamma; \Sigma; \ell' + v : S' \quad S' < S \quad \ell' \preceq \text{label}(S)}{\Gamma; \Sigma; \ell' + \text{ref}^S v : \text{Ref}_\ell^S} \\
\end{align*}
\]
and the result holds.

**Case (Sderef).** Then \( t = \text{!a}_\ell \) and
\[
\begin{align*}
\text{(Sderef) } & \frac{\alpha : S \in \Sigma}{\Gamma; \Sigma; \ell' + \alpha_\ell : \text{Ref}_\ell^S} \\
\text{(Sderef) } & \frac{\alpha : S \in \Sigma}{\Gamma; \Sigma; \ell' + \text{!a}_\ell : S \lor \ell'} \\
\end{align*}
\]
Suppose \( \ell' \) such that \( \ell' \preceq \ell \), then by using induction hypotheses in the premise:
\[
\begin{align*}
\text{(Sderef) } & \frac{\alpha : S \in \Sigma}{\Gamma; \Sigma; \ell' + \alpha_{\ell'} : \text{Ref}_{\ell'}^S} \\
\text{(Sderef) } & \frac{\alpha : S \in \Sigma}{\Gamma; \Sigma; \ell' + \text{!a}_{\ell'} : S \lor \ell'} \\
\end{align*}
\]
where \( \ell' = \ell \). and the result holds.
Case (Sassgn). Then \( t = o_\ell := v \) and

\[
\begin{array}{c}
o : S \in \Sigma \\
\hline
\Gamma; \Sigma, o_\ell \vdash o_\ell : \text{Ref}_\ell S \\
S_2 <: S \\
\hline
\Gamma; \Sigma, o_\ell \vdash v : S_2 \\
\ell_\ell \vdash \ell \ll \text{label}(S) \\
\hline
\Gamma; \Sigma, o_\ell \vdash v : \text{Unit}_\ell \\
\end{array}
\]

(Sassgn)

Suppose \( \ell_\ell' \) such that \( \ell_\ell' \ll \ell_\ell \). Considering that \( \ell_\ell' \vdash \ell \ll \ell \ll \text{label}(S) \), and \( S'_2 <: S_2 <: S \), then:

\[
\begin{array}{c}
o : S \in \Sigma \\
\hline
\Gamma; \Sigma, \ell_\ell' \vdash o_\ell : \text{Ref}_\ell S \\
S'_2 <: S \\
\hline
\Gamma; \Sigma, \ell_\ell' \vdash v : S'_2 \\
\ell_\ell' \vdash \ell \ll \text{label}(S) \\
\hline
\Gamma; \Sigma, \ell_\ell' \vdash v : \text{Unit}_\ell \\
\end{array}
\]

(Sassgn)

but

\[\text{Unit}_\ell <: \text{Unit}_\ell\]

and therefore the result holds.

Case (S::). Then \( t = v :: S \) and

\[
\begin{array}{c}
\hline
\Gamma; \Sigma, \ell_\ell' \vdash v : S_1 \\
S_1 <: S \\
\hline
\Gamma; \Sigma, \ell_\ell' \vdash v :: S : S \\
\end{array}
\]

(S::)

Suppose \( \ell_\ell' \) such that \( \ell_\ell' \ll \ell_\ell \), then by Lemma 3.4

\[
\begin{array}{c}
\hline
\Gamma; \Sigma, \ell_\ell' \vdash v : S_1 \\
S_1 <: S \\
\hline
\Gamma; \Sigma, \ell_\ell' \vdash v :: S : S \\
\end{array}
\]

(S::)

and the result holds.

\[\Box\]

Lemma 3.3 (Substitution). If \( \Gamma, x : S_1 ; \Sigma, \ell_\ell' \vdash t : S \) and \( \Gamma, \Sigma, \ell_\ell \vdash v : S'_1 \) such that \( S'_1 <: S_1 \), then \( \Gamma, \Sigma; \ell_\ell' \vdash [v/x]t : S' \) such that \( S' <: S \).

Proof. By induction on the derivation of \( \Gamma, x : S_1 ; \Sigma, \ell_\ell \vdash t : S \).

\[\Box\]

Lemma 3.4. If \( \Gamma, \Sigma, \ell_\ell \vdash v : S \) then \( \forall \ell_\ell' \), \( \Gamma, \Sigma, \ell_\ell' \vdash v : S \).

Proof. By induction on the derivation of \( \Gamma, \Sigma, \ell_\ell \vdash v : S \) observing that for values, there is no premise that depends on \( \ell_\ell \).

\[\Box\]

Proposition 3.5 (\( \to \) is well defined). If \( ; ;, \Sigma, \ell_\ell \vdash t : S \), \( ; ;, \Sigma, \mu \) and \( \forall \ell_\ell \), such that \( \ell_\ell <: \ell_\ell \), \( t \mid \mu \ell_\ell \to t' \mid \mu' \) then, for some \( \Sigma' \supseteq \Sigma \), \( ; ;, \Sigma', \ell_\ell' \vdash t' : S' \), where \( S' <: S \) and \( ; ;, \Sigma' \vdash \mu' \).

Proof.

Case (S@). Then \( t = b_1 \ell_1 \oplus b_2 \ell_2 \) and

\[
\begin{array}{c}
\hline
; ;, \Sigma, \ell_\ell \vdash b_1 \ell_1 : \text{Bool}_{\ell_1} \\
\hline
; ;, \Sigma, \ell_\ell \vdash b_2 \ell_2 : \text{Bool}_{\ell_2} \\
\hline
; ;, \Sigma, \ell_\ell \vdash b_1 \ell_1 \oplus b_2 \ell_2 : \text{Bool}_{(\ell_1 \lor \ell_2)} \\
\end{array}
\]

(S@)
Suppose $\ell_r$ such that $\ell_r \ll \ell_c$, then

$$
\frac{\begin{array}{c}
 b_1\ell_{t_1} \oplus b_2\ell_{t_2} \mid \mu \\
 \end{array}}{
 \ell_r \rightarrow (b_1 \oplus b_2)(\ell_{t_1} \lor \ell_{t_2}) \mid \mu }
$$

Then

$$
\frac{\begin{array}{c}
 (S\oplus) & \\
 \ell_c \vdash (b_1 \oplus b_2)(\ell_{t_1} \lor \ell_{t_2}) : \text{Bool}(\ell_{t_1} \lor \ell_{t_2}) \\
\end{array}}{
 \ell_r \rightarrow (S)_{\oplus} 
}$$

**Case (Sprot).** Then $t = \text{prot}_\ell(v)$ and

$$
\frac{\begin{array}{c}
 \vdots; \Sigma; \ell_c \lor \ell + v : S \\
\end{array}}{
 \vdots; \Sigma; \ell_c \vdash \text{prot}_\ell(v) : S \lor \ell }
$$

Suppose $\ell_r$ such that $\ell_r \ll \ell_c$, then

$$
\vdots; \Sigma; \ell_c \vdash v : S \lor \ell
$$

and the result holds.

**Case (Sapp).** Then $t = (\lambda x : S_{11}, t)_{\ell} v$ and

$$
\frac{\begin{array}{c}
 \vdots; x : S_{11}; \Sigma; \ell_c^\ell \vdash t : S_{12} \\
\end{array}}{
 \vdots; \Sigma; \ell_c \vdash (\lambda x : S_{11}, t)_{\ell} : S_{11} \rightarrow S_{12} }
$$

$$
\frac{\vdots; \Sigma; \ell_c \vdash v : S_2}{
 \vdots; \Sigma; \ell_c \vdash (\lambda x : S_{11}, t)_{\ell} v : S_{12} \lor \ell }
$$

Suppose $\ell_r$ such that $\ell_r \ll \ell_c$, and

$$
(\lambda x : S_{11}, t)_{\ell} v \mid \mu \quad \ell_r \rightarrow \text{prot}_\ell([v/x]t) \mid \mu
$$

But as $\ell_c \lor \ell \ll \ell_c^\ell$ then by Lemma 3.2, $\vdots; \Sigma; \ell_c \lor \ell + t : S_{12}', where S_{12}' \ll S_{12}$.

By Lemma 3.3 and Lemma 3.4, $\vdots; \Sigma; \ell_c \lor \ell + [v/x]t : S_{12}'\lor$, where $S_{12}'\lor \ll S_{12}\lor$.

Then

$$
\frac{\vdots; \Sigma; \ell_c \lor \ell + [v/x]t : S_{12}'\lor}{
 \vdots; \Sigma; \ell_c \vdash \text{prot}_\ell([v/x]t) : S_{12}'\lor}
$$

Where $S_{12}'\lor \ll S_{12}\lor \ell$ and the result holds.

**Case (Sif-true).** Then $t = \text{if true}_\ell$ then $t_1$ else $t_2$ and

$$
\frac{\begin{array}{c}
 \vdots; \Sigma; \ell_c \vdash \text{true}_\ell : \text{Bool}_\ell \\
\end{array}}{
 \vdots; \Sigma; \ell_c \lor \ell + t_1 : S_{1} }
$$

$$
\frac{\begin{array}{c}
 \vdots; \Sigma; \ell_c \lor \ell + t_2 : S_{2} \\
\end{array}}{
 \vdots; \Sigma; \ell_c \lor \ell + \text{if true}_\ell \then t_1 \else t_2 : (S_{1} \lor S_{2}) \lor \ell }
$$

Suppose $\ell_r$ such that $\ell_r \ll \ell_c$, then if

$$
\text{if true}_\ell \then t_1 \else t_2 \mid \mu \quad \ell_r \rightarrow \text{prot}_\ell(t_1) \mid \mu
$$
Then

\[
D_1\quad::\quad;\Sigma; \ell_c \vdash \ell \vdash t_1 : S_1 \\
(Sprot)\quad;\Sigma; \ell_c \vdash \text{prot}_\ell(t_1) : S_1 \vee \ell
\]

and by definition of the join operator, \(S_1 \vee \ell \vdash (S_1 \vee S_2) \vee \ell\) and the result holds.

Case (Sif-false). Analogous to case (if-true).

Case (Sref). Then \(t = \text{ref}^S v\) and

\[
(Sref)\quad;\Sigma; \ell_c \vdash v : S' \quad S' \vdash S \quad \ell_c \leq \text{label}(S)
\]

Suppose \(\ell_r \leq \ell_c\), then

\[
\text{ref}^S v \mid \mu \quad \ell_r \rightarrow o_\perp \mid \mu[o \mapsto v \vee \ell_r]
\]

where \(o \notin \text{dom}(\mu)\).

Let us take \(\Sigma' = \Sigma, o : S\) and let us call \(\mu' = \mu[o \mapsto v \vee \ell_r]\). Then as \(\text{dom}(\mu) = \text{dom}(\Sigma)\) then \(\text{dom}(\mu') = \text{dom}(\Sigma')\). Also, as \(\ell_r \leq \ell_c \leq \text{label}(S)\) then by Lemma 3.4, \(;\Sigma'; \perp \vdash v : S' \vee \ell_r\) and \(S' \vee \ell_r \vdash; \Sigma(o) = S\). Therefore \(;\Sigma' \vdash \mu'\).

Then

\[
(Si)\quad;\Sigma; \ell_c \vdash o_\perp : \text{Ref}_\perp S
\]

and the result holds.

Case (Sderef). Then \(t = !o_\ell\) and

\[
(Sderef)\quad;\Sigma; \ell_c \vdash !o_\ell : S \vee \ell
\]

Suppose \(\ell_r \leq \ell_c\), then

\[
!o_\ell \mid \mu \quad \ell_r \rightarrow v \vee \ell \mid \mu \text{ where } \mu(o) = v
\]

Also \(;\Sigma \vdash \mu\) then \(;\Sigma; \perp \vdash \mu(o) : S'\) and \(S' \vdash; S\). By Lemma 3.4, \(;\Sigma; \ell_c \vdash v : S'\)

\[
;\Sigma; \ell_c \vdash v \vee \ell : S' \vee \ell
\]

But \(S' \vee \ell \vdash; S\). \(S\) and the result holds.

Case (Sasgn). Then \(t = o_\ell := v\) and

\[
(Sasgn)\quad;\Sigma; \ell_c \vdash o_\ell := v : \text{Unit}_\perp
\]

Suppose \(\ell_r \leq \ell_c\), then

\[
o_\ell := v \mid \mu \quad \ell_r \rightarrow \text{unit}_\perp \mid \mu[o \mapsto v \vee \ell_r \vee \ell]
\]

Let us call \(\mu' = \mu[o \mapsto v \vee \ell_r \vee \ell]\). Also \(;\Sigma \vdash \mu\) then \(\text{dom}(\mu') = \text{dom}(\Sigma)\), and \(;\Sigma; \ell_c \vdash v : S_2\) where \(S_2 \vdash; S\). Therefore \(;\Sigma; \ell_c \vdash v \vee \ell_r \vdash; S_2 \perp \vee \ell_r \vdash; S\). But \(\ell_r \vee \ell \leq \ell_c \vee \ell \leq \text{label}(S)\), then \(S_2 \vee \ell_r \perp \ell \vdash; S\) and therefore \(;\Sigma \vdash \mu'\). Also

\[
(Su)\quad;\Sigma; \ell_c \vdash \text{unit}_\perp : \text{Unit}_\perp
\]
but

\[
\text{Unit}_\bot <: \text{Unit}_\bot
\]

and therefore the result holds.

**Case \((S::)\).** Then \(t = v :: S\) and

\[
\frac{D}{; \Sigma; \ell_c \vdash v :: S_1 \quad S_1 <: S} {; \Sigma; \ell_c \vdash v :: S : S}
\]

Suppose \(\ell_r\) such that \(\ell_r \ll \ell_c\), then

\[
v :: S | \mu \xrightarrow{\ell_r} v \bowtie label(S) | \mu
\]

But \(S_1 <: S\) then \(S_1 \bowtie S = S\) and therefore \(S_1 \bowtie label(S) = S\). Therefore:

\[
\Gamma; \Sigma; \ell_c \vdash v \bowtie label(S) : S
\]

and the result holds.

\[\square\]

**Proposition 3.6 (Canonical forms).** Consider a value \(v\) such that \(; \Sigma; \ell_c \vdash v :: S\). Then:

1. If \(S = \text{Bool}\; \ell\) then \(v = b_\ell\) for some \(b\).
2. If \(S = \text{Unit}\; \ell\) then \(v = \text{unit}_\ell\).
3. If \(S = S_1 \xrightarrow{\ell'} \ell_r S_2\) then \(v = (\lambda x : S_1.t_2)\) for some \(t_2\) and \(\ell'\).
4. If \(S = \text{Ref}\; \ell\) then \(v = o_\ell\) for some location \(o\).

**Proof.** By inspection of the type derivation rules. \[\square\]

**Proposition 3.7 (Type Safety).** If \(; \Sigma; \ell_c \vdash t : S\) then either

- \(t\) is a value \(v\)
- for any store \(\mu\) such that \(\Sigma \vdash \mu\) and any \(\ell' \ll \ell_c\), we have \(t | \mu \xrightarrow{\ell'} t' | \mu'\) and \(; \Sigma'; \ell_c \vdash t' : S'\) for some \(S' \ll S\), and some \(\Sigma' \supseteq \Sigma\) such that \(\Sigma' \vdash \mu'\).

**Proof.** By induction on the structure of \(t\).

**Case \((Sb, Su, S\lambda, Sl)\).** \(t\) is a value.

**Case \((Sprot)\).** Then \(t = \text{prot}_\ell(t)\) and

\[
\frac{; \Sigma; \ell_c \bowtie t_1 :: S_1} {; \Sigma; \ell_c \vdash \text{prot}_\ell(t_1) :: S_1 \bowtie \ell}
\]

By induction hypotheses, one of the following holds:

1. \(t_1\) is a value. Then by \((R\rightarrow)\) and Canonical Forms (Lemma 3.6), \(t | \mu \xrightarrow{\ell_r} t' | \mu\) and by Prop 3.5, \(; \Sigma; \ell_c \vdash t' :: S'\) where \(S' \ll S\) and the result holds.
2. Suppose \(\ell_r\) such that \(\ell_r \ll \ell_c\), then

\[
\frac{t_1 | \mu \xrightarrow{\ell_r \bowtie \ell} t_2 | \mu'} {\text{prot}_\ell(t_1) | \mu \xrightarrow{\ell_r} \text{prot}_\ell(t_2) | \mu'}
\]

As \(\ell_r \ll \ell_c\) then \(\ell_r \bowtie \ell \ll \ell_c \bowtie \ell\). Using induction hypotheses \(; \Sigma'; \ell_c \bowtie t_2 :: S_1'\) where \(S_1' \ll S_1\) and \(; \Sigma' \vdash \mu'\). Therefore
By induction hypotheses, one of the following holds:

\[
\vdash \Sigma; \ell_c \vee \ell \vdash t_2 : S'_1
\]

but \( S'_1 \vee \ell < S_1 \vee \ell \) and the result holds.

**Case \((S\oplus)\).** Then \( t = t_1 \oplus t_2 \) and

\[
\vdash \Sigma; \ell_c \vdash t_1 : \text{Bool}_{\ell_1} \quad \vdash \Sigma; \ell_c \vdash t_2 : \text{Bool}_{\ell_2}
\]

By induction hypotheses, one of the following holds:

1. \( t_1 \) is a value. Then by induction on \( t_2 \) one of the following holds:
   
   \((S\oplus)\)

   \[
   t | \mu \xrightarrow{\ell_r} t' | \mu
   \]

   and by Prop 3.5, \( \vdash \Sigma; \ell_c \vdash t' : S' \), where \( S' < : S \), therefore the result holds.

2. \( t_2 | \mu \xrightarrow{\ell_r'} t'_2 | \mu' \) for all \( \ell_r' \) such that \( \ell_r' \leq \ell_c \), in particular we pick \( \ell_r' = \ell_r \). Then by induction hypothesis, \( \vdash \Sigma'; \ell_c \vdash t_2 : \text{Bool}_{\ell_2'} \), where \( \text{Bool}_{\ell_2'} < : \text{Bool}_{\ell_2} \) and \( \vdash \Sigma \vdash \mu' \).

Then by \((Sf)\), \( t | \mu \xrightarrow{\ell_r} t_1 \oplus t'_2 | \mu' \) and:

\[
\vdash \Sigma; \ell_c \vdash t_1 : \text{Bool}_{\ell_1} \quad \vdash \Sigma; \ell_c \vdash t'_2 : \text{Bool}_{\ell_2'}
\]

but

\[
(\ell_1 \vee \ell_2') \leq (\ell_1 \vee \ell_2)
\]

\[
\text{Bool}_{(\ell_1 \vee \ell_2')} < : \text{Bool}_{(\ell_1 \vee \ell_2)}
\]

and the result holds.

3. \( t_1 | \mu \xrightarrow{\ell_r} t'_1 | \mu' \) for all \( \ell_r \) such that \( \ell_r' \leq \ell_c \), in particular we pick \( \ell_r' = \ell_r \). Then by induction hypotheses, \( \vdash \Sigma'; \ell_c \vdash t'_1 : \text{Bool}_{\ell_1} \) where \( \text{Bool}_{\ell_1'} < : \text{Bool}_{\ell_1} \), and \( \vdash \Sigma \vdash \mu' \).

Then by \((Sf)\), \( t | \mu \xrightarrow{\ell_r} t'_1 \oplus t_2 | \mu' \) and:

\[
\vdash \Sigma; \ell_c \vdash t'_1 : \text{Bool}_{\ell_1} \quad \vdash \Sigma; \ell_c \vdash t_2 : \text{Bool}_{\ell_2}
\]

but

\[
(\ell_1' \vee \ell_2) \leq (\ell_1 \vee \ell_2)
\]

\[
\text{Bool}_{(\ell_1' \vee \ell_2)} < : \text{Bool}_{(\ell_1 \vee \ell_2)}
\]

and the result holds.

**Case \((S\text{app})\).** Then \( t = t_1 t_2, S = S_{12} \vee \ell \) and

\[
\vdash \Sigma; \ell_c \vdash t_1 : S_{11} \xrightarrow{\ell'} \ell S_{12} \quad \vdash \Sigma; \ell_c \vdash t_2 : S_2
\]

\[
S_{12} < : S_{11} \quad \ell_c \vee \ell \leq \ell'_c
\]

By induction hypotheses, one of the following holds:

1. \( t_1 \) is a value. Then by Canonical Forms (Lemma 3.6), and induction on \( t_2 \) one of the following holds:
(a) \( t_2 \) is a value. Then by Canonical Forms (Lemma 3.6)

\[
\begin{array}{c|c|c}
\text{(R→)} & t | \mu & t' | \mu \\
\hline
\end{array}
\]

and by Prop 3.5 \( ; \Sigma; c + t' : S', \) where \( S' <: S \), therefore the result holds.

(b) \( t_2 | \mu \xrightarrow{t} t_2' | \mu' \) for all \( \ell_r' \) such that \( \ell_r' \ll \ell_c \), in particular we pick \( \ell_r' = \ell_r \). Then by induction hypothesis, \( ; \Sigma; c + \ell_r : S_1' \) and \( ; \Sigma + \mu' \).

Then by (Sf), \( t | \mu \xrightarrow{t} t_1 t_2 | \mu' \). But \( S_2' <: S_2 : S_{11} \) and then:

\[
\begin{array}{c|c|c|c|c}
; \Sigma; c + t_1 : S_{11} & t_1' : S_1' & t_2' : S_2' & S_2' <: S_{11} & \ell_c \ll \ell < \ell_c' \\
\hline
\text{(Sapp)} & ; \Sigma; c + t_1 t_2 : S_{12} & \ell_c \ll \ell < \ell_c' & \text{and the result holds.}
\end{array}
\]

(2) \( t_1 | \mu \xrightarrow{t_1} t_1' | \mu' \) for all \( \ell_r' \) such that \( \ell_r' \ll \ell_c \), in particular we pick \( \ell_r' = \ell_r \). Then by induction hypotheses, \( ; \Sigma; c + t_1' : S_1' \rightarrow t_1' S_{12} \) where \( S_1' \rightarrow t_1' S_{12} : S_1' \rightarrow t_1' S_{12} \) and \( ; \Sigma + \mu' \). Then by (Sf), \( t | \mu \xrightarrow{t_1' t_2} t_1 t_2 | \mu' \). By definition of subtyping, \( S_2 <: S_{11} : S_1' \ll c' \ll \ell' \) and \( \ell < \ell_c' \). Therefore \( \ell_c \ll \ell < \ell_c < \ell_c' \).

\[
\begin{array}{c|c|c|c|c}
; \Sigma; c + t_1' : S_1' \rightarrow t_1' S_{12} & ; \Sigma; c + t_2 : S_2 & S_2' <: S_{11} & \ell_c \ll \ell < \ell_c' \\
\hline
\text{(Sapp)} & ; \Sigma; c + t_1 t_2 : S_{12} & \ell_c \ll \ell < \ell_c' & \text{but } S_{12} \ll \ell' < S_{12} \ddag \ell \text{ and the result holds.}
\end{array}
\]

Case (Sf). Then \( t = \) \( t_0 \) then \( t_1 \) else \( t_2 \)

\[
\begin{array}{c|c|c|c|c|c}
; \Sigma; c + t_0 : \text{Boo}_c \\
\hline
\text{(Sf)} & ; \Sigma; c + \ell + t_1 : S_1 & ; \Sigma; c + \ell + t_2 : S_2 & ; \Sigma; c + \ell + t_1 t_2 : (S_1 \ll \ell) \ddag \ell \\
\end{array}
\]

By induction hypotheses, one of the following holds:

1. \( t_0 \) is a value. Then by Canonical Forms (Lemma 3.6)

\[
\begin{array}{c|c|c}
\text{(R→)} & t | \mu & t' | \mu \\
\hline
\end{array}
\]

and by Prop 3.5 \( ; \Sigma; c + t' : S', \) where \( S' <: S \), therefore the result holds.

2. \( t_0 | \mu \xrightarrow{t_0} \mu' \) for all \( \ell_r' \) such that \( \ell_r' \ll \ell_c \), in particular we pick \( \ell_r' = \ell_r \). Then by induction hypothesis, \( ; \Sigma; c + t_0 : \text{Boo}_c \), where \( \text{Boo}_c <: \text{Boo}_c \) and \( ; \Sigma + \mu' \). Then by (Sf), \( t | \mu \xrightarrow{t_0} t_1 \) else \( t_2 | \mu' \). As \( \ell_c \ll \ell < \ell_c \ll \ell \), by Lemma 3.2, \( ; \Sigma; c + \ell \ll t_1 : S_1' \) and \( ; \Sigma; c + \ell < t_1 t_2 : S_2' \), where \( S_2' <: S_1' \) and \( S_2' <: S_2 \). Therefore:

\[
\begin{array}{c|c|c|c|c|c}
; \Sigma; c + t_0 : \text{Boo}_c \\
\hline
\text{(Sf)} & ; \Sigma; c + t_1 : S_1' & ; \Sigma; c + \ell + t_2 : S_2' \\
\end{array}
\]

but by definition of join and subtyping \( (S_1' \ll S_2') \ll \ell' <: (S_1 \ll S_2) \ll \ell \) and the result holds.
Case \((S::)\). Then \(t = t_1 :: S_2\) and
\[
(S::) \quad \vdash \Sigma; \ell_c + t_1 : S_1 \quad S_1 <: S_2
\]
By induction hypotheses, one of the following holds:

1. \(t_1\) is a value. Then
   \[
   (R\rightarrow) \quad t \mid \mu \xrightarrow{\ell_r} t' \mid \mu
   \]
   and by Prop 3.5, \(\vdash \Sigma; \ell_c + t' : S'\), where \(S' <: S\), therefore the result holds.

2. \(t_1 \mid \mu \xrightarrow{\ell_r} t'_1 \mid \mu'\) for all \(\ell_r' \leq \ell_c\), in particular we pick \(\ell_r' = \ell_r\). Then by induction hypothesis, \(\vdash \Sigma; \ell_c + t'_1 : S'_1\), where \(S'_1 <: S_1\) and \(\vdash \Sigma' + \mu'\). Then by \((Sf)\), \(t \mid \mu \xrightarrow{\ell_r} t'_1 :: S_2 \mid \mu'\).
   Also, \(S'_1 <: S_1 :: S_2\) and therefore:
\[
(S::) \quad \vdash \Sigma; \ell_c + t'_1 :: S'_1 \quad S'_1 <: S_2
\]
and the result holds.

Case \((Sref)\). Then \(t = \text{ref}^S t\) and
\[
(Sref) \quad \vdash \Sigma; \ell_c + t_1 : S'_1 \quad S'_1 <: S_1 \quad \ell_c \leq \text{label}(S_1)
\]
By induction hypotheses, one of the following holds:

1. \(t_1\) is a value. Then
   \[
   (R\rightarrow) \quad t \mid \mu \xrightarrow{\ell_r} t' \mid \mu
   \]
   and by Prop 3.5, \(\vdash \Sigma; \ell_c + t' : S'\), where \(S' <: S\) and \(\vdash \Sigma' + \mu'\), therefore the result holds.

2. \(t_1 \mid \mu \xrightarrow{\ell_r} t'_1 \mid \mu'\) for all \(\ell_r' \leq \ell_c\), in particular we pick \(\ell_r' = \ell_r\). Then by induction hypothesis, \(\vdash \Sigma; \ell_c + t'_1 : S''_1\) where \(S''_1 <: S_1\) and \(\vdash \Sigma' + \mu'\). Then by \((Sf)\), \(t \mid \mu \xrightarrow{\ell_r} \text{ref}^{S_1} t'_1 \mid \mu'\)
   and:
\[
(Sref) \quad \vdash \Sigma; \ell_c + t'_1 : S''_1 \quad S''_1 <: S_1 \quad \ell_c \leq \text{label}(S_1)
\]
and the result holds.

Case \((Sderef)\). Then \(t = !t_1\) and
\[
(Sderef) \quad \vdash \Sigma; \ell_c + t_1 : \text{Ref}^S S_1
\]
By induction hypotheses, one of the following holds:

1. \(t_1\) is a value. Then by Canonical Forms (Lemma 3.6)
   \[
   (R\rightarrow) \quad t \mid \mu \xrightarrow{\ell_r} t' \mid \mu
   \]
   and by Prop 3.5, \(\vdash \Sigma; \ell_c + t' : S'\), where \(S' <: S\), therefore the result holds.
(2) \( t_1 \vdash \mu \overset{\ell_r}{\rightarrow} t_1' \mid \mu' \) for all \( \ell_r' \) such that \( \ell_r' \leq \ell_c \), in particular we pick \( \ell_r' = \ell_r \). Then by induction hypothesis, \( \vdash \Sigma; \ell_c \vdash t_1' : \text{Ref}_c S_1 \) where \( \text{Ref}_c S_1 <: \text{Ref}_c S_1 \) and \( \vdash \Sigma' \vdash \mu' \). Then by \((Sf)\), \( t \vdash \mu \overset{\ell_r}{\rightarrow} t' \mid \mu' \) and:

\[
\frac{\vdash \Sigma; \ell_c \vdash t_1' : \text{Ref}_c S_1}{
\vdash \Sigma; \ell_c \vdash t_1' : S_1 \vee \ell'}
\]

but \( S_1 \vee \ell' <: S_1 \vee \ell \) and the result holds.

Case (Sasgn). Then \( t = t_1 := t_2 \) and

\[
\frac{\vdash \Sigma; \ell_c \vdash t_1 : \text{Ref}_c S_1 \quad \vdash \Sigma; \ell_c \vdash t_2 : S_2 \quad S_2 <: S_1 \quad \ell_c \vee \ell \leq \text{label}(S_1)}{
\vdash \Sigma; \ell_c \vdash t_1 := t_2 : \text{Unit}_\bot}
\]

By induction hypotheses, one of the following holds:

1. \( t_1 \) is a value. Then by Canonical Forms (Lemma 3.6), and induction on \( t_2 \) one of the following holds:
   a. \( t_2 \) is a value. Then by Canonical Forms (Lemma 3.6)

\[
\frac{t \mid \mu \overset{\ell_r}{\rightarrow} t' \mid \mu'}{t \mid \mu \overset{\ell_r}{\rightarrow} t' \mid \mu'}
\]

and by Prop 3.5, \( \vdash \Sigma; \ell_c \vdash t' : S' \), where \( S' <: S \) and \( \vdash \Sigma' \vdash \mu' \), therefore the result holds.

b. \( t_2 \mid \mu \overset{\ell_r}{\rightarrow} t_2' \mid \mu' \) for all \( \ell_r' \) such that \( \ell_r' \leq \ell_c \), in particular we pick \( \ell_r' = \ell_r \). Then by induction hypothesis, \( \vdash \Sigma'; \ell_c \vdash t_2' : S_2' \) where \( S_2' <: S_2 \) and \( \vdash \Sigma' \vdash \mu' \).

Then by \((Sf)\), \( t \mid \mu \overset{\ell_r}{\rightarrow} t_1 := t_2' \mid \mu' \). As \( S_2' <: S_2 <: S_1 \), then:

\[
\frac{\vdash \Sigma; \ell_c \vdash t_1 : \text{Ref}_c S_1 \quad \vdash \Sigma; \ell_c \vdash t_2' : S_2'}{S_2' <: S_1 \quad \ell_c \vee \ell \leq \text{label}(S_1)}
\]

and the result holds.

(2) \( t_1 \mid \mu \overset{\ell_r}{\rightarrow} t_1' \mid \mu' \) for all \( \ell_r' \) such that \( \ell_r' \leq \ell_c \), in particular we pick \( \ell_r' = \ell_r \). Then by induction hypotheses, \( \vdash \Sigma'; \ell_c \vdash t_1' : \text{Ref}_c S_1 \), where \( \text{Ref}_c S_1 <: \text{Ref}_c S_1 \) and \( \vdash \Sigma' \vdash \mu' \). Then by \((Sf)\), \( t \mid \mu \overset{\ell_r}{\rightarrow} t_1 := t_2 \mid \mu' \). As \( \ell' \leq \ell \) then \( \ell_c \vee \ell' \leq \ell_c \vee \ell \leq \text{label}(S_1) \), and therefore:

\[
\frac{\vdash \Sigma; \ell_c \vdash t_1' : \text{Ref}_c S_1 \quad \vdash \Sigma; \ell_c \vdash t_2 : S_2}{S_2 <: S_1 \quad \ell_c \vee \ell' \leq \text{label}(S_1)}
\]

and the result holds.

\[\Box\]

### 3.2 SSLRef: Noninterference

In this section we present the proof of noninterference for SSLRef. Section 3.3 present some auxiliary definitions and section 3.4 present the proof of noninterference.
3.3 Definitions

To define the fundamental property of the step-indexed logical relations we first define how to relate substitutions:

Definition 3.8. Let \( \rho \) be a substitution, \( \Gamma \) and \( \Sigma \) a type substitutions. We say that substitution \( \rho \) satisfy environment \( \Gamma \) and \( \Sigma \), written \( \rho \models \Gamma; \Sigma \), if and only if \( \text{dom}(\rho) = \Gamma \) and \( \forall x \in \text{dom}(\Gamma), \forall \ell_c, \Gamma; \Sigma; \ell_c \vdash \rho(x) : S', \) where \( S' \ll \Gamma(x) \).

Definition 3.9 (Related substitutions). Tuples \( \langle \ell_1, \rho_1, \mu_1 \rangle \) and \( \langle \ell_2, \rho_2, \mu_2 \rangle \) are related on \( k \) steps, notation \( \Gamma; \Sigma \vdash \langle \ell_1, \rho_1, \mu_1 \rangle \approx_{\ell_o}^k \langle \ell_2, \rho_2, \mu_2 \rangle, \) if \( \rho_1 \models \Gamma; \Sigma; \ell_1 \vdash \rho(t) : S' \) and \( S' \ll \Gamma(x) \).

3.4 Proof of noninterference

Lemma 3.10 (Substitution preserves typing). If \( \Gamma; \Sigma; \ell \vdash t : S \) and \( \rho \models \Gamma; \Sigma \), then \( \Gamma; \Sigma; \ell \vdash \rho(t) : S' \) and \( S' \ll S \).

Proof. By induction on the derivation of \( \Gamma; \Sigma; \ell \vdash t : S \).

Lemma 3.11. Consider stores \( \mu_1, \mu_2, \mu_1', \mu_2' \) such that \( \mu_1 \rightarrow \mu_1' \), and substitutions \( \rho_1 \) and \( \rho_2 \), such that \( \Gamma; \Sigma \vdash \langle \ell_1, \rho_1, \mu_1 \rangle \approx_{\ell_o}^k \langle \ell_2, \rho_2, \mu_2 \rangle, \) then if \( \forall j \leq k, \) if \( \Sigma \subseteq \Sigma', \Sigma' + \mu_1' \approx_{\ell_o}^j \mu_2' \) then \( \Gamma; \Sigma \vdash \langle \ell_1, \rho_1, \mu_1' \rangle \approx_{\ell_o}^j \langle \ell_2, \rho_2, \mu_2' \rangle \).
Theorem 3.12 (Substitution preserves typing).  If \( \Gamma ; \Sigma ; \ell \vdash t : S \) then \( \forall \ell' \ll \ell, \Gamma ; \Sigma ; \ell' \vdash \ell : S \).

Proof. By induction on the derivation of \( \Gamma ; \Sigma ; \ell \vdash t \in S \).

Lemma 3.13 (Downward Closed / Monotonicity). If

(1) \( \Sigma \vdash \langle \ell_1, v_1, \mu_1 \rangle \approx^{k}_{\ell_o} \langle \ell_2, v_2, \mu_2 \rangle : S \) then
\[ \forall j \leq k, \Sigma \vdash \langle \ell_1, v_1, \mu_1 \rangle \approx^{j}_{\ell_o} \langle \ell_2, v_2, \mu_2 \rangle : S \]

(2) \( \Sigma \vdash \langle \ell_1, t_1, \mu_1 \rangle \approx^{k}_{\ell_o} \langle \ell_2, t_2, \mu_2 \rangle : C(S) \) then
\[ \forall j \leq k, \Sigma \vdash \langle \ell_1, t_1, \mu_1 \rangle \approx^{j}_{\ell_o} \langle \ell_2, t_2, \mu_2 \rangle : C(S) \]

(3) \( \Sigma \vdash \mu_1 \approx^{k}_{\ell_o} \mu_2 \) then \( \forall j \leq k, \Sigma \vdash \mu_1 \approx^{j}_{\ell_o} \mu_2 \)

Proof. By induction on type \( S \) and the definition of related stores.

Lemma 3.14. Consider simple values \( v_1 : S_i \) and
\[ \Sigma \vdash \langle \ell_1, v_1, \mu_1 \rangle \approx^{k}_{\ell_o} \langle \ell_2, v_2, \mu_2 \rangle : S \]
Then
\[ \Sigma \vdash \langle \ell_1, (v_1 \lor \ell), \mu_1 \rangle \approx^{k}_{\ell_o} \langle \ell_2, (v_2 \lor \ell), \mu_2 \rangle : S \lor \ell \]

Proof. By induction on type \( S \). We proceed by definition of related values and observational-monotonicity of the join, considering that the label stamping can only make values non observable.

Lemma 3.15 (Reduction preserves relations). Consider \( \Sigma ; \ell_i \vdash t_1 \in T \langle S \rangle, \mu_i \in \text{STORE}, \Sigma \vdash \mu_i, \) and \( \Sigma \vdash \mu_1 \approx^{k}_{\ell_o} \mu_2. \) Consider \( j < k, \) posing \( t_1 \mid \mu_i \xleftarrow{\ell_i} j t'_i \mid \mu'_i, \Sigma \subseteq \Sigma', \Sigma' \vdash \mu'_i \) we have
\[ \Sigma \vdash \langle \ell_1, t_1, \mu_1 \rangle \approx^{k}_{\ell_o} \langle \ell_2, t_2, \mu_2 \rangle : C(S) \] if and only if \( \Sigma' \vdash \langle \ell_1, t'_i, \mu'_i \rangle \approx^{k-j}_{\ell_o} \langle \ell_2, t'_2, \mu'_2 \rangle : C(S) \)

Proof. Direct by definition of
\[ \Sigma \vdash \langle \ell_1, t_1, \mu_1 \rangle \approx^{k}_{\ell_o} \langle \ell_2, t_2, \mu_2 \rangle : C(S) \] and transitivity of \( \xleftarrow{j} \).

Lemma 3.16. Consider term \( \Sigma ; \ell \vdash t : S, \) store \( \mu \) and \( j > 0, \) such that \( t \mid \mu \xleftarrow{\ell} j t' \mid \mu'. \) Then \( \mu \xleftarrow{k} \mu' \).

Proof. Trivial by induction on the derivation of \( t \). The only rules that change the store are the ones for reference and assignment, neither of which remove locations.

Lemma 3.17. Suppose that \( \Sigma \vdash \langle \ell_1 \lor \ell'_1, t_1, \mu_1 \rangle \approx^{k}_{\ell_o} \langle \ell_2 \lor \ell'_2, t_2, \mu_2 \rangle : C(S), \) and that \( \ell_i \vdash \text{prot}_{C_i}(t) : S_i \lor \ell'_i \lor \ell'_i < : S \lor \ell \) for \( i \in \{1, 2\} \). If \( \ell_i \approx^{k}_{\ell_o} \ell_2, \) and \( \ell'_i \approx^{k}_{\ell_o} \ell'_2, \)
then \( \Sigma \vdash \langle \ell_1, \text{prot}_{C_i}(t_1), \mu_1 \rangle \approx^{k}_{\ell_o} \langle \ell_2, \text{prot}_{C_i}(t_2), \mu_2 \rangle : C(S \lor \ell) \)

Proof. Consider \( j < k, \) we know by definition of related computations that
\[ t_i \mid \mu_i \xleftarrow{\ell \lor \ell'} j t'_i \mid \mu'_i \]
then \( \mu'_i \approx^{k}_{\ell_o} \mu'_2, \) and by Lemma 3.16 \( \mu_i \xleftarrow{k} \mu'_i. \) If \( t'_i \) are reducible after \( k - 1 \) steps, then the result holds immediately by \( \text{Rprot}(i) \). The interest case if \( t'_i \) are irreducible after \( j < k \) steps:
Suppose that after $j$ steps $t'_j = v_i$, then $\Sigma' \vdash (\ell_1 \lor \ell'_1, v_1, \mu'_1) \approx^{k \cdot j}_{t'_o} (\ell_2 \lor \ell'_2, v_2, \mu'_2) : S$, for some $\Sigma'$ such that $\Sigma \subseteq \Sigma'$.

Therefore:

$$\begin{align*}
\text{Let us suppose } \Sigma'; \ell_i \vdash v_i : S''_i, \text{ where } S''_i : \ell'_i < S. \text{ Then } \Sigma'; \ell_i \vdash v_i \lor \ell'_i : S''_i \lor \ell'_i, \text{ and } S''_i \lor \ell'_i < S \lor \ell. \\
\text{If } \neg \text{obs}_{t_o}(\ell'_i) \text{ by monotonicity of the join either } \neg \text{obs}_{t_o}(\ell'_i) \lor \neg \text{obs}_{t_o}(\ell_i) \text{. If } \neg \text{obs}_{t_o}(\ell'_i) \text{ then } \neg \text{obs}_{t_o}(S \lor \ell'_i) \text{ and the result holds. If } \neg \text{obs}_{t_o}(\ell_i) \text{ then obs}_{t_o}(\ell_i) \text{, the result holds immediately. If obs}_{t_o}(\ell_i \lor \ell'_i, S) \text{ then obs}_{t_o}(\ell_i \lor \ell'_i), \text{ then the result follows by Lemma 3.14, and by backward preservation of the relations (Lemma 3.15).}
\end{align*}$$

\[ \square \]

**Lemma 3.18.** Consider $\ell$, such that $\neg \text{obs}_{t_o}(\ell)$, then then $\forall k > 0$, such that, $\Sigma; \ell \vdash t : S, \Sigma \vdash \mu$, $t \vdash \mu \quad \ell \mapsto k' \vdash \mu'$, then $\forall \ell'$,

1. $\forall o \in \text{dom}(\mu') \setminus \text{dom}(\mu), \neg \text{obs}_{t_o}(\ell', \mu'(o))$.
2. $\forall o \in \text{dom}(\mu) \cap \text{dom}(\mu') \land o \neq \mu'(o), \neg \text{obs}_{t_o}(\text{label}(\Sigma(o)))$.

**Proof.** We use induction on the derivation of $t$. The interest cases are the last step of reduction rules for references and assignments.

Case ($t = o_{ref} := v$). We are only updating the heap so we only have to prove (1) and (2). Then

$$\begin{align*}
o_{ref} := v & \quad \ell \mapsto \text{unit}_\bot \vdash \mu[o \mapsto (v \lor (\ell \lor \ell''))] \\
\text{Next we have to prove that obs}_{t_o}(\text{label}(\Sigma(o))) \text{ is not defined. As } \Sigma; \ell \vdash t : S, \text{ then we know that } \\
\ell \lor \ell'' & \approx \text{label}(\Sigma(o)), \text{ and as } \neg(\text{obs}_{t_o}(\ell)) \text{ by monotonicity of the join the result holds.}
\end{align*}$$

Case ($t = \text{ref}^\Sigma v$). We are extending the heap, so we need to only prove (1). Then

$$\begin{align*}
\text{ref}^\Sigma v \vdash \mu & \quad \ell \mapsto o_\bot \vdash \mu[o \mapsto (v \lor \ell)] \\
\text{where } o \notin \text{dom}(\mu). \text{ We need to prove that obs}_{t_o}(\text{label}(v \lor \ell)) \text{ does not hold, which follows directly} \\
\text{by monotonicity of the join.}
\end{align*}$$

\[ \square \]

**Lemma 3.19.** Consider $\ell$, such that $\text{obs}_{t_o}(\ell)$ does not hold, then then $\forall k > 0$, such that, $\Sigma; \ell \vdash t_i : S_i$, and that $t_i \vdash \mu_i \quad \ell \mapsto k \cdot t'_i \vdash \mu'_i$, then if $\Sigma \vdash \mu_1 \approx^k_{t_o} \mu_2$, then $\Sigma' \vdash \mu'_1 \approx^k_{t_o} \mu'_2$ for some $\Sigma'$ such that $\Sigma \subseteq \Sigma'$ and that $\Sigma'; \ell \vdash t'_i : S'_i$, where $S'_i : \ell'_i < S_i$.

**Proof.** By Lemma 3.18 we know the result holds.

1. $\forall o \in \text{dom}(\mu'_i) \setminus \text{dom}(\mu_i), \text{obs}_{t_o}(\ell, \mu'_i(o)) \text{ does not hold, i.e. new locations are not observable and therefore as } \Sigma'; \ell \vdash \mu'_i(o) : S \land S < : \Sigma'(o), \text{ then } \neg \text{obs}_{t_o}(\text{label}(\Sigma(o)))$.
2. $\forall o \in \text{dom}(\mu'_i) \cap \text{dom}(\mu_i) \land o \neq \mu'(o), \text{obs}_{t_o}(\text{label}(\Sigma(o)))$ i.e. for all updated references they have to be previously not observable, and by definition therefore related, and second they are still non observable after the update, and by definition those locations are still related under $\ell$ because $\Sigma(o) = \Sigma'(o)$.

Therefore $\Sigma' \vdash \mu'_1 \approx^k_{t_o} \mu'_2$ and the result holds. 

\[ \square \]
Lemma 3.20. Suppose that $\Sigma; \ell_1 \vdash \text{prot}_{\ell_1}(t_1) : S' \not\vdash \ell_1', S' \not\vdash \ell_1' : S$ for $i \in \{1, 2\}$, where $\neg \text{obs}_{\ell_1}(\ell_1 \lor \ell_1')$. Also consider two stores $\mu_1$ such that $\Sigma \vdash \mu_1 \approx_{\ell_1} \mu_2$.

Then $\Sigma \vdash \langle \ell_1, \text{prot}_{\ell_1}(t_1), \mu_1 \rangle \approx_{\ell_0} \langle \ell_2, \text{prot}_{\ell_2}(t_2), \mu_2 \rangle : C(S)$

Proof. Suppose that at least $j$ more steps, where $j < k$, both subterms reduce to a value:

$$t \vdash_{\ell_i} \mu_i \quad \ell_i \lor \ell_i' \quad v_i \vdash_{\ell_i'} \mu_i'$$

Therefore:

$$\text{prot}_{\ell_i}(t) \vdash_{\ell_i} \mu_i'$$

By definition, assuming $\langle v_i \rangle \vdash_{\ell_i} \mu_i'$, we have $\Sigma \vdash \langle \ell_1, \text{prot}_{\ell_1}(t_1), \mu_1 \rangle \approx_{\ell_0} \langle \ell_2, \text{prot}_{\ell_2}(t_2), \mu_2 \rangle : C(S)$.

As the values can be radically different, we need to make sure that both values are not observables. If $\neg \text{obs}_{\ell_0}(\ell_i)$ then the values are not observables because the security context is not observable. Let us assume that $\text{obs}_{\ell_0}(\ell_i)$ holds, but $\neg \text{obs}_{\ell_0}(\ell_i')$ not. Then by monotonicity of the join, $\neg \text{obs}_{\ell_0}(\text{label}(v_i) \lor \ell_i')$ and the result follows.

Now we have to prove that the resulting stores are related, for some $\Sigma'$ such that $\Sigma \subseteq \Sigma'$. But by Lemma 3.19 the result follows immediately.

Next, we present the Noninterference proposition.

Proposition 2.5 (Security Type Soundness). If $\Gamma; \Sigma; \ell_c \vdash t : S_1' \implies \forall S, S_1' : S, \Gamma; \Sigma; \ell_c \models t : S$

Proof. We proceed by proving a more general proposition instead:
If $\Gamma; \Sigma; \ell_1 \vdash t : S_1', S_1' : S$, then $\forall \mu_1 \in \text{Store}, \Sigma \vdash \mu_1$, and $\forall k \geq 0, \forall \rho_i \in \text{SubST}, \Gamma; \Sigma \vdash \langle \ell_1, \rho_1, \mu_1 \rangle \approx_{\ell_0} \langle \ell_2, \rho_2, \mu_2 \rangle$, we have $\Sigma \vdash \langle \ell_1, \rho_1(t), \mu_1 \rangle \approx_{\ell_0} \langle \ell_2, \rho_2(t), \mu_2 \rangle : C(S)$.

By induction on the derivation of term $t$. Let us take an arbitrary index $k \geq 0$.

Case (a). $t = x$ and $\Gamma(x) = S$. $\Gamma; \Sigma \vdash \langle \ell_1, \rho_1, \mu_1 \rangle \approx_{\ell_0} \langle \ell_2, \rho_2, \mu_2 \rangle$ implies by definition that $\Sigma \vdash \langle \ell_1, \rho_1(x), \mu_1 \rangle \approx_{\ell_0} \langle \ell_2, \rho_2(x), \mu_2 \rangle : S$, and the result holds immediately.

Case (b). $t = b_g$. By definition of substitution, $\rho_1(b_g) = \rho_2(b_g) = b_g$. By definition, $\Sigma \vdash \langle \ell_1, b_g, \mu_1 \rangle \approx_{\ell_0} \langle \ell_2, b_g, \mu_2 \rangle : \text{Bool}$, as required.

Case (a). $t = o_g$, and $\Sigma(o) = S$, where $S = \text{Ref}_{\ell_1} S_1$. By definition of substitution, $\rho_1(o_g) = \rho_2(o_g) = o_g$. We know that $\Sigma; \ell_1 \vdash o_g : \text{Ref}_{\ell_1} S_1$. By definition of related stores, $\Sigma \vdash \langle \ell_1, o_g, \mu_1 \rangle \approx_{\ell_0} \langle \ell_2, o_g, \mu_2 \rangle : \text{Ref}_{\ell_1} S_1$ as required, and the result holds.

Case (a). $t = \lambda x^\ell : S_1' : \ell_1$. Then $S'_1 = S'_{11} \rightarrow_{\ell_1} S'_{12}$, and $S = S_1 \rightarrow_{\ell_1} S_2$, where $S' < S$.

By definition of substitution, assuming $x \notin \text{dom}(\rho_1)$, and Lemma 3.10:

$$\Gamma; \Sigma; \ell_1 \vdash \rho_1(t) = \Gamma; \Sigma; \ell_1 \vdash (\lambda x^\ell : S_1 \vdash \rho_1(t_1))_{\ell_1} : S'_1 \rightarrow_{\ell_1} S''_{12}$$

where $S''_{12} < S_2$. Consider $j \leq k$, $\mu_1', \mu_2'$ such that $\mu_1 \rightarrow \mu_1'$ and $\Sigma \subseteq \Sigma' \wedge \mu_1' \approx_{\ell_0} \mu_2'$, and assume two values $v_1$ and $v_2$ such that $\Sigma' \vdash \langle \ell_1, v_1, \mu_1' \rangle \approx_{\ell_0} \langle \ell_2, v_2, \mu_2' \rangle : S_1$.  

□
We need to show that:

$$
\Sigma' \vdash \langle \ell_1, (\lambda^x : S_1') t_1 \rangle_{\ell_1} \sigma_1, \mu_1' \rangle
\approx_{\ell_\sigma}^J
\langle \ell_2, (\lambda^x : S_1') t_2 \rangle_{\ell_1} \sigma_2, \mu_2' \rangle : C(S_2)
$$

Then:

$$
\begin{align*}
\langle \lambda^x : S_1' \rangle_{\ell_1} \sigma_1, \mu_1' &\quad \rightarrow_{\ell_1} \text{prot}_{\ell_1}([\sigma_1/x]t_1) \mid \mu_1' \\
\langle \lambda^x : S_1' \rangle_{\ell_1} \sigma_1, \mu_1' &\quad \rightarrow_{\ell_1}^* \text{prot}_{\ell_1}([\sigma_1/x]t_1) \mid \mu_1'
\end{align*}
$$

We then extend the substitutions to map $x$ to the arguments:

$$
\rho'_1 = \rho_1[x \mapsto \sigma_1]
$$

We know that $\Sigma' \vdash \langle \ell_1, \sigma_1, \mu_1' \rangle \approx_{\ell_\sigma}^J \langle \ell_2, \sigma_2, \mu_2' \rangle : S_1$. So as $\mu_1 \rightarrow \mu'_1$ then by Lemma 3.11, $\Gamma, x : S_1; \Sigma' \vdash \langle \ell_1, \mu_1', \mu_1' \rangle \approx_{\ell_\sigma}^J \langle \ell_2, \mu_2', \mu_2' \rangle$.

By Lemma 3.10, $\Gamma, \Sigma' ; \ell_e'' \vdash \rho'_1(t_1)$: $S_1''$ where $S_1'' <: S_1 <: S_2$. We know that $\ell_1 \vee \ell' \leq \ell_e''$, therefore by Lemma 3.2, $\Gamma, \Sigma'; \ell_1 \vee \ell \vdash \rho'_1(t_1)$: $S_1''$. Then by induction hypothesis and Lemma 3.13:

$$
\Sigma' \vdash \langle \ell_1 \vee \ell', \mu_1' \rangle \approx_{\ell_\sigma}^{j-1} \langle \ell_2 \vee \ell', \mu_2' \rangle : C(S_2),
$$

Finally, by Lemma 3.17:

$$
\Sigma' \vdash \langle \ell_1, \text{prot}_{\ell_1}(\rho'_1(t_1)), \mu_1' \rangle
\approx_{\ell_\sigma}^J \langle \ell_2, \text{prot}_{\ell'_1}(\rho_2'(t_1)), \mu_2' \rangle : C(S_2)
$$

and finally the result holds by backward preservation of the relations (Lemma 3.15).

---

Case (!). $t = !t'$, where $\Sigma; \ell_i \vdash t' : \text{Ref}_{\ell_i''} S_1$, where $S_1 \vee \ell'' <: S = S_1 \vee \ell$.

By definition of substitution:

$$
\rho_i(t) = !\rho_i(t')
$$

We have to show that

$$
\Sigma \vdash \langle \ell_1, !\rho_1(t'), \mu_1 \rangle
\approx_{\ell_\sigma}^k \langle \ell_2, !\rho_2(t'), \mu_2 \rangle : C(\text{Ref}_\ell S_1)
$$

By Lemma 3.10:

$$
\Sigma; \ell_i \vdash !\rho_1(t') : S_1 \vee \ell'''
$$

where $\ell'''' \leq \ell''' \leq \ell$. By induction hypotheses on the subterm:

$$
\Sigma \vdash \langle \ell_1, \rho_1(t'), \mu_1 \rangle \approx_{\ell_\sigma}^k \langle \ell_2, \rho_2(t'), \mu_2 \rangle : C(\text{Ref}_\ell S_1)
$$

Consider $j < k$, then by definition of related computations

$$
\rho_i(t') \mid \mu_i \rightarrow_{\ell_i}^j t'_i \mid \mu_i \quad \rightarrow \quad \Sigma' \subseteq \Sigma', \Sigma' \vdash \mu_i \approx_{\ell_\sigma}^{k-j} \mu_2' \wedge (\text{irred}(t'_i)) \Rightarrow \Sigma' \vdash \langle \ell_1, t'_i, \mu'_1 \rangle \approx_{\ell_\sigma}^{k-j} \langle \ell_2, t'_2, \mu_2' \rangle : \text{Ref}_\ell S_1
$$

If terms $t'_i$ are reducible after $j = k - 1$ steps, then

$$
!\rho_i(t) \mid \mu_i \rightarrow_{\ell_i}^j t'_i \mid \mu_i
$$

and the result holds.

If after at most $j$ steps $t'_i$ is irreducible it means that for some $j' \leq j$, $!\rho_i(t) \mid \mu_i \rightarrow_{\ell_i}^{j'} !\sigma_1 \mid \mu_1'$. If $j' = j$ then we use the same same argument for reducible terms and the result holds.
Let us consider now \( j' < j \). Then \( \Sigma' \vdash \langle \ell_1, v_1, \mu_i' \rangle \approx_{\ell_o}^{k-j'} \langle \ell_2, v_2, \mu_2' \rangle : \text{Ref}_\ell S_1 \). By Lemma 3.6, each \( v_i \) is a location \( o_{i \ell'} \), such that \( \Sigma'(o_{i \ell'}) = \text{Ref}_\ell S_1 \) and \( \ell'_i \leq \ell' \). Then:

\[
\rho_i(t) \mid \mu \xrightarrow{\ell_i} j' + 1 \quad o_{i \ell'} \mid \mu'_i \\
\xrightarrow{\ell_i} 1 \quad \text{prot}_{\ell'}(v'_i) \mid \mu'_i
\]

with \( \ell'_i \leq \ell''_i \), \( v'_i = \mu'_i(o_{i \ell'}) \). As \( \Sigma' \vdash \langle \ell_1, v_1, \mu_i' \rangle \approx_{\ell_o}^{k-j'} \langle \ell_2, v_2, \mu_2' \rangle : \text{Ref}_\ell S_1 \), then by By monotonicity of the join either both \( \text{obs}_{\ell_o}(\ell'_i) \) or \( \neg \text{obs}_{\ell_o}(\ell'_i) \). Finally as \( \Sigma' \vdash \langle \ell_1, v'_1, \mu'_1 \rangle \approx_{\ell_o}^{k-j'} \langle \ell_2, v'_2, \mu'_2 \rangle : S_1 \), by Lemma 6.60,

\[
\Sigma' \vdash \langle \ell_1, \text{prot}_{\ell'}(v'_i), \mu'_1 \rangle \\
\approx_{\ell_o}^{j} \langle \ell_2, \text{prot}_{\ell'}(v'_2), \mu'_2 \rangle : C(S_1 \lor \ell)
\]

and finally the result holds by backward preservation of the relations (Lemma 3.15).

---

Case (\( = \)). \( t = t_1 := t_2 \). Then \( S = \text{Unit}_\bot \).

By definition of substitution:

\[
\rho_i(t) = \rho_i(t_1) = \rho_i(t_2)
\]

and Lemma 3.10:

\[
\Sigma; \ell_i \vdash \rho_i(t_1) := \rho_i(t_2) : \text{Unit}_\bot
\]

We have to show that

\[
\Sigma \vdash \langle \ell_1, \rho_i(t_1) := \rho_i(t_2), \mu_i \rangle \\
\approx_{\ell_o}^{k} \langle \ell_2, \rho_2(t_1) := \rho_2(t_2), \mu_2 \rangle : C(S)
\]

By induction hypotheses

\[
\Sigma \vdash \langle \ell_1, \rho_i(t_1), \mu_i \rangle \approx_{\ell_o}^{k} \langle \ell_2, \rho_2(t_1), \mu_2 \rangle : C(S_1)
\]

Suppose \( j_1 < k \), and that \( \rho_i(t_1) \) are irreducible after \( j_1 \) steps (otherwise, similar to case \( ! \), the result holds immediately). Then by definition of related computations:

\[
\rho_i(t_1) \mid \mu_i \xrightarrow{\ell_i} j_1 v_i \mid \mu_i' \implies \Sigma \subseteq \Sigma', \Sigma' \vdash \mu_i' \approx_{\ell_o}^{k-j_1} \mu_2' = \Sigma' \vdash \langle \ell_1, v_i, \mu_i' \rangle \approx_{\ell_o}^{k-j_1} \langle \ell_2, v_2, \mu_2' \rangle : \text{Ref}_\ell S_1
\]

By Lemma 3.16 \( \mu_i \rightarrow \mu'_i \), and \( \mu'_i \approx_{\ell_o}^{k-j_1} \mu'_2 \) then by Lemma 6.41, \( \Sigma' \vdash \langle \ell_1, \rho_i(t_1), \mu_i' \rangle \approx_{\ell_o}^{k-j_1} \langle \ell_2, \rho_2(t_1), \mu_2' \rangle : C(S_2) \).

By induction hypotheses:

\[
\Sigma' \vdash \langle \ell_1, \rho_i(t_1), \mu_i' \rangle \approx_{\ell_o}^{k} \langle \ell_2, \rho_2(t_1), \mu_2' \rangle : C(S_2)
\]

Again, consider \( j_2 = k - j_1 \), if after \( j_2 \) steps \( \rho_i(t_2) \) is reducible or is a value, the result holds immediately. The interest case if after \( j_2 < j_2 \) steps \( \rho_i(t^S) \) reduces to values \( v'_i \):

\[
\rho_i(t^S) \mid \mu_i \xrightarrow{\ell_i} j_1 + j_2 v'_i \mid \mu_i'' \implies \Sigma' \subseteq \Sigma'', \Sigma'' \vdash \mu_{i''} \approx_{\ell_o}^{k-j_1-j_2} \mu_{i''} = \Sigma'' \vdash \langle \ell_1, v'_i, \mu_{i''} \rangle \approx_{\ell_o}^{k-j_1-j_2} \langle \ell_2, v'_2, \mu_{i''} \rangle : S_2
\]

Then

\[
\rho_i(t^S) \mid \mu_i \xrightarrow{\ell_i} j_1 + j_2 v'_i := v'_i \mid \mu_{i''} \approx_{\ell_o}^{k-j_1-j_2} \mu_{i''}
\]

As both values \( v_i \) are related at some reference type, then by canonical forms (Lemma 3.6) they both must be locations \( o_{i \ell'} \) for some \( S_1' <: S_1 \). We consider when the values are observable and the locations are identical (otherwise the result is trivial):

\[
v_i := v'_i \mid \mu_{i''} \\
o_{i \ell'} := v'_i \mid \mu_{i''}
\]

\[
\ell_i \equiv 1 \quad \text{unit}_\bot \mid \mu_{i''}
\]
Where \( \mu'' = \mu' '[o \mapsto (v'_1 \land (\ell'_1 \land \ell'_2))] \). As \( \Sigma'' \vdash (\ell_1, v'_1, \mu''_1) \approx_{\ell'_o}^k \ell'_1 \approx_{\ell'_o}^k \langle \ell_2, v'_2, \mu''_2 \rangle : S_2 \), and as \( \ell_1, v'_1 \preceq \text{label}(S_1) \), where \( \ell'_1 \preceq \ell \), and \( \text{label}(v'_1) \preceq \text{label}(S_1) \), then \( \Sigma''; \ell_1 \vdash v'_1 \land (\ell_1 \land \ell'_1) : S' \) and \( S' \preceq S_1 \). Then by monotonicity of the join Lemma 3.14,

\[
\Sigma'' \vdash (\ell_1, (v'_1 \land (\ell_1 \land \ell'_1))), \mu''_1
\]

But if \( \neg \text{obs}_{\ell_o}(\ell_1) \) then by monotonicity of the join \( \neg \text{obs}_{\ell_o}(v'_1 \land (\ell_1 \land \ell'_1)) \). Therefore, \( \forall \ell'_1'' \) such that \( \ell''_1 \approx_{\ell'_o}^k \ell''_2 \),

\[
\Sigma'' \vdash (\ell''_1, (v''_1 \land (\ell_1 \land \ell'_1))), \mu''_1
\]

As every values are related at type Unit, we only have to prove that \( \Sigma'' \vdash \mu''_1 \approx_{\ell'_o}^k \mu'' \), but using monotonicity (Lemma 6.47), it is trivial to prove that because either both both stores update the same location \( o \) to values that are related, therefore the result holds.

---

Case (ref.). \( t = \text{ref}^S_i \, t^S_i \). Then \( S = \text{Ref}_\perp S_1 \).
By definition of substitution:

\( \rho_i(t) = \text{ref}^S_i \, \rho_i(t') \)

and Lemma 3.10:

\( \ell_1 \vdash \text{ref}^S_i \, \rho_i(t') : \text{Ref}_\perp S_1 \)

We have to show that

\[
\Sigma \vdash (\ell_1, \text{ref}^S_i \, \rho_i(t'), \mu_1) \approx_{\ell'_o}^k (\ell_2, \text{ref}^S_i \, \rho_2(t'), \mu_2) : C(S_1)
\]

As \( \Sigma; \ell_1 \vdash \rho_i(t') : S'_i \) where \( S'_i \preceq S_1 \), by induction hypotheses:

\[
\Sigma \vdash (\ell_1, \rho_i(t'), \mu) \approx_{\ell'_o}^k (\ell_2, \rho_2(t'), \mu) : C(S_1)
\]

Consider \( j < k \), by definition of related computations

\[
\rho_i(t') | \mu_1 \rightarrow_{\ell_1}^j t'_i | \mu'_1 \quad \Sigma \subseteq \Sigma', \Sigma' \vdash \mu_2 \approx_{\ell'_o}^{j-k} \mu'_2 \land \text{irred}(t'_i) \Rightarrow \Sigma' \vdash (\ell_1, t'_i, \mu'_1) \approx_{\ell'_o}^k (\ell_2, t'_2, \mu'_2) : S'_i
\]

If terms \( t'_i \) are reducible after \( j = k - 1 \) steps, then

\[
\text{ref}^S_i \, \rho_i(t') | \mu_1 \rightarrow_{\ell_1}^j \text{ref}^S_i \, t'_i | \mu'_1 \quad \text{and the result holds.}
\]

If after at most \( j \) steps \( t'_i \) is irreducible, it means that for some \( j' \leq j \) \( \text{ref}^S_i \, \rho_i(t') | \mu_1 \rightarrow_{\ell_1}^{j'} \text{ref}^S_i \, v_1 | \mu''_1 \).
If \( j' = j \) then we use the same same argument for reducible terms and the result holds.
Let us consider now \( j' < j \). Then:

\[
\rho_i(t) | \mu \rightarrow_{\ell_1}^{j'+1} \text{ref}^S_i \, v_1 | \mu''_1
\]

with, \( \mu''_1 = \mu' '[o \mapsto (v_1 \land \ell_1)] \). Also, as \( \Sigma' \vdash (\ell_1, v_1, \mu'_1) \approx_{\ell'_o}^{k-j'} (\ell_2, v_2, \mu'_2) : S_1 \), then \( \Sigma'' \vdash (\ell_1, v_1, \mu''_1) \approx_{\ell'_o}^{j'} (\ell_2, v_2, \mu''_2) : S_1 \), with \( \Sigma'' = \Sigma', \mu : S_1 \). And as \( \text{label}(v_1) \land \ell_1 \preceq \text{label}(S_1) \), then by Lemma 3.14,

\[
\Sigma'' \vdash (\ell_1, v_1 \land \ell_1, \mu'') \approx_{\ell'_o}^{k-j} (\ell_2, v_2 \land \ell_2, \mu'_2) : S_1
\]

If \( \neg \text{obs}_{\ell_o}(\ell_1) \) then by monotonicity of the join \( \neg \text{obs}_{\ell_o}(\text{label}(\ell'_1 \land \ell_1)) \) and \( \neg \text{obs}_{\ell_o}(\text{label}(\Sigma''(o))) \). Therefore, \( \forall \ell''_1 \) such that \( \ell''_1 \approx_{\ell'_o}^k \ell''_2 \),

\[
\Sigma'' \vdash (\ell''_1, v_1 \land \ell_1, \mu''_1) \approx_{\ell'_o}^{j-k} (\ell''_2, v_2 \land \ell_2, \mu''_2) : S_1
\]

By definition of related stores \( \Sigma'' - \mu'' \approx_{\ell'_o}^{j-k} \mu''_2 \). Then by Monotonicity of the relation (Lemma 6.47) \( \Sigma'' - \mu''_1 \approx_{\ell'_o}^{j-k-2} \mu''_2 \) and the result holds.
Case ($\oplus$). $t = t_1 \oplus t_2$

By definition of substitution:

$$\rho_i(t) = \rho_i(t_1) \oplus \rho_i(t_2)$$

and Lemma 3.10:

$$\Sigma; \ell_i + \rho_i(t_1) \oplus \rho_i(t_2) : S''$$

with $S''_i : S''_i : S$. We use a similar argument to case $\equiv$ for reducible terms. The interest case is when we suppose some $j_1$ and $j_2$ such that $j_1 + j_2 < k - 3$ where:

$$\rho_i(t_1) | \mu_i \xrightarrow{\ell_i} j_1 v_{i1} | \mu_i' \implies \Sigma \subseteq \Sigma', \Sigma' + \mu_i' \approx^{k-j_1} \mu'_2 \wedge \Sigma' + \langle \ell_1, v_{i1}, \mu_i' \rangle \approx^{k-j_1} \langle \ell_2, v_{i1}, \mu'_2 \rangle : S_1$$

$$\rho_i(t_2) | \mu_i' \xrightarrow{\ell_i} j_2 v_{i2} | \mu_i'' \implies \Sigma' \subseteq \Sigma'', \Sigma'' + \mu_i'' \approx^{k-j_2} \mu''_2 \wedge \Sigma'' + \langle \ell_1, v_{i2}, \mu_i'' \rangle \approx^{k-j_2} \langle \ell_2, v_{i2}, \mu''_2 \rangle : S_2$$

By Lemma 3.6, each $v_{ij}$ is a boolean $(b_{ij})_{\ell_{ij}}$ then:

$$\rho_i(t) | \mu_i'' \xrightarrow{\ell_i} j_1 v_{i1} + j_2 v_{i2} | \mu_i''$$

with $b_i = b_{i1}[\oplus] b_{i2}$, $\ell'_{i} = \ell_{i1} \vee \ell_{i2}$, and $\ell'_{i} \leq \text{label}(S''_i) \leq \text{label}(S)$. It remains to show that:

$$\Sigma'' + \langle \ell_1, (b_1)_{\ell_{i1}} | \mu_i'' \rangle \approx^{k-j_1-j_2-3} \langle \ell_2, (b_2)_{\ell_{i2}} | \mu_i'' \rangle : S$$

If $\neg \text{obs}_{\ell_{o}}(\ell_i)$, then the result is trivial because the resulting booleans are also related as they are not observable.

If $\text{obs}_{\ell_{o}}(\ell_i)$, and $\neg \text{obs}_{\ell_{o}}(\ell'_{i1})$ or $\neg \text{obs}_{\ell_{o}}(\ell'_{i2})$, then by monotonicity of the join, $\neg \text{obs}_{\ell_{o}}(\ell'_{i})$ and the result holds. If $\text{obs}_{\ell_{o}}(\ell'_{i})$ then $\text{obs}_{\ell_{o}}(\ell_{i})$ and therefore $b_{i1} = b_{21}$ and $b_{i2} = b_{22}$, so $b_i = b_2$, and the result holds.

Case (app). $t = t_1 t_2$, with $\Sigma; \ell_i + t_1 : S_{i1} \xrightarrow{\ell_{ei}} \ell_{i1} S_{i2}$, and $\Sigma; \ell_i + t_2 : S''_{i1}$. Also $S_{i1} \xrightarrow{\ell_{ei}} \ell_{i1} S_{i2} <: S_1 \xrightarrow{\ell_{e}} \ell_{i1} S_{i2}$, and $S = S_2$.

By definition of substitution:

$$\rho_i(t) = \rho_i(t_1) \rho_i(t_2)$$

and Lemma 3.10:

$$\Sigma; \ell_i + \rho_i(t_1) \rho_i(t_2) : S''_{i2}$$

with $S''_{i2} : S_{i2} : S_2$. We use a similar argument to case $\equiv$ for reducible terms. The interest case is when we suppose some $j_1$ and $j_2$ such that $j_1 + j_2 < k$ where by induction hypotheses and the definition of related computations:

$$\rho_i(t_1) | \mu_i \xrightarrow{\ell_i} j_1 v_{i1} | \mu_i' \implies \Sigma \subseteq \Sigma', \Sigma' + \mu_i' \approx^{k-j_1} \mu'_2 \wedge \Sigma' + \langle \ell_1, v_{i1}, \mu_i' \rangle \approx^{k-j_1} \langle \ell_2, v_{i1}, \mu'_2 \rangle : S_1$$

$$\rho_i(t_2) | \mu_i' \xrightarrow{\ell_i} j_2 v_{i2} | \mu_i'' \implies \Sigma' \subseteq \Sigma'', \Sigma'' + \mu_i'' \approx^{k-j_2} \mu''_2 \wedge \Sigma'' + \langle \ell_1, v_{i2}, \mu_i'' \rangle \approx^{k-j_2} \langle \ell_2, v_{i2}, \mu''_2 \rangle : S_2$$

Then

$$\rho_i(t) | \mu_i \xrightarrow{\ell_i} j_1 j_2 v_{i1} v_{i2} | \mu_i''$$
If \( \text{obs}_{\ell_o}(\ell_i, v_{i1}) \) then, by definition of \( \approx_{\ell_o} \) at values of function type, we have:

\[
\Sigma' + \langle \ell_1, (v_{i1} v_{i12}), \mu''_1 \rangle \\
\approx_{\ell_o}^{k-h_j} \langle \ell_2, (v_{21} v_{222}), \mu''_2 \rangle : C(S_2 \land \ell)
\]

Finally, by backward preservation of the relations (Lemma 3.15) the result holds.

If \( \neg \text{obs}_{\ell_o}(\ell_i, v_{i1}) \), and we assume by canonical forms that \( v_{i1} = (\lambda \ell'_i x.t_i)_{\ell'_i} \) then, either \( \neg \text{obs}_{\ell_o}(\ell_i) \) or \( \neg \text{obs}_{\ell_o}(\ell'_i) \) and

\[
(v_{i1} v_{i12}) | \mu''_1
= (\lambda \ell'_i x.t_i)_{\ell'_i} v_{i12}) | \mu''_1
\overset{\ell_i}{\longrightarrow}^1 \text{prot}_{\ell_i}(t'_i) | \mu''_1
\]

If either \( \neg \text{obs}_{\ell_o}(\ell_i) \) or \( \neg \text{obs}_{\ell_o}(\ell'_i) \) then by Lemma 3.20,

\[
\Sigma'' + \langle \ell_1, \text{prot}_{\ell_i}(t'_i), \mu''_1 \rangle \\
\approx_{\ell_o}^{k-h_j} \langle \ell_2, \text{prot}_{\ell_i}(t'_2), \mu''_2 \rangle : C(S_2 \land \ell)
\]

Finally, by backward preservation of the relations (Lemma 3.15) the result holds.

---

Case (if) \( t = t_1 \) then \( t_2 \) else \( t_3 \), with \( \Sigma; \ell_i \vdash t_1 : S_1, \Sigma; \ell'_i \vdash t_2 : S_2, \Sigma; \ell''_i \vdash t_3 : S_3, \ell'' = \ell \lor \text{label}(S_1) \), and \( S'' = S_2 \lor S_3 \triangleleft \vdash \Sigma \)

By definition of substitution:

\[
\rho_1(t) = \begin{cases} 
\rho_1(t_1) & \text{if } \rho_1(t_1) \text{ then } \\
\rho_1(t_2) & \text{else } \rho_1(t_3) \end{cases}
\]

We use a similar argument to case := for reducible terms. The interest case is when we suppose some \( j_1 \) and \( j_2 < k \) where by induction hypotheses and related computations we have that:

\[
\rho_1(t_1) | \mu_1 \overset{\ell_i}{\longrightarrow}^{j_1} v_{i1} | \mu'_1 \implies \Sigma' \subseteq \Sigma, \Sigma' + \mu'_1 \approx_{\ell_o}^{k-h_j} \mu'_2 \land \Sigma' + \langle \ell_1, v_{i11}, \mu''_1 \rangle \approx_{\ell_o}^{k-h_j} \langle \ell_2, v_{212}, \mu''_2 \rangle : S_1
\]

By Lemma 3.6, each \( v_{i11} \) is a boolean \( \langle b_{i11} \rangle_{\ell_{i11}} \), such that \( \Sigma'; \ell_i \vdash (b_{i11})_{\ell_{i11}} : \text{Bool}_{\ell_{i11}} \) and \( \text{Bool}_{\ell_{i11}} \triangleleft \vdash \Sigma_1 \), implies \( \Sigma_1 = \text{Bool}_{\ell_{i1}} \). Then:

\[
\rho_1(t) | \mu_1 \overset{\ell_i}{\longrightarrow}^{j_1+1} (b_{i1})_{\ell_{i11}} \text{ if } \rho_1(t_1) \text{ then } \rho_1(t_2) \text{ else } \rho_1(t_3) | \mu'_1
\]

Let us consider \( \neg \text{obs}_{\ell_o}(\ell_i, (b_{i11})_{\ell_{i11}}) \). Let us assume the worst case scenario and that both execution reduce via different branches of the conditional. Then

\[
\rho_1(t) | \mu_1 \overset{\ell_i}{\longrightarrow}^{j_1+2} \text{prot}_{\ell_{i11}}(\rho_1(t_2)) | \mu'_1
\rho_2(t) | \mu_2 \overset{\ell_i}{\longrightarrow}^{j_1+2} \text{prot}_{\ell_{i11}}(\rho_2(t_3)) | \mu'_2
\]

But because \( \neg \text{obs}_{\ell_o}(\ell_i, (b_{i11})_{\ell_{i11}}) \), then either \( \neg \text{obs}_{\ell_o}(\ell_i) \) or \( \neg \text{obs}_{\ell_o}(\ell_{i11}) \) and therefore, \( \neg \text{obs}_{\ell_o}(\ell_i \lor \ell_{i11}) \). Then by Lemma 3.20,

\[
\Sigma' + \langle \ell_1, \text{prot}_{\ell_{i11}}(\rho_1(t_2)), \mu'_1 \rangle \approx_{\ell_o} \langle \ell_2, \text{prot}_{\ell_{i11}}(\rho_2(t_3)), \mu'_2 \rangle
\]

and the result holds by backward preservation of the relations (Lemma 3.15).

Now let us consider if \( \text{obs}_{\ell_o}(\ell_i, (b_{i11})_{\ell_{i11}}) \) holds. Then by definition of \( \approx_{\ell_o} \) on boolean values, \( b_{i11} = b_{21} \). Because \( b_{i11} = b_{21} \), both \( \rho_1(t) \) and \( \rho_2(t) \) step into the same branch of the conditional. Let us assume the condition is true (the other case is similar):
Then by induction hypothesis $\Sigma' \vdash (\ell_1 \vee \ell_1, \rho_1(t_2), \mu^1_1) \approx_{\ell_0}^k (\ell_2 \vee \ell_2, \rho_2(t_2), \mu^2_0) : S_2$, and by Lemma 3.17,

$\Sigma' \vdash (\ell, \text{prot}_{\ell_11}(\rho_1(t_2)), \mu^1_1) \approx_{\ell_0}^k (\ell, \text{prot}_{\ell_21}(\rho_2(t_2)), \mu^2_0) : S$

and the result holds by backward preservation of the relations (Lemma 3.15).

Case (prot()). Direct by using Lemma 3.17.

\[\square\]

4 GRADUALIZING THE STATIC SEMANTICS

In section 4.1, we show the proof of optimality and soundness of the abstraction. In section 4.2, we present the proof for the Static Gradual Guarantee.

4.1 From Gradual Labels to Gradual Types

**Proposition 4.1** (\(\alpha\) is Sound). If \(\hat{\ell} \neq \emptyset\) then \(\hat{\ell} \subseteq \gamma(\alpha(\hat{\ell}))\).

**Proof.** By case analysis on the structure of \(\hat{\ell}\). If \(\hat{\ell} = \{\ell\}\) then \(\gamma(\alpha(\{\ell\})) = \gamma(\ell) = \{\ell\} = \hat{\ell}\), otherwise \(\gamma(\alpha(\hat{\ell})) = \gamma(\emptyset) = \text{LABEL} \supseteq \hat{\ell}\). \(\square\)

**Proposition 4.2** (\(\alpha\) is Optimal). If \(\hat{\ell} \subseteq \gamma(g)\) then \(\alpha(\hat{\ell})\) \(\subseteq g\).

**Proof.** By case analysis on the structure of \(g\). If \(g = \ell\), \(\gamma(\ell) = \{\ell\}\); \(\hat{\ell} \subseteq \ell \neq \emptyset\) implies \(\alpha(\hat{\ell}) = \alpha(\{\ell\}) = \ell \subseteq g\). If \(\hat{\ell} = \emptyset\), \(\alpha(\hat{\ell})\) is undefined. If \(g = \emptyset\), \(g' \subseteq g\) for all \(g'\). \(\square\)

**Proposition 4.3** (\(\alpha\) is Sound and Optimal). If \(\hat{\ell} \neq \emptyset\) then,

(i) \(\hat{\ell} \subseteq \gamma(\alpha(\hat{\ell}))\).

(ii) If \(\hat{\ell} \subseteq \gamma(g)\) then \(\alpha(\hat{\ell}) \subseteq g\).

**Proof.** Trivial using Prop 4.1 and 4.2. \(\square\)

**Proposition 4.4** (\(\alpha_S\) is Sound). If \(\hat{S}\) valid, then \(\hat{S} \subseteq \gamma_S(\alpha_S(\hat{S}))\).

**Proof.** By well-founded induction on \(\hat{S}\) according to the ordering relation \(\hat{S} \sqsubseteq \hat{S}\) defined as follows:

\[
\text{dom}({\hat{S}}) \sqsubseteq \hat{S} \\
\text{cod}({\hat{S}}) \sqsubseteq \hat{S}
\]

Where \(\text{dom}, \text{cod} : \mathcal{P}(\text{GType}) \rightarrow \mathcal{P}(\text{GType})\) are the collecting liftings of the domain and codomain functions \(\text{dom}, \text{cod}\) respectively, e.g.,

\[
\text{dom}(\hat{S}) = \{\text{dom}(S) \mid S \in \hat{S}\}
\]

We then consider cases on \(\hat{S}\) according to the definition of \(\alpha_S\).

Case (\([\text{Bool}_{\ell_1}]\)).

\[
\gamma_S(\alpha_S([\text{Bool}_{\ell_1}])) = \gamma_S(\text{Bool}_{\alpha([\ell_1])}) \\
= \{\text{Bool}_\ell \mid \ell \in \gamma(\alpha([\ell_1]))\} \\
\supseteq \{\text{Bool}_{\ell_1}\} \text{ by soundness of } \alpha.
\]
Case (\(S_{i1} \xrightarrow{\ell_i} t_i S_{i2} \)).

\[
\gamma_S(\alpha_S(\{ S_{i1} \xrightarrow{\ell_i} t_i S_{i2} \}))
= \gamma_S(\alpha_S(\{ S_{i1} \xrightarrow{\ell_i} t_i S_{i2} \}))
= \gamma_S(\alpha_S(\{ S_{i1} \xrightarrow{\ell_i} t_i S_{i2} \}))
\]

by induction hypothesis on \(\{ S_{i1} \xrightarrow{\ell_i} t_i S_{i2} \}\), and soundness of \(\alpha\).

Case (\(\{ \text{Ref}_{\ell_i} S_i \}\)).

\[
\gamma_S(\alpha_S(\{ \text{Ref}_{\ell_i} S_i \}))
= \gamma_S(\alpha_S(\{ \text{Ref}_{\ell_i} S_i \}))
= \gamma_S(\alpha_S(\{ \text{Ref}_{\ell_i} S_i \}))
\]

by induction hypothesis on \(\{ S_{i1} \xrightarrow{\ell_i} t_i S_{i2} \}\), and soundness of \(\alpha\).

\[
\square
\]

**Proposition 4.5 (\(\alpha_S\) is Optimal).** If \(\hat{S}\) valid and \(\hat{S} \subseteq \gamma_S(U)\) then \(\alpha_S(\hat{S}) \subseteq U\).

**Proof.** By induction on the structure of \(U\).

Case (\(\text{Bool}_g\)). \(\gamma_S(\text{Bool}_g) = \{ \text{Bool}_\ell | \ell \in \gamma(g) \}\)

So \(\hat{S} = \{ \text{Bool}_\ell | \ell \in \hat{\ell} \}\) for some \(\hat{\ell} \subseteq \gamma(g)\). By optimality of \(\alpha\), \(\alpha(\hat{\ell}) \subseteq g\), so \(\alpha_S(\{ \text{Bool}_\ell | \ell \in \hat{\ell} \}) = \text{Bool}_{\alpha(\hat{\ell})} \subseteq \text{Bool}_g\).

Case (\(U_1 \xrightarrow{g} U_2\)). \(\gamma_S(U_1 \xrightarrow{g} U_2) = \gamma_S(U_1) \xrightarrow{\gamma(g)} \gamma_S(U_2)\).

So \(\hat{S} = \{ S_{i1} \xrightarrow{\ell_i} S_{i2} \}\), with \(\{ S_{i1} \} \subseteq \gamma_S(U_1)\), \(\{ S_{i2} \} \subseteq \gamma_S(U_2)\), \(\{ \ell_{c_1} \} \subseteq \gamma(g_c)\) and \(\{ \ell_{c_1} \} \subseteq \gamma(g)\). By induction hypothesis, \(\alpha_S(\{ S_{i1} \}) \subseteq U_1\) and \(\alpha_S(\{ S_{i2} \}) \subseteq U_2\), and by optimality of \(\alpha\), \(\alpha(\{ \ell_{c_1} \}) \subseteq g_c\) and \(\alpha(\{ \ell_{c_1} \}) \subseteq g\). Hence \(\alpha_S(\{ S_{i1} \xrightarrow{\ell_i} S_{i2} \}) = \alpha_S(\{ S_{i1} \}) \xrightarrow{\alpha(\ell_{c_1})} \alpha_S(\{ S_{i2} \}) \subseteq U_1 \xrightarrow{g_c} U_2\).

Case (\(\text{Ref}_g U\)). \(\gamma_S(\text{Ref}_g U) = \{ \text{Ref}_\ell S | \ell \in \gamma(g), S \in \gamma(U) \}\)

So \(\hat{S} = \{ \text{Ref}_\ell S | \ell \in \hat{\ell}, S \in \gamma(U) \}\) for some \(\{ S_{i1} \} \subseteq \gamma_S(U)\) and some \(\hat{\ell} \subseteq \gamma(g)\). By induction hypothesis \(\alpha_S(\{ S_{i1} \}) \subseteq U\) and by optimality of \(\alpha\), \(\alpha(\hat{\ell}) \subseteq g\), so \(\alpha_S(\{ \text{Ref}_\ell S | \ell \in \hat{\ell}, S \in \gamma(U) \}) = \text{Ref}_{\alpha(\hat{\ell})} \alpha_S(\{ S_{i1} \}) \subseteq \text{Ref}_g U\).

\[
\square
\]

**Proposition 2.9 (\(\alpha_S\) is Sound and Optimal).** Assuming \(\hat{S}\) valid:

(i) \(\hat{S} \subseteq \gamma_S(\alpha_S(\hat{S}))\)

(ii) If \(\hat{S} \subseteq \gamma_S(U)\) then \(\alpha_S(\hat{S}) \subseteq U\).

**Proof.** Trivial using Prop 4.4 and 4.5.
4.2 Static Criteria for Gradual Typing

In this section we present the proof of Static Gradual Guarantee for GSLRef.

**Proposition 4.6 (Static Conservative Extension).** Let $\tau;\gamma$ denote SSLRef's type system. Then for any static language term $t \in \text{TERM}$, $\cdot;\Sigma;\ell_\epsilon \vdash_S t : S$ if and only if $\cdot;\Sigma;\ell_\epsilon \vdash : S$.

**Proof.** By induction over the typing derivations. The proof is trivial because static types are given singleton meanings via concretization. \(\square\)

**Definition 4.7 (Term precision).**

\[
\begin{align*}
\text{(Px)} & \quad x \sqsubseteq x \\
\text{(Pb)} & \quad g \sqsubseteq g’ \quad b_g \sqsubseteq b_{g’} \\
\text{(Pu)} & \quad g \sqsubseteq g’ \quad \text{unit}_g \sqsubseteq \text{unit}_{g’} \\
\text{(Papp)} & \quad t \sqsubseteq t’ \quad U_1 \sqsubseteq U_1’ \quad g \sqsubseteq g’ \\
\text{(Pu)} & \quad t \sqsubseteq t’ \quad U_1 \sqsubseteq U_1’ \quad g \sqsubseteq g’ \\
\text{(Pprot)} & \quad t \sqsubseteq t’ \quad g \sqsubseteq g’ \quad \text{prot}_g(t) \sqsubseteq \text{prot}_{g’}(t’) \\
\text{(Pif)} & \quad t \sqsubseteq t’ \quad \text{if } t \text{ then } t_1 \text{ else } t_2 \sqsubseteq \text{if } t’ \text{ then } t_1 \text{ else } t_2 \\
\text{(Pref)} & \quad t \sqsubseteq t’ \quad U \sqsubseteq U’ \\
\text{(Pdref)} & \quad t \sqsubseteq t’ \quad \text{ref}^U(t) \sqsubseteq \text{ref}^{U’}(t’) \\
\text{(Pasgn)} & \quad t \sqsubseteq t’ \quad U \sqsubseteq U’
\end{align*}
\]

**Definition 4.8 (Type environment precision).**

\[
\begin{align*}
\text{\Gamma} \sqsubseteq \text{\Gamma’} \\
\text{\Gamma, x : U \sqsubseteq \Gamma’, x : U’}
\end{align*}
\]

**Lemma 4.9.** If $\Gamma;::;g_c \vdash t : U$ and $\Gamma \sqsubseteq \Gamma’$, then $\Gamma’;::;g_c \vdash t : U’$ for some $U \sqsubseteq U’$.

**Proof.** Simple induction on typing derivations. \(\square\)

**Lemma 4.10.** If $U_1 \sqsubseteq U_2$ and $U_1 \sqsubseteq U_1’$ and $U_2 \sqsubseteq U_2’$ then $U_1’ \sqsubseteq U_2’$.

**Proof.** By definition of $\sqsubseteq$, there exists $\langle S_1, S_2 \rangle \in \gamma^3(U_1, U_2)$ such that $S_1 \sqsubseteq S_2$, $U_1 \sqsubseteq U_1’$ and $U_2 \sqsubseteq U_2’$ mean that $\gamma(U_1) \sqsubseteq \gamma(U_1’)$ and $\gamma(U_2) \sqsubseteq \gamma(U_2’)$, therefore $\langle S_1, S_2 \rangle \in \gamma^3(U_1’, U_2’)$. \(\square\)

**Lemma 4.11.** If $g_1 \sqsubseteq g_2 \sqsubseteq g_3$, $g_1 \sqsubseteq g_1’$, $g_2 \sqsubseteq g_2’$ and $g_3 \sqsubseteq g_3’$, then $g_1’ \sqsubseteq g_2’ \sqsubseteq g_3’$.

**Proof.** By definition of the consistent judgment, there exists $\langle \ell_1, \ell_2, \ell_3 \rangle \in \gamma^3(g_1, g_2, g_3)$ such that $\ell_1 \sqsubseteq \ell_2 \sqsubseteq \ell_3$, $g_1 \sqsubseteq g_1’$, $g_2 \sqsubseteq g_2’$ and $g_3 \sqsubseteq g_3’$ mean that $\gamma(g_1) \sqsubseteq \gamma(g_1’)$, $\gamma(g_2) \sqsubseteq \gamma(g_2’)$ and $\gamma(g_3) \sqsubseteq \gamma(g_3’)$ respectively. Therefore $\langle \ell_1, \ell_2, \ell_3 \rangle \in \gamma^3(g_1’, g_2’, g_3’)$.

**Lemma 4.12.** If $g_1 \preceq g_2$, $g_1 \sqsubseteq g_1’$ and $g_2 \sqsubseteq g_2’$, then $g_1’ \preceq g_2’$.

**Proof.** Using almost identical argument of Lemma 4.11. \(\square\)

**Proposition 4.13 (Static Gradual Guarantee).** Suppose $g_{c_1} \sqsubseteq g_{c_2}$ and $t_1 \sqsubseteq t_2$.

If $\cdot;::;g_c \vdash t_1 : U_1$ then $\cdot;::;g_{c_2} \vdash t_2 : U_2$ where $U_1 \sqsubseteq U_2$.

**Proof.** We prove the property on opens terms instead of closed terms: If $\Gamma;::;g_{c_1} \vdash t_1 : U_1$, $g_{c_1} \sqsubseteq g_{c_2}$ and $t_1 \sqsubseteq t_2$ then $\Gamma;::;g_{c_2} \vdash t_2 : U_2$ and $U_1 \sqsubseteq U_2$.

The proof proceed by induction on the typing derivation.

Case $(Ux, Ub, Uu)$. Trivial by definition of $\sqsubseteq$ using (Px), (Pb), (Pu) respectively.
Case \((U\lambda)\). Then \(t_1 = (\lambda x : U_1'. t)')_g\) and \(U_1 = U_1' \rightarrow_{g'} U_2'\). By \((U\lambda)\) we know that:

\[
\begin{align*}
\Gamma, x : U_1'; \ldots ; g_c' \vdash t : U_2' \\
\Gamma; \vdash g_c_1 + (\lambda x : U_1'. t)'_g : U_1' \rightarrow_{g'} U_2'
\end{align*}
\]

(1)

Consider \(g_{c_2}\) such that \(g_{c_1} \sqsubseteq g_{c_2}\) and \(t_1\) such that \(t_1 \sqsubseteq t_2\). By definition of term precision \(t_2\) must have the form \(t_2 = (\lambda x : U_1'''. t')''_g\) and therefore

\[
\begin{align*}
t \sqsubseteq t' \\
(\lambda x : U_1'. t)_g \sqsubseteq (\lambda x : U_1'''. t')''_g \\
g \sqsubseteq g'
\end{align*}
\]

(2)

Using induction hypotheses on the premise of 1, \(\Gamma, x : U_1'; \ldots ; g_{c_2} \vdash t' : U_2''\) with \(U_2' \sqsubseteq U_2''\). By Lemma 4.9, \(\Gamma, x : U_1'''; \ldots ; g_{c_2} \vdash t' : U_2''''\) where \(U_2''' \sqsubseteq U_2''''\). Then we can use rule \((U\lambda)\) to derive:

\[
\begin{align*}
\Gamma, x : U_1'''; \ldots ; g_{c_2}'' \vdash t' : U_2'''' \\
\Gamma; \vdash g_{c_1} + (\lambda x : U_1'. t)'_g : U_1'''' \rightarrow_{g'} U_2''''
\end{align*}
\]

Where \(U_2 \sqsubseteq U_2''\). Using the premise of 2 and the definition of type precision we can infer that

\[
U_1' \rightarrow_{g'} U_2' \sqsubseteq U_1'''' \rightarrow_{g'''} U_2''''
\]

and the result holds.

Case \((Uo)\). This case can not happen because initial programs do not contain locations.

Case \((Uprot)\). Then \(t_1 = \text{prot}_g(t)\) and \(U_1 = U \triangledown g\). By \((Uprot)\) we know that:

\[
\begin{align*}
\Gamma; \vdash g_{c_1} \triangledown g \vdash t : U \\
\Gamma; \vdash g_{c_1} + \text{prot}_g(t) : U \triangledown g
\end{align*}
\]

(3)

Consider \(g_{c_2}\) such that \(g_{c_1} \triangledown g \sqsubseteq g_{c_2} \triangledown g'\). Using induction hypotheses on the premises of 3, we can use rule \((Uprot)\) to derive:

\[
\begin{align*}
\Gamma; \vdash g_{c_2} \triangledown g' \vdash t' : U' \\
\Gamma; \vdash g_{c_2} + \text{prot}_{g'}(t') : U' \triangledown g'
\end{align*}
\]

For some \(U'\), where \(U \sqsubseteq U'\). Using the premise of 4 and the definition of join we can infer that

\[
U \triangledown g \sqsubseteq U' \triangledown g'
\]

and the result holds.

Case \((U\oplus)\). Then \(t_1 = t_1' \oplus t_2'\) and \(U_1 = \text{Bool}_{(g_1, g_2)}\). By \((U\oplus)\) we know that:

\[
\begin{align*}
\Gamma; \vdash g_{c_1} + t_1' : \text{Bool}_{g_1} \\
\Gamma; \vdash g_{c_1} + t_2' : \text{Bool}_{g_2}
\end{align*}
\]

\[
\Gamma; \vdash g_{c_1} + t_1' \oplus t_2' : \text{Bool}_{(g_1, g_2)}
\]

(5)
Consider \( g_{c2} \) such that \( g_{c1} \subseteq g_{c2} \) and \( t_2 \) such that \( t_1 \subseteq t_2 \). By definition of term precision \( t_2 \) must have the form \( t_2 = t_1'' \oplus t_2'' \) and therefore

\[
(P\oplus) \quad t_1' \subseteq t_1'' \quad t_2' \subseteq t_2'' \quad t_1' \oplus t_2' \subseteq t_1'' \oplus t_2''.
\]

(6)

Using induction hypotheses on the premises of 5, we can use rule \((U\oplus)\) to derive:

\[
(U\oplus) \quad \Gamma; \cdot; g_{c2} \vdash t_1' : \text{Bool}_g' \quad \Gamma; \cdot; g_{c2} \vdash t_2' : \text{Bool}_g'
\]

\[
\Gamma; \cdot; g_{c2} \vdash t_1'' \oplus t_2'' : \text{Bool}_{(g'_1 \vee g'_2)}
\]

Where \( g_1' \subseteq g_1'' \) and \( g_2' \subseteq g_2'' \). Using the premise of 6 and the definition of type precision we can infer that

\[
(\text{Bool}_{(g'_1 \vee g'_2)}) \subseteq (\text{Bool}_{(g_1'' \vee g_2'')})
\]

and the result holds.

**Case** \((U\text{app})\). Then \( t_1 = t_1' t_2' \) and \( U_1 = U_{12} \sim g \). By \((U\text{app})\) we know that:

\[
(U\text{app}) \quad \Gamma; \cdot; g_{c2} \vdash t_1' : U_{11} \rightarrow g U_{12} \quad \Gamma; \cdot; g_{c2} \vdash t_2' : U_2'
\]

\[
\Gamma; \cdot; g_{c1} \vdash t_1' t_2' : U_{12} \rightarrow g
\]

(7)

Consider \( g_{c2} \) such that \( g_{c1} \subseteq g_{c2} \) and \( t_2 \) such that \( t_1 \subseteq t_2 \). By definition of term precision \( t_2 \) must have the form \( t_2 = t_1'' t_2'' \) and therefore

\[
(P\text{app}) \quad t_1' \subseteq t_1'' \quad t_2' \subseteq t_2'' \quad t_1' t_2' \subseteq t_1'' t_2''
\]

(8)

Using induction hypotheses on the premises of 7, \( \Gamma; \cdot; g_{c2} \vdash t_1'' : U_{11} \rightarrow g U_{12} \) and \( \Gamma; \cdot; g_{c2} \vdash t_2'' : U_2'' \), where \( U_2' \subseteq U_2'' \), \( U_{11} \rightarrow g U_{12} \subseteq U_{11} \rightarrow g' U_{12} \). By Lemma 4.10, \( U_2'' \subseteq U_1'' \). By definition of precision of types, \( g''_1 \subseteq g''_c \) and \( g \subseteq g' \), therefore by Lemma 4.11, \( g'' \vee g_{c2} \subseteq g''_c \). Then we can use rule \((U\text{app})\) to derive:

\[
(U\text{app}) \quad \Gamma; \cdot; g_{c2} \vdash t_1'' : U_{11} \rightarrow g' U_{12} \quad \Gamma; \cdot; g_{c2} \vdash t_2'' : U_2''
\]

\[
\Gamma; \cdot; g_{c2} \vdash t_1'' t_2'' : U_{12} \sim g'
\]

Using the definition of type precision we can infer that

\[
U_{12} \sim g \subseteq U_{12} \sim g'
\]

and the result holds.

**Case** \((U\text{if})\). Then \( t_1 = \text{if} \ t_2 \) else \( t_2 \) and \( U_1 = (U_1' \sim g U_1') \sim g \). By \((U\text{if})\) we know that:

\[
(U\text{if}) \quad \Gamma; \cdot; g_{c1} \vdash \text{if} \ t_1 \ \text{else} \ t_2' : U_2'
\]

(9)

Consider \( g_{c2} \) such that \( g_{c1} \subseteq g_{c2} \) and \( t_2 \) such that \( t_1 \subseteq t_2 \). By definition of term precision \( t_2 \) must have the form \( t_2 = \text{if} \ t' \ \text{then} \ t_1'' \ \text{else} \ t_2'' \) and therefore

\[
(P\text{if}) \quad t \subseteq t' \quad t_1' \subseteq t_1'' \quad t_2' \subseteq t_2''
\]

(10)
Consider any \( \ell' \) such that \( \ell \sqsubseteq \ell' \). As \( g_{c_1} \sim g \sqsubseteq g_{c_2} \sim g' \) then we can use induction hypotheses on the premises of 9 and derive:

\[
\frac{\Gamma; : g_{c_2} \vdash \ell' : \text{Bool}_g'}{\Gamma; : g_{c_2} \vdash t' : \text{Bool}_g'}
\]

\[
\frac{\Gamma; : g_{c_2} \vdash \ell' : U''_1 \rightarrow U''_2}{\Gamma; : g_{c_2} \vdash t' : U''_2}
\]

Where \( U'_1 \sqsubseteq U''_1 \) and \( U'_2 \sqsubseteq U''_2 \). Using the definition of type precision we can infer that

\[
(\text{Ref}_1') \sim (\text{Ref}_2') \sim g \sqsubseteq (\text{Ref}_1'') \sim (\text{Ref}_2'') \sim g'
\]

and the result holds.

**Case (U::)**. Then \( t_1 = t :: U_1 \). By (U::) we know that:

\[
\frac{\Gamma; : g_{c_1} \vdash t :: U_1' \quad U_1' \sqsubseteq U_1}{\Gamma; : g_{c_1} \vdash t :: U_1}
\]  

(11)

Consider \( g_{c_2} \) such that \( g_{c_1} \sqsubseteq g_{c_2} \) and \( t_2 \) such that \( t_1 \sqsubseteq t_2 \). By definition of term precision \( t_2 \) must have the form \( t_2 = t' :: U_2 \) and therefore

\[
\frac{\Gamma; : g_{c_2} \vdash t' :: U_2' \quad U_2' \sqsubseteq U_2}{\Gamma; : g_{c_2} \vdash t' :: U_2}
\]  

(12)

Using induction hypotheses on the premises of 11, \( \Gamma; : g_{c_1} \vdash t' : U_2' \) where \( U_1' \sqsubseteq U_2' \). We can use rule (U::) and Lemma 4.10 to derive:

\[
\frac{\Gamma; : g_{c_2} \vdash t' :: U_2' \quad U_2' \sqsubseteq U_2}{\Gamma; : g_{c_2} \vdash t' :: U_2}
\]  

Where \( U_1 \sqsubseteq U_2 \) and the result holds.

**Case (Uref).** Then \( t_1 = \text{Ref}_U t \) and \( U_1 = \text{Ref}_{g_{c_1}} U \). By (Uref) we know that:

\[
\frac{\Gamma; : g_{c_1} \vdash t :: U_1' \quad U_1' \sqsubseteq U \quad g_{c_1} \sim \text{label}(U)}{\Gamma; : g_{c_1} \vdash \text{Ref}_U t :: \text{Ref}_{\perp} U}
\]  

(13)

Consider \( g_{c_2} \) such that \( g_{c_1} \sqsubseteq g_{c_2} \) and \( t_2 \) such that \( t_1 \sqsubseteq t_2 \). By definition of term precision \( t_2 \) must have the form \( t_2 = \text{Ref}_{U'} t' \) and therefore

\[
\frac{\Gamma; : g_{c_2} \vdash t' :: U_2' \quad U_2' \sqsubseteq U' \quad g_{c_2} \sim \text{label}(U')}{\Gamma; : g_{c_2} \vdash \text{Ref}_{U'} t' :: \text{Ref}_{\perp} U'}
\]  

(14)

Using induction hypotheses on the premises of 13, we can use rule (Uref) and Lemma 4.10 and 4.12 to derive:

\[
\frac{\Gamma; : g_{c_2} \vdash t' :: U_2'' \quad U_2'' \sqsubseteq U' \quad g_{c_2} \sim \text{label}(U')}{\Gamma; : g_{c_2} \vdash \text{Ref}_{U'} t' :: \text{Ref}_{\perp} U'}
\]

Where \( U \sqsubseteq U' \) and \( U_1' \sqsubseteq U_1'' \). Using the the definition of type precision we can infer that

\[
\text{Ref}_{\perp} U \sqsubseteq \text{Ref}_{\perp} U'
\]

and the result holds.

**Case (Uderefl).** Then \( t_1 = !t \) and \( U_1 = U \sim g \). By (Uderefl) we know that:

\[
\frac{\Gamma; : g_{c_1} \vdash t :: \text{Ref}_{g} U}{\Gamma; : g_{c_1} \vdash t :: U \sim g}
\]  

(15)
Consider \( g_{c2} \) such that \( g_{c1} \subseteq g_{c2} \) and \( t_2 \) such that \( t_1 \subseteq t_2 \). By definition of term precision \( t_2 \) must have the form \( !t' \) and therefore

\[
(Pderef) \quad \frac{t \subseteq t'}{!t \subseteq !t'}
\]

(16)

Using induction hypotheses on the premises of 15, we can use rule \((U\text{deref})\) to derive:

\[
(U\text{deref}) \quad \frac{\Gamma; \vdash g_{c2} \vdash t' : \text{Ref}_g U'}{\Gamma; \vdash g_{c2} \vdash !t' : U' \triangleright g'}
\]

Where \( g \subseteq g' \) and \( U \subseteq U' \). Using the premise of 16 and the definition of type precision we can infer that

\[
U \triangleright g \subseteq U' \triangleright g'
\]

and the result holds.

Case \((U\text{asgn})\). Then \( t_1 = t'_1 := t'_2 \) and \( U_1 = \text{Unit}_\bot \). By \((U\text{asgn})\) we know that:

\[
\frac{\Gamma; \vdash g_{c1} \vdash t'_1 : \text{Ref}_g U'_1 \quad \Gamma; \vdash g_{c1} \vdash t'_2 : U'_2}{U'_2 \leq U'_1 \quad g \triangleright g_{c1} \leq \text{label}(U'_1)}
\]

(17)

Using induction hypotheses on the premises of 17, \( \Gamma; \vdash g_{c2} \vdash t'_1 : \text{Ref}_g U''_1 \) and \( \Gamma; \vdash g_{c2} \vdash t'_2 : U''_2 \), where \( \text{Ref}_g U'_1 \subseteq \text{Ref}_g U''_1 \) and \( U'_2 \subseteq U''_2 \). By definition of precision on types and Lemma 4.10, \( U''_2 \leq U''_1 \). Also, as, \( g \subseteq g' \) and \( U'_1 \subseteq U''_1 \), by Lemma 4.11, \( g' \triangleright g_{c2} \leq \text{label}(U'_1) \). Then we can use rule \((U\text{asgn})\) to derive:

\[
\frac{\Gamma; \vdash g_{c2} \vdash t'_1 : \text{Ref}_g U''_1 \quad \Gamma; \vdash g_{c2} \vdash t'_2 : U''_2}{U''_2 \leq U''_1 \quad g' \triangleright g_{c2} \leq \text{label}(U''_1)}
\]

\[
(U\text{asgn}) \quad \frac{\Gamma; \vdash g_{c2} \vdash t'_1 := t'_2 : \text{Unit}_\bot}{U''_1 \leq U''_2 \quad g \triangleright g_{c1} \leq \text{label}(U''_1)}
\]

(18)

Using the definition of type precision we can infer that

\[
\text{Unit}_\bot \subseteq \text{Unit}_\bot
\]

and the result holds. \( \Box \)
5 GRADUALIZING THE DYNAMIC SEMANTICS

In this section we present the formalization of the evidences for GSLRef. Section 5.1 presents the structure of evidence and the abstraction and concretization functions. In section 5.2, we show how to calculate the initial evidence. In particular we give definition for the initial evidence of consistent judgments for labels and types. In section 5.2, we present how to evolve evidence. We define the consistent transitivity operator, the meet operator and join of evidences. In section 5.4, we present the algorithmic definitions of initial evidence and consistent transitivity. Finally, in section 5.5, we present some of the proofs of the propositions for evidence presented.

5.1 Precise Evidence for Consistent Security Judgments

Definition 5.1 (Interval). An interval is a bounded unknown label \([\ell_1, \ell_2]\) where \(\ell_1\) is the upper bound and \(\ell_2\) is the lower bound.

\[
\ell \in \text{LABEL}^2 \\
\ell ::= [\ell, \ell] \quad \text{(interval)}
\]

Definition 5.2 (Interval Concretization). Let \(\gamma_1 : \text{LABEL}^2 \rightarrow \mathcal{P} \text{(LABEL)}\) be defined as follows:

\[
\gamma_1([\ell_1, \ell_2]) = \{\ell \mid \ell \in \text{LABEL}, \ell_1 \leq \ell \leq \ell_2\}
\]

We can only concretize valid intervals:

Definition 5.3 (Valid Gradual Label).

\[
\ell_1 \leq \ell_2 \quad \text{valid}([\ell_1, \ell_2])
\]

Definition 5.4 (Label Evidence Concretization). Let \(\gamma_{\ell_\ell} : \text{LABEL}^4 \rightarrow \mathcal{P} \text{(LABEL)}^2\) be defined as follows:

\[
\gamma_{\ell_\ell}(\langle \ell_1, \ell_2 \rangle) = \{\langle \ell_1, \ell_2 \rangle \mid \ell_1 \in \gamma_1(\ell_1), \ell_2 \in \gamma_1(\ell_2)\}
\]

Definition 5.5 (Interval Abstraction). Let \(\alpha : \mathcal{P} \text{(LABEL)} \rightarrow \text{LABEL}^2\) be defined as follows:

\[
\alpha_\ell(\emptyset) \text{ is undefined} \\
\alpha_\ell([\ell_i]) = [\lor \ell_i, \land \ell_i] \text{ otherwise}
\]

Definition 5.6 (Label Evidence Abstraction). Let \(\alpha_{\ell_\ell} : \mathcal{P} \text{(LABEL)}^2 \rightarrow \text{LABEL}^4\) be defined as follows:

\[
\alpha_{\ell_\ell}(\emptyset) \text{ is undefined} \\
\alpha_{\ell_\ell}(\langle [\ell_{11}, \ell_{21}] \rangle) = \langle \alpha_\ell([\ell_{11}]), \alpha_\ell([\ell_{21}]) \rangle \text{ otherwise}
\]

Definition 5.7 (Type Evidence). An evidence type is a gradual type labeled with an interval:

\[
E \in \text{GETYPE}, \quad \ell \in \text{LABEL}^2 \\
E ::= \text{Bool}, \ell \mid E \rightarrow_{\ell} E, \ell \mid \text{Ref}, \ell \mid \text{Unit}, \ell \quad \text{(evidence types)}
\]

Definition 5.8 (Type Evidence Concretization). Let \(\gamma_E : \text{GETYPE} \rightarrow \mathcal{P} \text{(TYPE)}\) be defined as follows:

\[
\gamma_E(\text{Bool}) = \{\text{Bool}, \ell \mid \ell \in \gamma_1(\ell)\} \\
\gamma_E(E_1 \rightarrow_{\ell} E_2) = \gamma_E(E_1) \rightarrow_{\gamma_1(\ell)} \gamma_E(E_2) \\
\gamma_E(\text{Ref}, \ell, E) = \{\text{Ref}, \ell, S \mid \ell \in \gamma_1(\ell), S \in \gamma_E(E)\}
\]

where \(\rightarrow\) is the set of all possible combinations of function types, using each member of the sets obtained by the \(\gamma_E\) and \(\gamma_1\) functions.
Type-Driven Gradual Security with References: Complete Definitions and Proofs

Definition 5.9 (Evidence Concretization). Let \( \gamma_{\ell_f} : \text{GETYPE}^2 \rightarrow \mathcal{P}(\text{TYPE}^2) \) be defined as follows:
\[
\gamma_{\ell_f}((E_1, E_2)) = \{ (S_1, S_2) \mid S_1 \in \gamma_E(E_1), S_2 \in \gamma_E(E_2) \}
\]

Definition 5.10 (Type Evidence Abstraction). Let the abstraction function \( \alpha_E : \mathcal{P}(\text{TYPE}) \rightarrow \text{GETYPE} \) be defined as:
\[
\alpha_E(\{ \text{Bool}_{\ell_i} \}) = \text{Bool}_{\alpha_{\ell_i}(\ell_i)}
\]
\[
\alpha_E(\{ S_{r_i} \}) = \alpha_E(\{ \overrightarrow{S_{r_i}} \}) \left( \alpha_{\ell_i}(\ell_i) \right) \alpha_E(\{ \overrightarrow{S_{r_i}} \})
\]
\[
\alpha_E(\{ \text{Ref}_{\ell_i} \}) = \text{Ref}_{\alpha_{\ell_i}(\ell_i)} \alpha_E(\{ \overrightarrow{S_{r_i}} \})
\]

\( \alpha_E(\hat{S}) \) is undefined otherwise

Definition 5.11 (Evidence Abstraction). Let \( \alpha_{\ell} : \mathcal{P}(\text{TYPE}^2) \rightarrow \text{GETYPE}^2 \) be defined as follows:
\[
\alpha_{\ell}(\emptyset) \text{ is undefined}
\]
\[
\alpha_{\ell}(\{ \overrightarrow{S_{r_i}} \}) = \langle \alpha_E(\{ \overrightarrow{S_{r_i}} \}), \alpha_E(\{ \overrightarrow{S_{r_i}} \}) \rangle \text{ otherwise}
\]

We can only abstract valid sets of security types, i.e. in which elements only defer by security labels.

Definition 5.12 (Valid Type Sets).

\[
\begin{array}{ccc}
\text{valid}(\{ \text{Bool}_{\ell_i} \}) & \text{valid}(\{ S_{r_i} \}) & \text{valid}(\{ \text{Ref}_{\ell_i} \}) \\
\text{valid}(\{ \overrightarrow{S_{r_i}} \}) & \text{valid}(\{ \overrightarrow{S_{r_i}} \}) & \text{valid}(\{ \overrightarrow{S_{r_i}} \}) \\
\text{valid}(\{ \text{Unit}_{\ell_i} \})
\end{array}
\]

Proposition 5.13 (\( \alpha_{\ell} \) is Sound). If \( \hat{\ell} \) is not empty, then \( \hat{\ell} \subseteq \gamma_{\ell}(\alpha_{\ell}(\hat{\ell})) \).

Proposition 5.14 (\( \alpha_{\ell} \) is Optimal). If \( \hat{\ell} \) is not empty, and \( \hat{\ell} \subseteq \gamma_{\ell}(i) \) then \( \alpha_{\ell}(\hat{\ell}) \subseteq i \).

Proposition 5.15 (\( \alpha_E \) is Sound). If valid(\( \hat{S} \)) then \( \hat{S} \subseteq \gamma_E(\alpha_E(\hat{S})) \).

Proposition 5.16 (\( \alpha_E \) is Optimal). If valid(\( \hat{S} \)) and \( \hat{S} \subseteq \gamma_E(E) \) then \( \alpha_E(\hat{S}) \subseteq E \).

With concretization of security type, we can now define security type precision.

Definition 5.17 (Interval and Type Evidence Precision).

1. \( t_1 \) is less imprecise than \( t_2 \), notation \( t_1 \subseteq t_2 \), if and only if \( \gamma_{\ell_f}(t_1) \subseteq \gamma_{\ell_f}(t_2) \); inductively:
\[
[\ell_3, \ell_4] \subseteq [\ell_1, \ell_2] \quad \ell_3 \leq \ell_1 \quad \ell_2 \leq \ell_4
\]

2. \( E_1 \) is less imprecise than \( E_2 \), notation \( E_1 \subseteq E_2 \), if and only if \( \gamma_E(E_1) \subseteq \gamma_E(E_2) \); inductively:
\[
E_{11} \subseteq E_{21} \quad E_{12} \subseteq E_{22} \quad E_{1i} \subseteq E_{2i} \quad E_{1i} \subseteq E_{2i}
\]

\[
\text{Ref}_{t_1} \subseteq \text{Ref}_{t_2} \quad \text{Ref}_{t_1} \subseteq \text{Ref}_{t_2} \quad E_{1i} \subseteq E_{2i}
\]
5.2 Initial evidence

With the definition of concretization and abstraction we can now define the initial evidence of label ordering and subtyping:

**Definition 5.18 (Initial Evidence of label ordering).** Let \( F_1 : \text{LABEL}^n \rightarrow \text{LABEL} \) and \( F_2 : \text{LABEL}^m \rightarrow \text{LABEL} \) be functions over labels. The initial evidence of the judgment \( F_1(\overline{g_l}) \triangleq F_2(\overline{g_j}) \), notation \( \mathcal{G}[F_1(\overline{g_l}) \triangleq F_2(\overline{g_j})] \), is defined as follows:

\[
\mathcal{G}[F_1(g_1, \ldots g_n) \triangleq F_2(g_{n+1}, \ldots g_{n+m})] = \
\alpha_t((\langle F_1(\overline{f_l}), F_2(\overline{f_j}) \rangle) | (\overline{f_l}) \in \gamma^n(\overline{g_l}[1/n]),
\langle \overline{f_j} \rangle \in \gamma^m(\overline{g_j}[n+1/m]) | F_1(\overline{f_l}) \triangleq F_2(\overline{f_j}))
\]

Suppose \( F_1 = F_{11} \)

**Definition 5.19 (Initial Evidence of subtyping).** Let \( F_1 : \text{TYPE}^n \rightarrow \text{TYPE} \) and \( F_2 : \text{TYPE}^m \rightarrow \text{TYPE} \) be functions over types. The initial evidence of the judgment \( F_1(\overline{U_l}) \triangleq F_2(\overline{U_j}) \), notation \( \mathcal{G}[F_1(\overline{U_l}) \triangleq F_2(\overline{U_j})] \), is defined as follows:

\[
\mathcal{G}[F_1(U_1, \ldots U_n) \triangleq F_2(U_{n+1}, \ldots U_{n+m})] = \
\alpha_t((\langle F_1(\overline{s_l}), F_2(\overline{s_j}) \rangle) | (\overline{s_l}) \in \gamma^n(\overline{U_l}[1/n]),
\langle \overline{s_j} \rangle \in \gamma^m(\overline{U_j}[n+1/m]) | F_1(\overline{s_l}) \triangleq F_2(\overline{s_j}))
\]

**Proposition 5.20.** [Elaboration preserves typing] Consider \( \Gamma; \Sigma; g_c \vdash t : U \) then if \( \Gamma; \Sigma; g_c \vdash t \leadsto t' : U \), and \( e = \delta_{\text{U}}(\ell, c) \), then \( \Gamma; \Sigma; e; g_c \vdash t' : U \)

**Proof.** Straightforward induction on type \( U \). \( \square \)

5.3 Evolving evidence: Consistent Transitivity

Now that we know how to extract initial evidence from consistent judgments, we need a way to combine evidences to use during program evaluation, i.e. we need to find a way to evolve evidence. We define consistent transitivity for label ordering and subtyping, \( \circ \leq \) and \( \circ \preceq \): respectively, to combine evidences as follows:

**Definition 5.21 (Consistent transitivity for label ordering).** Let function \( \circ \leq : \text{INTERVAL}^2 \times \text{INTERVAL}^2 \rightarrow \text{LABEL}^2 \) be defined as:

\[
\langle t_{11}, t_{12} \rangle \circ \leq \langle t_{21}, t_{22} \rangle = \alpha_t((\langle \ell_{11}, \ell_{22} \rangle \in \gamma_t((t_{11}, t_{22}))) | \exists \ell \in \gamma(t_{12}) \cap \gamma(t_{21}). \ell_{11} \preceq \ell \wedge \ell \preceq \ell_{22})
\]

**Proposition 5.22.** Suppose \( e_1 \vdash F_1(\overline{g_l}) \preceq F_2(\overline{g_j}) \) and \( e_2 \vdash F_2(\overline{g_j}) \preceq F_3(\overline{g_k}) \).

If \( e_1 \circ \leq e_2 \) is defined, then \( e_1 \circ \leq e_2 \vdash F_1(\overline{g_l}) \preceq F_3(\overline{g_k}) \)

**Proposition 5.23.** \( \gamma(t_{11} \cap t_{12}) = \gamma(t_{11}) \cap \gamma(t_{12}) \).

where \( t \cap t' = \alpha(\gamma(t) \cap \gamma(t')) \).

**Proposition 5.24.** \( \langle t_{11}, t_{12} \rangle \circ \preceq \langle t_{21}, t_{22}, t_{3} \rangle = \Delta \preceq \langle t_{11}, t_{21} \cap t_{22}, t_{3} \rangle \)

where

\[
\Delta \preceq \langle t_{11}, t_{22}, t_{3} \rangle = \alpha_t((\langle \ell_{11}, \ell_{3} \rangle \in \gamma_t((t_{11}, t_{3}))) | \exists \ell \in \gamma(t_{12}). \ell_{11} \preceq \ell_{2} \wedge \ell_{2} \preceq \ell_{3}))
\]
We deduce evidence for consistent transitivity for subtyping: where
\[ \langle E_{11}, E_{12} \rangle \vdash F_1(U_{1i}) <: F_2(U_{1j}) \quad \langle E_{21}, E_{22} \rangle \vdash F_2(U_{1j}) <: F_3(U_{2k}) \]

We deduce evidence for consistent transitivity for subtyping:
\[ \langle E_{11}, E_{12} \rangle \circ <: \langle E_{21}, E_{22} \rangle \vdash F_1(U_{1i}) <: F_3(U_{2k}) \]
where \( \circ <: \text{ETYPE}^2 \times \text{ETYPE}^2 \rightarrow \text{ETYPE}^2 \) is defined as:
\[ \langle E_{11}, E_{12} \rangle \circ <: \langle E_{21}, E_{22} \rangle = \alpha_{t}(\{(S_{11}, S_{22}) \in \gamma_{t}(\langle E_{11}, E_{22} \rangle) \mid \exists S_{2} \in \gamma_{E}(E_{22}) \cap \gamma_{E}(E_{21}).S_{1} <: S \land S <: S_{22}) \}

Proposition 5.26. \( \gamma_{E}(E_{1} \cap E_{2}) = \gamma_{E}(E_{1}) \cap \gamma_{E}(E_{2}) \).

Then following AGT,

Proposition 5.27.
\[ \langle E_{1}, E_{21} \rangle \circ <: \langle E_{22}, E_{3} \rangle = \delta <: \langle E_{1}, E_{21} \cap E_{22}, E_{3} \rangle \]
where
\[ \delta <: \langle E_{1}, E_{2}, E_{3} \rangle = \alpha_{t}(\{\langle S_{1}, S_{3} \rangle \in \gamma_{t}(\langle E_{1}, E_{3} \rangle) \mid \exists S_{2} \in \gamma_{E}(E_{2}) \cap \gamma_{E}(E_{21}).S_{1} <: S_{2} \land S_{2} <: S_{31}) \}

Definition 5.28 (Intervals join).
\[ [\ell_{1}, \ell_{2}] \sim [\ell_{3}, \ell_{4}] = [\ell_{1} \lor \ell_{3}, \ell_{2} \lor \ell_{4}] \]

Definition 5.29 (Evidence label join).
\[ \langle t_{1}, t_{2} \rangle \sim \langle t_{3}, t_{4} \rangle = \langle t_{1} \lor t_{3}, t_{2} \lor t_{4} \rangle \]

Definition 5.30.
\[ \text{Bool}_{t_{1}} \sim t_{2} = \text{Bool}_{t_{1} \lor t_{2}} \]
\[ E_{1} \overset{t_{2}}{\rightarrow} t_{1} E_{2} \sim \overset{t_{3}}{t_{1} \lor t_{2}} E_{2} \]
\[ \text{Ref}_{t_{1}} E \sim \overset{t_{2}}{t_{1} \lor t_{2}} E \]

Definition 5.31.
\[ \langle E_{1}, E_{2} \rangle \sim \langle t_{1}, t_{2} \rangle = \langle E_{1} \lor t_{1}, E_{2} \lor t_{2} \rangle \]

Proposition 5.32. If \( \varepsilon_{S} \vdash U_{1} \leq U_{2} \) and \( \varepsilon_{1} \vdash g_{1} \leq g_{2} \) then \( \varepsilon_{S} \sim \varepsilon_{1} \vdash U_{1} \sim g_{1} <: U_{2} \sim g_{2} \)

5.4 Algorithmic definitions

This section gives algorithmic definitions of consistent transitivity and initial evidence for label ordering and subtyping.

5.4.1 Label Evidences.

Definition 5.33 (Intervals join).
\[ [\ell_{1}, \ell_{2}] \sim [\ell_{3}, \ell_{4}] = [\ell_{1} \lor \ell_{3}, \ell_{2} \lor \ell_{4}] \]

Definition 5.34 (Intervals meet).
\[ [\ell_{1}, \ell_{2}] \searrow [\ell_{3}, \ell_{4}] = [\ell_{1} \land \ell_{3}, \ell_{2} \land \ell_{4}] \]
Definition 5.35. Let $F_1 : \text{LABEL}^n \to \text{LABEL}$ and $F_2 : \text{LABEL}^m \to \text{LABEL}$. The initial evidence for consistent judgment $F_1(\overline{g_1}) \leq F_2(\overline{g_2})$ is defined as follows:

$$
\begin{align*}
\text{bounds}(?) &= [\bot, \top] \\
\text{bounds}(\ell) &= [\ell, \ell] \\
\text{bounds}(x_1 \lor x_2) &= \text{bounds}(x_1) \lor \text{bounds}(x_2) \\
\text{bounds}(x_1 \land x_2) &= \text{bounds}(x_1) \land \text{bounds}(x_2) \\
\text{bounds}(x_1 \sqcap x_2) &= \text{bounds}(x_1) \sqcap \text{bounds}(x_2) \\
\text{bounds}(F_1(\overline{x_1}) \lor F_2(\overline{x_2})) &= \text{bounds}(F_1(\overline{x_1})) \lor \text{bounds}(F_2(\overline{x_2})) \\
\text{bounds}(F_1(\overline{x_1}) \land F_2(\overline{x_2})) &= \text{bounds}(F_1(\overline{x_1})) \land \text{bounds}(F_2(\overline{x_2})) \\
\text{bounds}(F_1(\overline{x_1}) \sqcap F_2(\overline{x_2})) &= \text{bounds}(F_1(\overline{x_1})) \sqcap \text{bounds}(F_2(\overline{x_2}))
\end{align*}
$$

where $F_1 : \text{LABEL}^n \to \text{LABEL}$ and $F_2 : \text{LABEL}^m \to \text{LABEL}$.

$$
\mathcal{J}(F_1(\overline{g_1}, \ldots, g_n) \leq F_2(\overline{g_{n+1}}, \ldots, g_{n+m})) = ([\ell_1, \ell_2], [\ell_1 \lor \ell_2])
$$

The algorithmic definition of meet:

$$
[\ell_1, \ell_2] \cap [\ell_3, \ell_4] = [\ell_1 \lor \ell_3, \ell_2 \land \ell_4] \quad \text{if valid}([\ell_1 \lor \ell_3, \ell_2 \land \ell_4]) \\
\text{else } \text{undefined otherwise}
$$

We calculate the algorithmic definition of $\Delta^\leq$:

$$
\Delta^\leq([\ell_1, \ell_2], [\ell_3, \ell_4], [\ell_5, \ell_6]) = ([\ell_1 \land \ell_3 \land \ell_5], [\ell_2 \lor \ell_4 \lor \ell_6])
$$

5.4.2 Type Evidences. We define a function $\text{liftP()}$ to transform functions over types into functions over labels. Also we define function $\text{invert()}$ to invert the operator on types, used in the domain and latent effect of function types. Finally we define function $\text{tomeet()}$ to transform type operators into meets, given the invariant property of references.

We start defining a pattern of operations:

Definition 5.36 (Operation pattern).

$$
\begin{align*}
P^T &\in \text{GPattern}, P^\ell \in \text{LPattern} \\
P^T &::= \_ | P^T \text{ op}^T P^T \quad \text{(pattern on types)} \\
op^T &::= \lor | \land | \sqcap \quad \text{(operations on types)} \\
P^\ell &::= \_ | P^\ell \text{ op}^\ell P^\ell \quad \text{(pattern on labels)} \\
op\ell &::= \lor | \land | \sqcap \quad \text{(operations on labels)}
\end{align*}
$$
We use case-based analysis to calculate the algorithmic rules for the initial evidence of consistent subtyping on gradual security types:

\[
\begin{align*}
\mathcal{G}\left[ \text{liftP}(G_1)(\overline{t_1}) <: \text{liftP}(G_2)(\overline{t_2}) \right] & = \langle t_1, t_2 \rangle \\
\mathcal{G}\left[ G_1(\text{Bool}_y) \triangleleft G_2(\text{Bool}_y) \right] & = \langle \text{Bool}_y, \text{Bool}_y \rangle \\
\mathcal{G}\left[ \text{invert}(G_2)(\overline{u_1}) <: \text{invert}(G_1)(\overline{u_2}) \right] & = \langle E_{21}, E_{11} \rangle \\
\mathcal{G}\left[ [G_1(\overline{u_1})] <: G_2(\overline{u_2}) \right] & = \langle E_{12}, E_{22} \rangle \\
\mathcal{G}\left[ \text{liftP}(\text{invert}(G_2))(\overline{t_2}) <: \text{liftP}(\text{invert}(G_1))(\overline{t_1}) \right] & = \langle t_{12}, t_{11} \rangle \\
\mathcal{G}\left[ [G_1(U_{11})] <: G_2(U_{12}) \right] & = \langle E_{11} \xrightarrow{t_{11}}_{t_{12}} E_{12}, E_{21} \xrightarrow{t_{21}}_{t_{22}} E_{22} \rangle \\
\mathcal{G}\left[ \text{liftP}(G_1)(\overline{t_1}) <: \text{liftP}(G_2)(\overline{t_2}) \right] & = \langle t_1, t_2 \rangle \\
\mathcal{G}\left[ \text{tomeet}(G_1)(\overline{u_1}) <: \text{tomeet}(G_2)(\overline{u_2}) \right] & = \langle E_1, E_2 \rangle \\
\mathcal{G}\left[ \text{tomeet}(G_2)(\overline{u_1}) <: \text{tomeet}(G_1)(\overline{u_1}) \right] & = \langle E_2', E_1' \rangle \\
\mathcal{G}\left[ G_1(\text{Ref}_y)(\overline{u_1}) <: G_2(\text{Ref}_y)(\overline{u_2}) \right] & = \langle \text{Ref}_y \ E_1 \cap E_1', \text{Ref}_y \ E_2 \cap E_2' \rangle \\
\end{align*}
\]

where \( G_1 : \text{GLABEL}^n \rightarrow \text{GLABEL} \) and \( G_2 : \text{GLABEL}^m \rightarrow \text{GLABEL} \), and \( G_1(x_1, \ldots, x_n) = P_1^T(x_1, \ldots, x_n) \), \( G_2(x_1, \ldots, x_n) = P_2^T(x_1, \ldots, x_m) \).

\[
\mathcal{G} \land (F(\overline{t_1}, \ldots, \overline{u_n})) = \mathcal{G}[F(\overline{t_1}, \ldots, \overline{u_n}) <: F(\overline{t_1}, \ldots, \overline{u_n})]
\]

We calculate a recursive meet operator for gradual types:

\[
\begin{align*}
\text{Bool}_y \cap \text{Bool}_x & = \text{Bool}_{y \cap x} \\
(E_{11} \xrightarrow{t_1}_{t_2} E_{12}) \cap (E_{21} \xrightarrow{t_2'}_{t_2''} E_{22}) & = (E_{11} \cap E_{21}) \xrightarrow{t_2' t_2''}_{t_2''} (E_{12} \cap E_{22}) \\
\text{Ref}_y \ E_1 \cap \text{Ref}_x \ E_2 & = \text{Ref}_{y \cap x} \ E_1 \cap \ E_2 \\
U \cap U' & \text{ undefined otherwise}
\end{align*}
\]

We calculate a recursive definition for \( \text{liftP} \) by case analysis on the structure of the second argument,
\[ \Delta^{<}(\text{Bool}_{i_1}, \text{Bool}_{i_2}, \text{Bool}_{i_3}) = \langle \text{Bool}_{i_1}', \text{Bool}_{i_2}' \rangle \]

\[ \Delta^{<}(\text{Ref}_{i_1}, E_1, \text{Ref}_{i_2} E_2, \text{Ref}_{i_3} E_3) = \langle \text{Ref}_{i_1}' E_1', \text{Ref}_{i_2}' E_2', \text{Ref}_{i_3}' E_3' \rangle \]

5.4.3 Evidence inversion functions. The evidence inversion functions are defined as follows

\[ \text{ilbl}(\langle \text{Bool}_{i_1}, \text{Bool}_{i_2} \rangle) = \langle i_1, i_2 \rangle \]

\[ \text{ilbl}(\langle \text{Unit}_{i_1}, \text{Unit}_{i_2} \rangle) = \langle i_1, i_2 \rangle \]

\[ \text{ilbl}(\langle \text{Ref}_{i_1}, U_{i_1}, \text{Ref}_{i_2} U_{i_2} \rangle) = \langle i_1, i_2 \rangle \]

\[ \text{ilbl}(\langle E_1 \overrightarrow{i_2} E_2, E'_1 \overrightarrow{i_2'} E'_2 \rangle) = \langle i_1, i_1' \rangle \]

\[ \text{iref}(\langle \text{Ref}_{i_1}, E_1, \text{Ref}_{i_2} E_2 \rangle) = \langle E_1, E_2 \rangle \]

\[ \text{iref}(\langle E_1, E_2 \rangle) = \text{undefined otherwise} \]

\[ \text{idom}(\langle E_1 \overrightarrow{i_2} E_2, E'_1 \overrightarrow{i_2'} E'_2 \rangle) = \langle E'_1, E_1 \rangle \]

\[ \text{idom}(\langle E_1, E_2 \rangle) = \text{undefined otherwise} \]

\[ \text{icod}(\langle E_1 \overrightarrow{i_2} E_2, E'_1 \overrightarrow{i_2'} E'_2 \rangle) = \langle E_2, E'_2 \rangle \]

\[ \text{icod}(\langle E_1, E_2 \rangle) = \text{undefined otherwise} \]

5.5 Proofs

Proposition 5.13 (\(\alpha_i\) is Sound). If \(\hat{\ell}\) is not empty, then \(\hat{\ell} \subseteq \gamma_i(\alpha_i(\hat{\ell}))\).

Proof. Suppose \(\hat{\ell} = \{ \overline{\ell}_i \}\). By definition of \(\alpha_{i\ell} \), \(\alpha_{i}(\{ \overline{\ell}_i \}) = [\overline{\ell}_i \wedge \overline{\ell}_i] \). Therefore

\[ \gamma_i(\alpha_i(\{ \overline{\ell}_i \})) = \{ \ell \mid \ell \in \text{LABEL}, \overline{\ell}_i \leq \ell \leq \overline{\ell}_i \} \]

And it is easy to see that if \(\ell \in \{ \overline{\ell}_i \}\), then \(\ell \in \gamma_i(\alpha_i(\{ \overline{\ell}_i \}))\), and therefore the result holds. \(\Box\)

Proposition 5.14 (\(\alpha_E\) is Optimal). If \(\hat{\ell}\) is not empty, and \(\hat{\ell} \subseteq \gamma_i(\ell)\) then \(\alpha_i(\hat{\ell}) \subseteq i\).

Proof. By case analysis on the structure of \(i\). If \(i = [\ell_1, \ell_2]\), \(\gamma_{i\ell}(\ell) = \{ \ell \mid \ell \in \text{LABEL}, \ell_1 \leq \ell \leq \ell_2 \} \); \(\hat{\ell} \subseteq \{ \ell \mid \ell \in \text{LABEL}, \ell_1 \leq \ell \leq \ell_2 \}\), \(\hat{\ell} \neq \emptyset\) implies \(\alpha_{i\ell}(\hat{\ell}) = [\ell_3, \ell_4]\), where \(\ell_1 \leq \ell_3\) and \(\ell_4 \leq \ell_2\), therefore \([\ell_3, \ell_4] \subseteq i\) (if \(\hat{\ell} = \emptyset\), \(\alpha_{i\ell}(\hat{\ell})\) is undefined). \(\Box\)

Proposition 5.15 (\(\alpha_E\) is Sound). If \(\text{valid}(\hat{S})\) then \(\hat{S} \subseteq \gamma_E(\alpha_E(\hat{S}))\).
Type-Driven Gradual Security with References: Complete Definitions and Proofs

Proof. By well-founded induction on $\hat{S}$. Similar to Prop 4.4.

Proposition 5.16 ($\alpha_E$ is optimal). If valid$(\hat{S})$ and $\hat{S} \subseteq \gamma_E(\hat{E})$ then $\alpha_E(\hat{S}) \subseteq E$.

Proof. By induction on the structure of $U$. Similar to Prop 4.5.

Proposition 5.23. $\gamma_i(t_1 \cap t_2) = \gamma_i(t_1) \cap \gamma_i(t_2)$.

Proof.

Let $\ell \in \gamma_i(t_1) \cap \gamma_i(t_1)$. We now that $\gamma_i(t_1 \cap t_2)$ is defined. Suppose $t_1 = [\ell_1, \ell_2]$ and $t_2 = [\ell_3, \ell_4]$.

Therefore $t_1 \cap t_2 = [\ell_1 \wedge \ell_3, \ell_2 \wedge \ell_4]$.

But $\gamma_i(t_1) \cap \gamma_i(t_1) = \{ \ell \mid \ell \in \text{Label}, \ell_1 \ll \ell \ll \ell_2 \} \cap \{ \ell \mid \ell \in \text{Label}, \ell_3 \ll \ell \ll \ell_4 \}$. Which is equivalent to $\{ \ell \mid \ell \in \text{Label}, \ell_1 \ll \ell \ll \ell_3 \ll \ell \ll \ell_4 \}$, equivalent to $\{ \ell \mid \ell \in \text{Label}, \ell_1 \wedge \ell_3 \ll \ell \ll \ell_2 \wedge \ell_4 \}$. Which is by definition $\gamma_i([\ell_1 \wedge \ell_3, \ell_2 \wedge \ell_4])$, and the result holds.

Proof. Follows directly from the definition of consistent transitivity and Prop 5.23.

Proposition 5.26. $\gamma_E(E_1 \cap E_2) = \gamma_E(E_1) \cap \gamma_E(E_2)$.

Proof. By induction on evidence types $\epsilon_1$ and $\epsilon_2$ and Prop 5.23.

Proposition 5.27.

\[ \langle E_1, E_2 \rangle \circ^< \langle E_{22}, E_3 \rangle = \Delta^< \langle E_1, E_{21} \cap E_{22}, E_3 \rangle \]

where

\[ \Delta^< \langle E_1, E_2, E_3 \rangle = \alpha_i(\{S_1, S_3 \in \gamma_i(\langle E_1, E_3 \rangle) \mid \exists S_2 \in \gamma_i(\langle E_2 \rangle \cdot S_1: S_2 \wedge S_2 \ll\ll S_3) \}) \]

Proof. Follows directly from the definition of consistent transitivity and Prop 5.26.

Proposition 5.32. If $\epsilon_S \vdash U_1 \ll U_2$ and $\epsilon_1 \vdash g_1 \ll g_2$ then $\epsilon_S \vdash \epsilon_1 \vdash U_1 \ll g_1 \ll U_2 \ll g_2$.

Proof. By induction on types $U_1$ and $U_2$, using the definition of $\ll\ll$; and Proposition 5.43.

Proposition 5.37. $[\ell_1, \ell_2] \wedge [\ell_3, \ell_4] = [\ell_1 \wedge \ell_3, \ell_2 \wedge \ell_4]$

Proof. Follows directly by definition of $\wedge$ and $\gamma$.

Proposition 5.38.

\[ \langle t_1, t_2 \rangle \ll \langle t_1', t_2' \rangle = \langle t_1 \ll t_1', t_2 \ll t_2' \rangle \]

Proof. Follows directly from the definition of consistent join monotonicity and Prop 5.37.

Proposition 5.39.

\[ [\ell_1, \ell_2] \cap [\ell_3, \ell_4] = [\ell_1 \wedge \ell_3, \ell_2 \wedge \ell_4] \]

if $\ell_1 \ll \ell_3 \ll \ell_2 \ll \ell_4$

\[ 1 \cap 1' \text{ undefined otherwise} \]

Proof. By definition of meet:

\[ [\ell_1, \ell_2] \cap [\ell_3, \ell_4] = \alpha_i([\ell' \mid \ell' \in \gamma([\ell_1, \ell_2]) \cap \gamma([\ell_3, \ell_4])] \]

But by definition of intersection on intervals, $\gamma([\ell_1, \ell_2]) \cap \gamma([\ell_3, \ell_4]) = \gamma([\ell_1 \wedge \ell_3, \ell_2 \wedge \ell_4])$ if $\ell_1 \ll \ell_3 \ll \ell_2 \ll \ell_4$ (otherwise the intersection is empty), and the result follows by definition of $\alpha_i$. □
Proposition 5.40.
\[ \ell_1 \preceq \ell_4 \quad \ell_3 \preceq \ell_6 \quad \ell_1 \preceq \ell_6 \]
\[ \Delta^\prec(\{\ell_1, \ell_2\}, [\ell_3, \ell_4], [\ell_5, \ell_6]) = \langle \ell_1, \ell_2 \land \ell_4 \land \ell_6 \rangle \]

Proof. By definition:
\[ \Delta^\prec(\{\ell_1, \ell_2\}, [\ell_3, \ell_4], [\ell_5, \ell_6]) = \alpha_i(\{(\ell_1', \ell_2'), (\ell_3', \ell_4') \in \gamma_i(\{\ell_1, \ell_2\}, [\ell_3, \ell_4]) \} \cup \exists \ell'_i \in \gamma_i([\ell_3, \ell_4]). \ell'_1 \preceq \ell'_2 \preceq \ell'_3 \]

It is easy to see that \( \alpha_i(\{(\ell_1', \ell_2') \}) = [\ell_1, \ell_2 \land \ell_4 \land \ell_6]. \) Similar argument is used to prove that \( \alpha_i(\{(\ell_1', \ell_2') \}) = [\ell_1 \land \ell_3 \land \ell_5 \land \ell_6]. \)

Lemma 5.41. Let \( \ell_i \in \text{LABEL}, \) then \( (\ell_1 \land \ell_2) \lor (\ell_3 \land \ell_4) \preceq (\ell_1 \lor \ell_3) \lor (\ell_2 \lor \ell_4). \)

Proof.
\[
\begin{align*}
(\ell_1 \land \ell_2) \lor (\ell_3 \land \ell_4) &\preceq (\ell_1 \lor (\ell_3 \land \ell_4)) \lor (\ell_2 \lor (\ell_3 \land \ell_4)) \\
&\preceq ((\ell_1 \lor \ell_3) \land (\ell_1 \lor \ell_4)) \lor ((\ell_2 \lor \ell_3) \land (\ell_2 \lor \ell_4)) \\
&\preceq (\ell_1 \lor \ell_3) \land (\ell_2 \lor \ell_4).
\end{align*}
\]

Proposition 5.42. Suppose \( \varepsilon_1 \vdash F_1(\overline{g_l}) \preceq F_2(\overline{g_l}) \) and \( \varepsilon_2 \vdash F_3(\overline{g_k}) \preceq F_3(\overline{g_k}). \)
If \( \varepsilon_1 \preceq \varepsilon_2 \) is defined, then \( \varepsilon_1 \preceq \varepsilon_2 \vdash F_3(\overline{g_k}) \preceq F_3(\overline{g_k}). \)

Proof. Suppose \( \varepsilon_1 = \langle t_{11}, t_{12} \rangle \) and \( \varepsilon_2 = \langle t_{21}, t_{22} \rangle. \) Then by definition of initial evidence:
\[ \langle t_{11}, t_{12} \rangle = \langle [\ell_1, \ell_2], [\ell_3, \ell_4] \rangle \preceq \gamma(\langle F_1(\overline{g_l}) \preceq F_2(\overline{g_l}) \rangle) = \langle t_{11}', t_{12}' \rangle \]
and
\[ \langle t_{21}, t_{22} \rangle = \langle [\ell_5, \ell_6], [\ell_7, \ell_8] \rangle \preceq \gamma(\langle F_3(\overline{g_k}) \preceq F_3(\overline{g_k}) \rangle) = \langle t_{21}', t_{22}' \rangle \]

Suppose that \( \gamma(\langle F_1(\overline{g_l}) \preceq F_3(\overline{g_k}) \rangle) = \langle t_{11}', t_{12}' \rangle. \) We have to prove that \( \langle t_{11}, t_{12} \rangle \circ \prec \langle t_{21}, t_{22} \rangle \subseteq \langle t_{11}', t_{12}' \rangle. \)

If \( \text{bounds}(F_1(\overline{g_l})) = [\ell_1', \ell_2'], \text{bounds}(F_2(\overline{g_l})) = [\ell_3', \ell_4'], \) and \( \text{bounds}(F_3(\overline{g_k})) = [\ell_5', \ell_6'] \) We know that \( \gamma(\langle F_1(\overline{g_l}) \preceq F_2(\overline{g_l}) \rangle) = \langle [\ell_1', \ell_2' \land \ell_4'], [\ell_3' \lor \ell_3', \ell_4'] \rangle. \) Therefore \( \ell_1' \preceq \ell_1, \ell_2' \preceq \ell_2 \land \ell_4', \ell_3' \lor \ell_4' \preceq \ell_3 \)
and \( \ell_4 \preceq \ell_4'. \)

Using the same argument,
\[ \gamma(\langle F_2(\overline{g_l}) \preceq F_3(\overline{g_k}) \rangle) = \langle [\ell_3', \ell_4' \land \ell_6'], [\ell_3' \lor \ell_3', \ell_6'] \rangle. \) Therefore \( \ell_3' \preceq \ell_5, \ell_6' \preceq \ell_4' \land \ell_6', \ell_3' \lor \ell_6' \preceq \ell_7 \)
and \( \ell_8 \preceq \ell_8'. \)

But \( \gamma(\langle F_1(\overline{g_l}) \preceq F_3(\overline{g_k}) \rangle) = \langle [\ell_1', \ell_2' \land \ell_4'], [\ell_1' \lor \ell_5', \ell_6'] \rangle \) and
\[ \langle t_{11}, t_{12} \rangle \circ \prec \langle t_{21}, t_{22} \rangle = \Delta^\prec \langle t_{11}, t_{12} \cap t_{21}, t_{22} \rangle = \]
\[ \Delta^\prec(\{\ell_1, \ell_2\}, [\ell_3 \lor \ell_5, \ell_4 \land \ell_6], [\ell_7, \ell_8]) \]
\[ = \langle [\ell_1, \ell_2 \land \ell_4 \land \ell_6 \land \ell_8], [\ell_1 \lor \ell_3 \lor \ell_5 \lor \ell_7, \ell_8] \rangle \]
we need to prove that
\[ \langle [\ell_1, \ell_2 \land \ell_4 \land \ell_6 \land \ell_8], [\ell_1 \land \ell_3 \land \ell_5 \land \ell_7, \ell_8]\rangle \sqsubseteq \langle [\ell'_1, \ell'_2 \land \ell'_4], [\ell'_1 \land \ell'_5, \ell'_6]\rangle \]
. But we know that \( \ell'_1 \ll \ell_1 \). Also that \( \ell_2 \ll \ell'_2 \land \ell'_4 \) and therefore \( \ell_2 \ll \ell'_2 \). The same for \( \ell_6 \ll \ell'_6 \) and therefore \( \ell_2 \land \ell_4 \land \ell_6 \land \ell_8 \ll \ell'_2 \land \ell'_6 \), i.e. \([\ell_1, \ell_2 \land \ell_4 \land \ell_6 \land \ell_8]\) \( \sqsubseteq [\ell'_1, \ell'_2 \land \ell'_4] \). The argument is applied for the second components and the result holds.
\[ \square \]

**Proposition 5.43.** Suppose \( \varepsilon_1 \vdash F_1_(\vec{\gamma}) \ll F_2_(\vec{\gamma}) \) and \( \varepsilon_2 \vdash F_2_(\vec{\gamma}) \subseteq F_2_(\vec{\gamma}) \)
Then \( \varepsilon_1 \sim \varepsilon_2 \vdash F_1_(\vec{\gamma}) \subseteq F_2_(\vec{\gamma}) \sigma_2 \)

**Proof.** By definition of initial evidence noticing that \( \varepsilon_1 \sim \varepsilon_2 \) can be more precise than the initial evidence of judgment

Suppose \( \varepsilon_1 = ([\ell_1, \ell_2], [\ell_3, \ell_4]) \) and \( \varepsilon_2 = ([\ell_5, \ell_6], [\ell_7, \ell_8]) \), then \( \varepsilon_1 \sim \varepsilon_2 = ([\ell_1 \land \ell_5, \ell_2 \land \ell_6], [\ell_3 \land \ell_6, \ell_4 \land \ell_8]) \).

If \( \text{bounds}(F_1(\vec{\gamma})) = [\ell'_{11} \land \ell'_{12}], \text{bounds}(F_1(\vec{\gamma})) = [\ell'_{121} \land \ell'_{122}], \text{bounds}(F_2(\vec{\gamma})) = [\ell'_{211} \land \ell'_{212}] \) and \( \text{bounds}(F_1(\vec{\gamma})) = [\ell'_{221} \land \ell'_{222}] \).

We know that \( \mathcal{J}[F_1(\vec{\gamma}) \ll F_2(\vec{\gamma})] = ([\ell'_{11}, \ell'_{12} \land \ell'_{122}], [\ell'_{11} \land \ell'_{121}, \ell'_{122}] \).

Therefore \( \ell'_{11} \ll \ell_1 \), \( \ell_2 \ll \ell'_{12} \land \ell'_{122}, \ell'_{11} \land \ell'_{121} \ll \ell_3 \) and \( \ell_4 \ll \ell'_{222} \). Using the same argument, \( \mathcal{J}[F_2(\vec{\gamma}) \subseteq F_2(\vec{\gamma})] = ([\ell'_{211}, \ell'_{212} \land \ell'_{222}], [\ell'_{211} \land \ell'_{221}, \ell'_{222}] \).

Therefore \( \ell'_{211} \ll \ell_5 \), \( \ell_6 \ll \ell'_{212} \land \ell'_{222}, \ell'_{211} \land \ell'_{221} \ll \ell_7 \) and \( \ell_8 \ll \ell'_{222} \).

But the \( \mathcal{J}[F_1(\vec{\gamma}) \ll F_2(\vec{\gamma})] = ([\ell'_{11}, \ell'_{12} \land \ell'_{12}], [\ell'_{11} \land \ell'_{12}, \ell'_{12}], \]

\( \text{bounds}(F_1(\vec{\gamma})) = \text{bounds}(F_1(\vec{\gamma})) \sim \text{bounds}(F_2(\vec{\gamma})) = [\ell'_{11} \land \ell'_{121}, \ell'_{122}, \ell'_{121} \land \ell'_{122}] \).

We need to prove that \( [\ell_1 \land \ell_5, \ell_2 \land \ell_6] \ll [\ell'_{11} \land \ell'_{121}, \ell'_{122}, \ell'_{121} \land \ell'_{122}] \), i.e. \( \ell'_{11} \land \ell'_{121} \ll \ell_1 \land \ell_5 \) and \( \ell_2 \land \ell_6 \ll \ell'_{12} \land \ell'_{212} \). But \( \ell_{111} \ll \ell_1 \) and \( \ell'_{211} \ll \ell_5 \), therefore \( \ell'_{111} \land \ell'_{211} \ll \ell_1 \land \ell_5 \). Similarly, as \( \ell_2 \ll \ell'_{121} \land \ell'_{212} \) and \( \ell_6 \ll \ell'_{121} \land \ell'_{212} \), then \( \ell_2 \land \ell_6 \ll \ell'_{121} \land \ell'_{212} \). Therefore \( [\ell_1 \land \ell_5, \ell_2 \land \ell_6] \ll [\ell'_{11} \land \ell'_{121}, \ell'_{122}] \).

Using analogous argument, we also know that \( [\ell_3 \land \ell_6, \ell_4 \land \ell_8] \ll [\ell'_{12}, \ell'_{211}, \ell'_{212}] \).

Therefore \( \varepsilon_1 \sim \varepsilon_2 \ll \mathcal{J}[F_2(\vec{\gamma}) \subseteq F_2(\vec{\gamma})] \), and the result holds.
\[ \square \]

**Lemma 5.44.** Let \( S_1, S_2 \in \text{Type} \). Then

1. If \( (S_1 \uplus S_2) \) is defined then \( S_1 \sqsubset (S_1 \uplus S_2) \).
2. If \( (S_1 \nintersection S_2) \) is defined then \( (S_1 \nintersection S_2) \ll S_1 \).

**Proof.** We start by proving (1) assuming that \( (S_1 \uplus S_2) \) is defined. We proceed by case analysis on \( S_1 \).

**Case (Bool) \( \ell \).** If \( S_1 = \text{Bool}_\ell \) then as \( (S_1 \uplus S_2) \) is defined then \( S_2 \) must have the form \( \text{Bool}_\ell \) for some \( \ell_2 \). Therefore \( (S_1 \uplus S_2) = \text{Bool}_\ell(\ell_1 \lor \ell_2) \). But by definition of \( \ll \), \( \ell' \ll (\ell' \land \ell_2) \) and therefore we use \( \ll (\text{Bool}) \) to conclude that \( \text{Bool}_\ell \ll (\text{Bool}(\ell_1 \lor \ell_2)) \), i.e. \( S_1 \ll (S_1 \uplus S_2) \).

**Case \( S \rightarrow \ell \).** If \( S_1 = S_1 \rightarrow \ell_1 \) then as \( (S_1 \uplus S_2) \) is defined then \( S_2 \) must have the form \( S_2 \rightarrow \ell_2 \) for some \( \ell_2, \ell_2 \) and \( \ell_2 \).
We also know that \((S_1 \lor S_2) = (S_1 \land S_{21}) \rightarrow (\ell_1 \lor \ell_2) (S_{12} \land S_{22})\). By definition of \(\ll\), \(\ell_1 \ll (\ell_1 \lor \ell_2)\).

Also, as \((S_1 \lor S_2)\) is defined then \((S_1 \land S_{21})\) is defined. Using the induction hypothesis of (2) on \(S_{11}\), \((S_{11} \land S_{21})\ll S_{11}\). Also, using the induction hypothesis of (1) on \(S_{12}\) we also know that \(S_{12}\ll (S_{12} \land S_{22})\). Then by \((\ll\ldots)\) we can conclude that \(S_{11} \rightarrow_{\ell_1} S_{12} \ll (S_{11} \land S_{21}) \rightarrow_{(\ell_1 \lor \ell_2)} (S_{12} \land S_{22})\), i.e. \(S_1 \ll (S_1 \lor S_2)\).

The proof of (2) is similar to (1) but using the argument that \((\ell_1 \land \ell_2) \ll \ell_1\).

\(\)\(^\square\)

**Lemma 5.45.** Let \(S \in \text{Type}\) and \(\ell \in \text{Label}\). Then \(S \ll S \lor \ell\).

**Proof.** Straightforward case analysis on type \(S\) using the fact that \(\ell \ll (\ell' \lor \ell)\) for any \(\ell'\).

\(\)\(^\square\)

**Lemma 5.46.** Let \(S_1, S_2 \in \text{Type}\) such that \(S_1 \ll S_2\), and let \(\ell_1, \ell_2 \in \text{Label}\) such that \(\ell_1 \ll \ell_2\). Then \(S_1 \lor \ell_1 \ll S_2 \lor \ell_2\).

**Proof.** Straightforward case analysis on type \(S\) using the definition of label stamping on types.

\(\)\(^\square\)

### 6 GSL\(^\varepsilon\): DYNAMIC PROPERTIES

Notice that for convenience, the proofs and properties are defined over intrinsic terms [Garcia et al. 2016] instead of terms of the internal language. They are actually the same as terms of the internal language, but keeping all static annotations explicitly. First we introduce the static semantics of intrinsic terms in Sec. 6.1. Their dynamic semantics in Sec. 6.2. The relation between intrinsic and evidence-augmented terms in Sec. 6.3. Then the proof of type safety is presented Sec. 6.4. The proof of dynamic gradual guarantee for GSL\(^\varepsilon\) without the specific check in rule \((r7)\) in section 6.5, and the proof of noninterference in Sec. 6.6.

#### 6.1 Intrinsic Terms: Static Semantics

Following Garcia et al. [2016], we develop *intrinsically typed* terms [Church 1940]: a term notation for gradual type derivations. These terms serve as our internal language for dynamic semantics: they play the same role that cast calculi play in typical presentations of gradual typing [Siek and Taha 2006]. Intrinsically-typed terms \(t^U\) comprise a family \(\Gamma[U]\) of type-indexed sets, such that ill-typed terms do not exist. They are built up from disjoint families \(\chi^U \in V[U]\) and \(o^U \in L[U]\) of intrinsically typed variables and locations respectively. Unless required, we omit the type exponent on intrinsic terms, writing \(t \in \Gamma[U]\).

To each typing rule corresponds an intrinsic term formation rule that captures all the information needed to ensure that an intrinsic term is isomorphic to a typing derivation. Because intrinsic variables and locations reflect their typings, intrinsic terms do not need explicit type environments \(\Gamma\) or store environments \(\Sigma\); however, the typing judgment depends on a security effect \(g_c\), which intrinsic terms must account for.

Additionally, because intrinsic terms represent typing derivations of programs as they reduce, they must account for the possibility that runtime values have more precise types than those used in the original typing derivation. For instance, the term in function position of an application can be a subtype of the function type used to type-check the program originally. The formation rule of the application intrinsic term must permit this extra subtyping leeway, justified by evidence. The same holds for the security information. Therefore, an intrinsic term has the general form \(\phi \triangleright t\)

, where the context information \(\phi \triangleq \langle e g_c, g_c \rangle\) contains the static program counter label \(g_c\) used
\[ \varepsilon \in \text{Evidence}, \quad e \in \text{EvTerm}, \quad ev \in \text{EvValue}, \quad v \in \text{Value}, \]
\[ u \in \text{SimpleValue}, \quad g \in \text{EvFrame}, \quad f \in \text{TmFrame} \]
\[ u ::= x^U | b_g \mid (\lambda^g x^U.f)_g \mid o_g^U \mid \text{unit}_g \]
\[ v ::= u \mid e \cdot u : U \]
\[ \varepsilon ::= (E_1, E_2) \mid (t_1, t_2) \]
\[ \mu ::= \bullet \mid \mu, o^U \mapsto v \]
\[ p ::= x^U \mid o^U \]
\[ q ::= p \mid \varepsilon \cdot p : U \]
\[ h ::= \square @ \theta \cdot e \mid ev @ \theta \cdot e \mid \square @ o^U \cdot e \mid ev @ o^U \cdot e \mid \square :: U \mid \text{if} \theta \cdot \square \text{then} e \text{else} e \mid \overline{!U} \cdot \square :: \varepsilon \cdot e \mid ev :: \varepsilon \cdot e \mid ref^U_{\theta} \cdot \square \mid prot^U_{\theta} \phi' \cdot (et) \]

Fig. 23. GSLRef: Syntax of the Intrinsic Term Language

6.2 Intrinsic Terms: Dynamic Semantics

Next we present the full definition of the intrinsic reduction rules in Figure 25, and the full definition of notions of intrinsic reduction in Figure 26.

Because the security context information of a term is maintained at each step, we also adopt the lightweight notation \( \overline{I}_1 | \mu_1 \mapsto \overline{I}_2 | \mu_2 \), to denote the reduction of the intrinsic term \( \phi \mapsto \overline{I}_1 \in \mathbb{T}[U] \) in store \( \mu_1 \) to the intrinsic term \( \phi \mapsto \overline{I}_2 \in \mathbb{T}[U] \) in store \( \mu_2 \). We note \( C[U] \) the combination of a term \( \overline{I} \in \mathbb{T}[U] \) (without context) and a store \( \mu \). Function applications reduce to to an error if consistent transitivity fails to justify \( U_2 <: U_{11} \). Conditionals similarly reduce to a new prot term, which is constructed using the static and dynamic information of the conditional term. Assignments may reduce to an ascribed unit value. Similarly to references, the stored value is ascribed the statically determined type \( U \). Therefore consistent transitivity may fail to justify that the actual type of the

1We use color to make distinctions when is needed: green is for effects and static information; orange is for the runtime information of the security effect.

2Evidence inversion functions (idom, icod, iref, ilbl and ilat) manifest the evidence for the inversion principles on consistent subtyping judgments; e.g. starting from the evidence that \( U_1 \leq U_2 \), ilbl produces the evidence of the judgment \( \text{label}(U_1) \leq \text{label}(U_2) \).
stored value is a subtype of $U$. As the value is stamped with actual labels, the term may also reduce to an error if consistent transitivity cannot support the judgment $\phi \cdot g_c \land \ell \leq U$.

### 6.3 Relating Intrinsic and Evidence-augmented Terms

In this section we present the translation rules from GSLRef terms to intrinsic terms in Figure 27. Also this section presents the erasure function in in Figure 28—highlighting the syntactics differences between terms in gray—along properties that relates evidence-augmented terms and intrinsic terms.

In particular we identify four important properties. First, that given a source language the translation to intrinsic term is equal to the translation of the source term to an evidence-augmented term:

**Proposition 6.1.** If $\Gamma; \Sigma; g_c \vdash t \leadsto i : U$ and $\Gamma; \Sigma; g_c \vdash t' : U$, then $|i| = t'$.

**Proof.** By induction on the type derivation of $t$. 

Second, given a reducible intrinsic term $i$, if it reduces to an error, then it erasure also reduces to an error; or, if reduces to an intrinsic term $i'$, then the erasure of $i'$ also reduces to the erasure of $i'$:

**Proposition 6.2.** Consider $\phi = e g_c, \phi \vdash i : T[U]$, and $\Gamma; \Sigma; g_c \vdash t : U$, such that $\Sigma \models \mu_2$.

Then if $i = t$ and $\mu_1 = \mu'_1$ then either
\[\phi \vdash C[U] \times (C[U] \cup \{\text{error}\})\]

\[
\begin{array}{c}
\text{(R->)} t^U | \mu \xrightarrow{\phi} r & r \in C[U] \cup \{\text{error}\} \\
\text{(Rf)} \quad \iota_1 | \mu \xrightarrow{\phi} \iota_2 | \mu' \\
\text{(Rprot)} \quad \iota_1 | \mu \xrightarrow{\phi'} \iota_2 | \mu' \\
\text{(Rh)} \quad et \rightarrow_c et' \\
\text{(Rerr)} \quad \text{error} \\
\text{(Rproth)} \quad \text{error} \\
\end{array}
\]

- \[\iota_1 | \mu_1 \xrightarrow{\phi} \iota'_1 | \mu_2 \Rightarrow |\iota'_1| |\mu_2| \xrightarrow{\text{eq}} |\iota'| |\mu_2|, \text{ or}\]
- \[\iota_1 | \mu_1 \xrightarrow{\phi} \text{error} \Rightarrow |\iota_1| |\mu_2| \text{error}\]

**Proof.** By induction on the type derivation of \(\iota_1\).

**Case (I-).** Then \(\iota = \varepsilon_1 \iota' :: U\) and by (E-), \(t = \varepsilon_1 t'\) for some \(t'\) such that \(\iota' = t'\). Suppose that \(\varepsilon_1 :: U' \leq U\). By inspection on the type derivations, \(\phi \triangleright \iota' \in T[U']\) and \(\vdash ; |\varepsilon_1 | \varepsilon_1 t' : U'\).

Let us suppose that \(\iota' | \mu_1 \xrightarrow{\phi} \iota'' | \mu_2\), then by induction hypothesis \(t' | \mu_2 \xrightarrow{\text{eq}} t'' | \mu_2\) and \(\iota'' = t''\) and \(\mu'_1 = \mu'_2\). Then \(\varepsilon_1 \iota' :: U | \mu_1 \xrightarrow{\phi} \varepsilon_1 \iota'' :: U | \mu'_1\) and \(\varepsilon_1 t' | \mu_2 \xrightarrow{\text{eq}} \varepsilon_1 t'' | \mu'_2\). But as \(\mu'_1 = \mu'_2\), and by (E:) \(\varepsilon_1 \iota'' :: U = \varepsilon_1 t''\), the result holds.

Let us suppose now that \(\iota'' = \varepsilon_2 u :: U'\). Then as \(\iota' = t', t' = \varepsilon_2 u',\) for some \(u'\) such that \(u = u'\). If \(\varepsilon_2 \circ \varepsilon_1\) is not defined the result holds immediately. Suppose \(\varepsilon_2 \circ \varepsilon_1 = \varepsilon'\), then \(\varepsilon_1 (\varepsilon_2 u :: U') :: U | \mu_1 \xrightarrow{\phi} \varepsilon' u :: U | \mu_1\) and \(\varepsilon_1 (\varepsilon_2 u') | \mu_2 \xrightarrow{\text{eq}} \varepsilon' u' | \mu_2\). But as \(\mu_1 = \mu_2\), and by (E:) \(\varepsilon' u :: U = \varepsilon' u'\), the result holds.

If \(\iota' = u\), then as \(\iota' = t', t' = \varepsilon_2 u',\) for some \(u'\) such that \(u = u'\), and the result holds immediately.

The other cases proceed analogous.

Fourth, if an intrinsic term type checks, then its erasure also type checks to the same type.

**Proposition 6.3.** Consider \(\phi \triangleright \iota \in T[U]\) then, for \(\Gamma \models \iota\) and \(\Sigma \models \iota, \Gamma; \Sigma; |\phi| \vdash \iota : U\).

**Proof.** By induction on the type derivation of \(\iota\).
Notions of Reduction

\[ \phi : \mathbb{C}[U] \times (\mathbb{C}[U] \cup \{ \text{error} \}) \]

\[ \varepsilon_1(b_1)_{g_1} \bowtie \varepsilon_2(b_2)_{g_2} \mid \mu \xrightarrow{\phi} (\varepsilon_1 \setminus \varepsilon_2)(b_1 \bowtie b_2)_{(g_1 \setminus g_2)} : \text{Bool} \mid \mu \]

\[ \text{prot}^U_{\varepsilon_2} \phi'(\varepsilon_1 u) \mid \mu \xrightarrow{\phi} (\varepsilon_1 \setminus \varepsilon_2)(u \setminus g') : U \setminus g \mid \mu \]

\[ \varepsilon_1(\lambda g'_{x_{\text{U}11}, t_{\text{s}}})_{g_{\text{s}}} \xrightarrow{\text{prot}^U_{\varepsilon_2} \phi'_{\varepsilon_2} u} \varepsilon_2' u \mid \mu \xrightarrow{\phi} \left\{ \begin{array}{ll}
\text{error} & \text{if } \varepsilon \text{ or } \varepsilon' \text{ are not defined}
\end{array} \right. \]

where \( \varepsilon' = \varepsilon_2 \circ \epsilon \setminus \text{idom}(\varepsilon_1) \), \( \varepsilon' = (\phi \cdot \epsilon \setminus \text{ilbl}(\varepsilon_1)) \circ \epsilon \setminus \text{ilat}(\varepsilon_1) \)

and \( \phi' = \langle \varepsilon', \phi \cdot g_c \setminus g_2, g_2' \rangle \)

if \( \varepsilon_1 \text{true}_{g_1} \), then \( \text{error} \)

else \( \text{error} \)

\[ \text{ref}^U_{\varepsilon_2} u \mid \mu \xrightarrow{\phi} \left\{ \begin{array}{ll}
\text{error} & \text{if } \phi \cdot \epsilon_\ell \text{ is not defined}
\end{array} \right. \]

where \( \epsilon' = \epsilon \setminus (\phi \cdot \epsilon \setminus \epsilon_\ell) \)

\[ !\text{Ref}_{\varepsilon_2}^U u' \mid \mu \xrightarrow{\phi} \text{prot}^U_{\varepsilon_2} \phi'_{\varepsilon_2} (\text{iref}(\epsilon) u) \]

where \( \mu(o_U) = u \) and \( \phi' = \langle \phi \cdot \epsilon \setminus \text{ilbl}(\varepsilon), \phi \cdot g_c \setminus g, \phi \cdot g_c \setminus g \rangle \)

\[ \varepsilon_1 o_U^g \xrightarrow{\phi} \varepsilon_1 \varepsilon_2 u \mid \mu \xrightarrow{\phi} \left\{ \begin{array}{ll}
\text{error} & \text{if } \epsilon' \text{ is not defined, or}
\phi \cdot \epsilon \setminus \text{ilbl}(\varepsilon_1) \leq \text{ilbl}(\varepsilon) \text{ does not hold}
\end{array} \right. \]

where \( \epsilon' = (\epsilon_2 \circ \epsilon \setminus \text{iref}(\epsilon_1)) \setminus (\phi \cdot \epsilon \setminus \text{ilbl}(\varepsilon_1)) \circ \epsilon \setminus \text{ilat}(\epsilon_1) \)

and \( \mu(o_U) = \epsilon u' : U \)

\[ \rightarrow_c : \text{EvTerm} \times (\text{EvTerm} \cup \{ \text{error} \}) \]

\[ \varepsilon_1(\varepsilon_2 v : U) \rightarrow_c \left\{ \begin{array}{ll}
\epsilon_2 \circ \epsilon : \varepsilon_1 v & \text{if not defined}
\end{array} \right. \]

\[ \langle \ell_1, \ell_2, [\ell_3, \ell_4] \rangle \leq \langle \ell'_1, \ell'_2, [\ell'_3, \ell'_4] \rangle \iff \ell_3 \leq \ell'_4 \]

Fig. 26. GSLRef: Intrinsic Notions of Reduction

Finally, if an evidence-augmented term type checks, then there must exists some intrinsic term that have the same type and that it erasure is the original evidence-augmented term.

**Proposition 6.4.** Consider \( \Gamma; \Sigma; \epsilon g_c \vdash t : U \). Then \( \exists ! \ell \) such that \( |\ell| = t \) and \( |\phi| = \epsilon g_c \) and \( \phi \cdot \ell \in \Gamma[U] \)

**Proof.** By induction on the type derivation of \( t \).
Figure 27. GSL\textsubscript{Ref} translation to GSL\textsubscript{Ref} intrinsic terms

**Case (ε’t’).** Then \( t = \epsilon' t' \), for some \( \epsilon', t' \). But we know that \( \Gamma; \Sigma; \epsilon g_c + \epsilon' t' : U \) and suppose \( \epsilon' + U' \subseteq U \) and \( \epsilon + g_c \leq g_c \). Then by choosing \( \phi = \langle \epsilon, g_c \rangle g_c' \) and induction hypothesis on \( t', \exists ! t' \) such that \( \phi \triangleright t' \in T[U'] \).

The other cases proceed analogous.

**Lemma 6.5.** Consider \( \phi \triangleright \bar{t}_1 \in T[U] \). If \( |\bar{t}_1| \subseteq |\bar{t}_2| \) then \(|\bar{t}_1| \subseteq |\bar{t}_2| \).

**Proof.** By induction on the type derivation of \( \bar{t}_1 \) and the definition of \(||\).

**Lemma 6.6.** Consider \( \phi \triangleright \bar{t}_1 \in T[U] \). If \(|\bar{t}_1| \subsetneq |\bar{t}_2| \) then \( \exists ! \bar{t}_2 \), such that \( |\bar{t}_1| \subseteq |\bar{t}_2| \) and that \(|\bar{t}_2| = t_2 \).
Also, the store must preserve types between intrinsic locations and underlying values.

\[ \epsilon \ \text{depends on} \ \Gamma \]

In this section we present the proof of type safety for GSL.  

\section{Type Safety}

In this section we present the proof of type safety for \( \text{GSL}^{r}_{\text{Ref}} \).

We define what it means for a store to be well typed with respect to a term. Informally, all free locations of a term and of the contents of the store must be defined in the domain of that store. Also, the store must preserve types between intrinsic locations and underlying values.

\begin{definition}[\( \mu \) is well typed] \( \mu \) is said to be well typed with respect to an intrinsic term \( t^{U} \), written \( t^{U} \vdash \mu \), if:
\begin{enumerate}
\item \( \text{freeLocs}(t^{U}) \subseteq \text{dom}(\mu) \), and
\item \( \forall \nu \in \text{cod}(\mu), \nu \vdash \mu \) and
\item \( \forall o^{U} \in \text{dom}(\mu), \forall \phi, \phi \vdash \mu(o^{U}) \in \mathbb{T}[U] \).
\end{enumerate}
\end{definition}

\begin{lemma}
Suppose \( \phi \vdash t^{U} \in \mathbb{T}[U] \), then \( \forall g^{r}, \forall \epsilon, \phi^{r} \leq \phi \cdot g^{r} \) and \( \epsilon^{r} \vdash g^{r} \leq \phi \cdot g^{r} \), \( \phi^{r} = \langle \epsilon, g^{r}, \phi, g^{r} \rangle \) then \( \phi^{r} \vdash t^{U} \in \mathbb{T}[U] \).
\end{lemma}

\begin{proof}
By induction on the derivation of \( \phi \vdash t^{U} \in \mathbb{T}[U] \). Noticing that no typing derivation depends on \( \epsilon^{r}, g^{r} \), save for the judgement \( \epsilon^{r} \vdash g^{r} \leq g^{r} \) which is premise of this lemma.
\end{proof}

\begin{lemma}
Suppose \( \phi \vdash \nu \in \mathbb{T}[U] \), then \( \forall \phi^{r}, \phi \vdash \nu \in \mathbb{T}[U] \).
\end{lemma}
Type-Driven Gradual Security with References: Complete Definitions and Proofs

Proof. By induction on the derivation of \( \phi' \triangleright v \) observing that for values, there is no premise that depends on the security effect.

Lemma 6.10 (Canonical forms). Consider a value \( v \in T[U] \). Then either \( v = u \), or \( v = e u : U \) with \( u \in T[U'] \) and \( e : U' \leq U \). Furthermore:

1. If \( U = Bool_g \) then either \( v = b_g \) or \( v = e b_g : Bool_g \) with \( b_g \in T[Bool_g] \) and \( e : Bool_g \leq Bool_g \).
2. If \( U = U_1 \xrightarrow{g_e} ^g_y U_2 \) then either \( v = (\lambda^g x . t U_1) \) with \( t U_2 \in T[U_2] \) or \( v = e (\lambda^g x . t U_1) y : U_1 \xrightarrow{g_e} ^g_y U_2 \).
3. If \( U = Ref_g U_1 \) then either \( v = o^U_{g_1} \) or \( v = e o^U_{g_1} : Ref_g U_1 \) with \( o^U_{g_1} \in Ref_g U_1 \) and \( e : Ref_g U_1 \leq Ref_g U_1 \).

Proof. By direct inspection of the formation rules of gradual intrinsic terms (Figure 24).

Lemma 6.11 (Substitution). If \( \phi \triangleright t^U \in T[U] \) and \( \phi \triangleright v \in T[U_1] \), then \( \phi \triangleright [v/x]^U t^U \in T[U] \).

Proof. By induction on the derivation of \( \phi \triangleright t^U \).

Proposition 6.12 (\( \rightarrow \) is well defined). If \( t^U \mid \mu \xrightarrow{r} r \) and \( t^U \vdash \mu' \), then \( r \in Config_U \cup \{ error \} \) and if \( r = t^U \mid \mu' \in Config_U \) then also \( t^U \vdash \mu' \) and \( dom(\mu) \subseteq dom(\mu') \).

Proof. By induction on the structure of a derivation of \( t^U \mid \mu \xrightarrow{r} r \), considering the last rule used in the derivation.

Case (\( \oplus \)). Then \( t^U = b_1 \ell_1 \oplus g b_2 \ell_2 \). By construction we can suppose that \( g = g_1 \top g_2 \), then

\[
\frac{\phi \triangleright b_1 \ell_1 \in Bool_g \quad \ell_1 \vdash Bool_g \leq Bool_g}{\phi \triangleright b_1 \ell_1 \oplus g b_2 \ell_2 \in T[Bool_g]}
\] (I\( \oplus \))

Therefore

\[
\phi \triangleright \ell_1 \top g \ell_2 (b_1 \top g) (b_2) :: Bool_g \mid \mu
\]

Then

\[
\frac{\phi \triangleright \ell_1 \top g \ell_2 \in T[Bool_g]}{\phi \triangleright (\ell_1 \top g \ell_2) (b_1 \top g) (b_2) :: Bool_g \mid \mu}
\] (I\( \oplus \))

and the result holds.

Case (Iprot). Then \( t^U = \phi \triangleright prot^U_{\ell g} \phi' (e u) \) and

\[
\frac{\phi \triangleright u \in T[U'] \quad e : U' \leq U \quad g' \leq g}{\phi \triangleright \phi' (e u) \in T[U \top g']}
\] (Iprot)

Therefore

\[
\phi \triangleright prot^U_{\ell g} \phi' (e u) \mid \mu \xrightarrow{\ell \top g} (\ell \top e \ell) (u \top g') :: U \top g \mid \mu
\]
But by Lemma 6.9, $\phi \triangleright u \in T[U']$. Therefore by definition of join $\phi \triangleright (u \triangleright g') \in T[U' \triangleright g']$. Then using Lemma 5.43

$$
\begin{align*}
\phi \triangleright (u \triangleright g') & \in T[U' \triangleright g'] \\
& (\epsilon \triangleright \epsilon \ell U) + U' \triangleright g' \leq U \triangleright g \\
\phi \triangleright (\epsilon \triangleright \epsilon \ell) (u \triangleright g') & :: U \triangleright g \in T[U \triangleright \ell]
\end{align*}
$$

and the result holds.

**Case (lapp).** Then $t^U = \varepsilon_1 (\lambda \theta_{\ell, \epsilon}^{U_{11}}, t^{U_{12}})_{g_1} @ t_{\ell, \epsilon}^{U_{11} \rightarrow U_{12}} \varepsilon_2 u$ and $U = U_2 \triangleright \epsilon \triangleright g$. Then

$$
\begin{align*}
\phi \triangleright t^{U_{12}} & \in T[U_{12}] \\
& \phi, \epsilon + \phi, \epsilon \leq \phi, \epsilon
\end{align*}
$$

$$
\begin{align*}
\phi \triangleright (\lambda \theta_{\ell, \epsilon}^{U_{11}}, t^{U_{12}})_{g_1} & \in T[U_{11} \rightarrow t_{g_1} U_{12}] \\
\phi \triangleright u & \in T[U'_2] \\
\varepsilon_2 + U'_2 & \leq U_1 \\
\epsilon \ell + \varepsilon_2 \triangleright g \leq \theta_{\ell}^{g'} & \phi, \epsilon + \phi, \epsilon \leq \phi, \epsilon
\end{align*}
$$

(lapp)

If $\epsilon' = (\epsilon_2 \circ \varepsilon_{11} : idom(\varepsilon_1))$, or $\epsilon' = (\phi, \epsilon \triangleright ilbl(\varepsilon_1)) \circ \varepsilon_\ell \circ \varepsilon_{11} \triangleright ilat(\varepsilon_1)$ are not defined, then $t^U \mid \mu \rightarrow \text{error}$, and then the result hold immediately. Suppose that consistent transitivity do holds, then if $\phi' = (\phi', \epsilon \triangleright g, \varepsilon_{11}, \varepsilon_{11}^{\prime})$

$$
\varepsilon_1 (\lambda \theta_{\ell, \epsilon}^{U_{11}}, t^{U_{12}})_{g_1} @ t_{\ell, \epsilon}^{U_{11} \rightarrow U_{12}} \varepsilon_2 u \mid \mu \rightarrow \text{prot}^{t_{ilbl(\varepsilon_1)} g_1 t_{ilat(\varepsilon_1)}} (icod(\varepsilon_1)([(\epsilon' u :: U_{11}) / x^{U_{11}}]) t^{U_{12}}) \mid \mu
$$

As $\varepsilon_2 + U'_2 \leq U_1$ and by inversion lemma $idom(\varepsilon_1) + U_1 \leq U_{11}$, then $\epsilon' \triangleright U'_2 \leq U_{11}$. Therefore $\phi' \triangleright \epsilon' u :: U_{11} \in T[U_{11}]$, and by Lemma 6.11,

$$
\phi' \triangleright [(\epsilon' u :: U_{11}) / x^{U_{11}}] t^{U_{12}} \in T[U_{12}].
$$

We know that $\epsilon \ell \triangleright \varepsilon_2 \triangleright g \leq \theta_{\ell}^{g'}$. By inversion on the label of types, $ilbl(\varepsilon_1) \triangleright g_1 \leq g$. Also by monotonicity of the join, $\phi, \epsilon \triangleright ilbl(\varepsilon_1) \triangleright \phi, \epsilon \leq g_1 \leq g$. Then, by inversion on the latent effect of function types, $ilat(\varepsilon_1) \triangleright g_1 \leq g_1^{\prime \prime}$. Therefore combining evidences, as $\phi', \epsilon = (\phi, \epsilon \triangleright ilbl(\varepsilon_1)) \circ \varepsilon_{11} \triangleright ilat(\varepsilon_1)$, we may justify the runtime judgment $\phi', \epsilon \triangleright \phi, \epsilon \leq g_1 \leq g_1^{\prime \prime}$.

Let us call $t^{U_{12}} = [(\epsilon' u :: U_{11}) / x^{U_{11}}] t^{U_{12}}$. By Lemma 6.8, $\phi' \triangleright t^{U_{12}} \in T[U_{12}]$. Then

$$
\begin{align*}
\phi, \epsilon + \phi, \epsilon \leq \phi, \epsilon \\
\phi' \triangleright t^{U_{12}} & \in T[U_{12}] \\
\phi' \triangleright \text{prot}^{t_{ilbl(\varepsilon_1)} g_1 t_{ilat(\varepsilon_1)}} (icod(\varepsilon_1) t^{U_{12}}) & \in T[U_2 \triangleright \ell]
\end{align*}
$$

and the result holds.
Case (lif-true). Then \( t^U = \text{if}^{\mathsf{if}} \varepsilon_1 \cdot b \eta_1 \text{ then } \varepsilon_2 \cdot t^U_2 \text{ else } \varepsilon_3 \cdot t^U_3 \), \( U = (U_2 \triangleright U_3) \triangleright g \) and

\[
\begin{align*}
\phi \triangleright b \eta_1 & \in T[\mathsf{Bool}_{\eta_1}] & \varepsilon_1 & \vdash \mathsf{Bool}_{\eta_1} \leq \mathsf{Bool}_g \\
\phi' = (\phi \cdot \varepsilon \triangleright \mathit{ilbl}(\varepsilon_1)(\phi \cdot g_c \triangleright g_1), \phi \cdot g_c \triangleright g) & \phi \cdot \varepsilon + \phi \cdot g_c \leq \phi \cdot g_c \\
\phi' \triangleright t^U_2 & \in T[U_2] & \varepsilon_2 & \vdash U_2 \leq (U_2 \triangleright U_3) \\
\phi' \triangleright t^U_3 & \in T[U_3] & \varepsilon_3 & \vdash U_3 \leq (U_2 \triangleright U_3)
\end{align*}
\]

Therefore

\[
\begin{align*}
\text{if}^{\mathsf{if}} \varepsilon_1 \cdot b \eta_1 \text{ then } \varepsilon_2 \cdot t^U_2 \text{ else } \varepsilon_3 \cdot t^U_3 & \in T[(U_2 \triangleright U_3) \triangleright g]
\end{align*}
\]

But

\[
\begin{align*}
\phi \cdot \varepsilon + \phi \cdot g_c & \leq \phi \cdot g_c \\
\phi' \triangleright t^U_2 & \in T[U_2] & \varepsilon_2 & \vdash U_2 \leq (U_2 \triangleright U_3) \\
\mathit{ilbl}(\varepsilon_1) & + g_1 \triangleright g & \phi & \vdash \mathsf{prot}_{\mathit{ilbl}(\varepsilon_1) \cdot g_1}(\phi'(\varepsilon_2 \cdot t^U_2)) \in T[(U_2 \triangleright U_3) \triangleright g]
\end{align*}
\]

and the result holds.

Case (lif-false). Analogous to case (if-true).

Case (liref). Then \( t^U = \mathsf{ref}^{\mathsf{ref}}_{\varepsilon_\ell} \cdot \varepsilon u \) and

\[
\begin{align*}
\phi \cdot \varepsilon + \phi \cdot g_c & \leq \phi \cdot g_c & \phi \triangleright u & \in T[U''] & \varepsilon \cdot U'' & \leq U' & \varepsilon_\ell + g_c & \triangleright \mathit{label}(U')
\end{align*}
\]

\[
\phi \vdash \mathsf{ref}^{\mathsf{ref}}_{\varepsilon_\ell} \cdot \varepsilon u \in T[\mathsf{Ref}_U U']
\]

If \( \varepsilon' = \varepsilon \triangleright (\phi \cdot \varepsilon \circ \varepsilon_\ell) \) is not defined, then \( t^U'' \vdash \mu \overset{\phi}{\longrightarrow} \text{error} \), and then the result hold immediately. Suppose that consistent transitivity does hold, then

\[
\begin{align*}
\text{ref}^{\mathsf{ref}}_{\varepsilon_\ell} \cdot \varepsilon u \vdash \mu \overset{\phi}{\longrightarrow} \varepsilon' \cdot u \triangleright (\phi \cdot \varepsilon \circ \varepsilon_\ell) : U''
\end{align*}
\]

where \( o'' \notin \mathit{dom}(\mu) \).

We know that \( \varepsilon_\ell + g_c \triangleright \mathit{label}(U') \), therefore \( \phi \cdot \varepsilon \circ \varepsilon_\ell + \phi \cdot g_c \triangleright \mathit{label}(U') \). We also know that \( \varepsilon \vdash U'' \leq U' \). Therefore combining both evidences we can justify that \( \varepsilon \triangleright (\phi \cdot \varepsilon \circ \varepsilon_\ell) + U'_2 \triangleright \phi \cdot g_c : \leq U' \).

But

\[
\begin{align*}
\phi \cdot \varepsilon + \phi \cdot g_c & \leq \phi \cdot g_c \\
o''_U & \in T[\mathsf{Ref}_U U']
\end{align*}
\]

Let us call \( \mu' = \mu[o''_U \mapsto \varepsilon'(u \triangleright \phi \cdot g_c) : U''] \). It is easy to see that \( \mathit{freeLocs}(o''_U) = o''_U \) and \( \mathit{dom}(\mu') = \mathit{dom}(\mu) \cup o''_U \), then \( \mathit{freeLocs}(o''_U) \subseteq \mathit{dom}(\mu') \). Given that \( t^U'' + \mu \) then \( \mathit{freeLocs}(u) \subseteq \mathit{dom}(\mu) \), and therefore \( \forall u \in \mathit{cod}(\mu') = \mathit{cod}(\mu) \cup (\varepsilon'(u \triangleright \phi \cdot g_c) : U''), \mathit{freeLocs}(u') \subseteq \mathit{dom}(\mu') \). Finally as \( t^U'' + \mu \) and \( \mu'(o''_U) = o''_U \cdot \phi \cdot g_c : U'' \in T[U'] \) then we can conclude that \( t^U'' + \mu' \) and \( \mathit{dom}(\mu) \subseteq \mathit{dom}(\mu') \), and the result holds.

Case (leref). Then \( t^U = \mathsf{!ref}_{\varepsilon_\ell} \cdot \varepsilon_\ell o''_U, U = U' \triangleright \varepsilon \cdot g \) and

\[
\begin{align*}
\phi \triangleright o''_U & \in T[\mathsf{Ref}_{\varepsilon_\ell} U'''] \\
\varepsilon & \vdash \mathsf{ref}_{\varepsilon_\ell} U''' \leq \mathsf{Ref}_{\varepsilon_\ell} U' \\
\phi \cdot \varepsilon + \phi \cdot g_c & \leq \phi \cdot g_c \\
\phi & \vdash \mathsf{!ref}_{\varepsilon_\ell} U' \cdot o''_{\varepsilon_\ell} \in T[U' \triangleright g]
\end{align*}
\]
Then for $\phi' = ((\phi. e \lor ilbl(e)) (\phi. g) \lor g') (\phi. g) \lor g)$

$$t^{\text{Ref}_e} U = \epsilon_{\phi} U' \mid u \xrightarrow{\phi} \text{prot}^{\phi. U'} \phi'(\text{iref}(e)u) \mid \mu$$

where $\mu(o^{U''}) = v$. As the store is well typed, therefore $\phi \lor v \in T[U'']$. By Lemma 6.9, $\phi \lor v \in T[U'']$.

By inversion lemma on references, $\text{iref}(e) + g' \subset g$ and $\text{iref}(e) + U'' \subset U'$

$$\phi.e + (\phi.g) \subset (\phi.g) \quad \phi' \lor v \in T[U'']$$

and the result holds.

**Case (lassgn).** Then $t^U = \epsilon_1^{\phi} g_1 : \epsilon_2^{\phi} u$ and

$$\epsilon_1 \vdash \text{Ref}_e U' \leq \text{Ref}_g U_1 \quad \phi \lor o^{U'} \in T[\text{Ref}_e U_1']$$

$$\epsilon_2 \vdash U_2 \leq U_1 \quad \phi \lor u \in T[U_2]$$

$$\phi + \phi.g \lor g \leq \text{label}(U_1)$$

If $\epsilon' = (\epsilon_2 \triangleright o^{\leq} \text{iref}(e_1)) \triangleright ((\phi.e \lor ilbl(e_1)) \triangleright (\phi.g \lor g))$ is not defined, then $t^{U'} \mid \mu \not\rightarrow \text{error}$, and then the result hold immediately. Suppose that consistent transitivity does hold, then

$$\epsilon_1^{\phi} g_1 : \epsilon_2^{\phi} u \in T[\text{Unit}_U]$$

We know that $\epsilon_2 \vdash \phi.g \lor g \leq \text{label}(U_1)$. Then by inversion on reference evidence types and inversion in the label of types, $\text{iref}(e_1) + \text{label}(U_1) \leq \text{label}(U_1')$. But $\text{ilbl}(e_1) + g' \subset g$, using monotonicity of the join, $(\phi.e \lor ilbl(e_1)) \triangleright (\phi.g \lor g')$. Therefore

$$((\phi.e \lor ilbl(e_1)) \triangleright (\phi.g \lor g')) \triangleright (\phi.g \lor g)$$

and by Proposition 5.43 we can then justify that $\epsilon' \vdash U_2 \lor (\phi.g \lor g) \leq U_1'$.

Let us call $\mu' = \mu[o^{U''} \triangleright (\epsilon_1 \lor ((\phi.e \lor ilbl(e_1)) \triangleright (\phi.g \lor g))) :: U_1']$. As freeLocs(unit$_U) = \emptyset$ then freeLocs(unit$_U) \subseteq \mu'$.

As $t^U \not\rightarrow \mu$ then freeLocs(u) $\in \text{dom}(\mu)$, and as dom(\mu) = dom(\mu') then it is trivial to see that $\forall \nu' \in \text{cod}(\mu'), \text{freeLocs}(\nu') \subseteq \text{dom}(\mu')$, and the result holds.

\[\square\]

**Proposition 6.13 (\rightarrow IS WELL DEFINED).** If $\not\rightarrow$ is well defined, then $\not\rightarrow$ is well defined, and if $r = t^U \mid \mu \not\rightarrow r$ and $t^U \vdash \mu$, then $r \in \text{Config}_U \cup \{\text{error}\}$.

**Proof.** By induction on the structure of a derivation of $t^U \mid \mu \not\rightarrow r$.

**Case (R\rightarrow).** $t^U \mid \mu \not\rightarrow r$. By well-definedness of \rightarrow (Prop 6.12), $r \in \text{Config}_U \cup \{\text{error}\}$ and if $r = t'^U \mid \mu' \in \text{Config}_U$ then also $t'^U \vdash \mu'$ and dom(\mu) $\subseteq$ dom(\mu').
Case (Rprot). \( t^U = \text{prot}^{\phi,U'}_{\epsilon^g} \phi'(\epsilon^1_U) \) and

\[
\begin{align*}
\phi, e &\vdash \phi, g_e \triangleleft \phi, g_e' \quad \epsilon_e' \triangleright g_r \quad \epsilon_e' \triangleright g' \triangleleft g_e' \\
\phi' \triangleright t^1_U &\in T[U'] \\
\epsilon \triangleright U'' &\leq U' \\
\epsilon \triangleright g' &\triangleleft g
\end{align*}
\]

Using induction hypothesis on the premise of \((\text{Rprot})\), then

\[
\begin{align*}
t^1_U &\in \mu' \quad \epsilon' \triangleright t^1_U \in \mu' \\
\text{prot}^{\phi,U'}_{\epsilon^g} \phi'(\epsilon^1_U) &\in \mu' \\
\phi' &\triangleright t^2_U \in T[U''] \\
\epsilon' \triangleright g' &\triangleleft g
\end{align*}
\]

where \( \phi' \triangleright t^2_U \in T[U''] \), \( t^2_U \vdash \mu' \) and \( \text{dom}(\mu) \subseteq \text{dom}(\mu') \). Therefore

\[
\begin{align*}
\phi, e &\vdash \phi, g_e' \triangleleft \phi, g_e' \quad \epsilon_e' \triangleright g_r \quad \epsilon_e' \triangleright g' \triangleleft g_e' \\
\phi' &\triangleright t^2_U \in T[U''] \\
\epsilon \triangleright U'' &\leq U' \\
\epsilon \triangleright g' &\triangleleft g
\end{align*}
\]

and the result holds.

Case (Rf). \( t^U = f[t^U_1] \), \( \phi \triangleright f[t^U_1] \in T[U] \), \( t^U_1 \in \mu' \), and consider \( F : T[U'] \to T[U] \), where \( F(\phi \triangleright t^U_1) = \phi \triangleright f[t^U_1] \). By induction hypothesis, \( \phi \triangleright t^U_1 \in T[U'] \), so \( F(\phi \triangleright t^U_1) = \phi \triangleright f[t^U_1] \in T[U] \).

By induction hypothesis we also know that \( t^U \vdash \mu' \).

If \( \text{freeLocs}(t^U_1) \subseteq \mu', \text{freeLocs}(f[t^U_1]) \subseteq \mu, \) and \( \text{dom}(\mu) \subseteq \text{dom}(\mu') \), then it is easy to see that \( \text{freeLocs}(f[t^U_1]) \subseteq \mu' \), and therefore conclude that \( f[t^U_1] \vdash \mu' \).

Case (Rh). \( t^U = h[et], \phi \triangleright h[et] \in T[U] \), and consider \( G : \text{EvLabel} \times \text{GLabel} \times \text{GLabel} \times \text{EvTerm} \to T[U], G(\phi, et) = \phi \triangleright h[et] \) and \( et \longrightarrow_c et' \). Then there exists \( U_e, U_x \) such that \( et \triangleright \epsilon_e U^U_v \) and \( \epsilon_e \triangleright U_e \triangleleft U_x \). Also, \( \epsilon_e = \epsilon_v \circ \epsilon_e \triangleleft U_v \), with \( v \in T[U_v] \) and \( \epsilon_v \triangleright U_v \leq U_e \).

We know that \( \epsilon_e = \epsilon_v \circ \epsilon_e \) is defined, and \( et \triangleright \epsilon_e U_v \longrightarrow_c \epsilon_e v = et' \). By definition of \( \circ \triangleleft \): we have

\[
\epsilon_v \triangleright U_v \triangleleft U_x, \text{ so } G(\phi, et') = \phi \triangleright h[et'] \in T[U].
\]

As \( \text{freeLocs}(et) = \text{freeLocs}(et') \) and \( \mu' = \mu \) then it is easy to conclude that \( h[et'] \vdash \mu \).

Case (Rprot()h). Similar case to (Rh) case, using \( P : \text{EvTerm} \to T[U], P(et) = \phi \triangleright \text{prot}^{\phi,U}_{\epsilon^g} \phi'(et) \).

\( \Box \)

Now we can establish type safety: programs do not get stuck, though they may terminate with cast errors. Also the store of a program is well typed.

**Proposition 6.14 (Type Safety).** If \( \phi \triangleright t^U \in T[U] \) then either \( t^U \) is a value \( v; t^U \vdash \mu \quad \phi \triangleright \text{error}; \) or if \( t^U \vdash \mu \) then \( t^U \vdash \mu \quad \phi \triangleright t^U \vdash \mu' \) for some term \( \phi \triangleright t^U \in T[U] \) and some \( \mu' \) such that \( t^U \vdash \mu' \) and \( \text{dom}(\mu) \subseteq \text{dom}(\mu') \).

**Proof.** By induction on the structure of \( \phi \triangleright t^U \).

Case (Iu,II, Ib, IX, I\lambda). \( t^U \) is a value.
Case (\(I_{prot}\)). \(t^U = \text{prot}_{e_{i'}}^{g,U} \phi'(et^{U'})\), and

\[
\begin{align*}
\phi, e &\vdash \phi, g_c \ll \phi, g_c \\
\phi' &\vdash t^{U'} \in T[U'] \\
\varepsilon &\vdash t^{U} \ll U & \varepsilon &\vdash g' \ll g
\end{align*}
\]

By induction hypothesis on \(t^{U'}\), one of the following holds:

1. \(t^{U'}\) is a simple value, then by (R\(\rightarrow\)), \(t^U | \mu \rightarrow^\phi \nu | \mu\), and by Prop 6.13, \(\phi \rightarrow \nu \in T[U]\) and the result holds.
2. \(t^{U'}\) is an ascribed value \(v\), then, \(et^{U'} \rightarrow_c et'\) for some \(et' \in \text{EvTerm} \cup \{\text{error}\}\). Hence \(t^U | \mu \rightarrow r\) for some \(r \in \text{Config}_U \cup \{\text{error}\}\) by Prop 6.13 and either (Rg), or (Rgerr).
3. \(t^{U'} | \mu \rightarrow r_1\) for some \(r_1 \in T[U_1] \cup \{\text{error}\}\). Hence \(t^U | \mu \rightarrow r\) for some \(r \in \text{Config}_U \cup \{\text{error}\}\) by Prop 6.13 and either (Rprot), or (Rprot(err)).

Case (I::). \(t^U = \varepsilon_1 t^{U_1} :: U_2\), and

\[
\phi \vdash t^{U_1} \in T[U_1] \\
\varepsilon_1 \vdash U_1 \ll U_2 \\
\phi, e \vdash \phi, g_c \ll \phi, g_c \\
\phi \vdash \varepsilon_1 t^{U_1} :: U_2 \in T[U_2]
\]

By induction hypothesis on \(t^{U_1}\), one of the following holds:

1. \(t^{U_1}\) is a value, in which case \(t^U\) is also a value.
2. \(t^{U_1} | \mu \rightarrow r_1\) for some \(r_1 \in T[U_1] \cup \{\text{error}\}\). Hence \(t^U | \mu \rightarrow r\) for some \(r \in \text{Config}_U \cup \{\text{error}\}\) by Prop 6.13 and either (Rf), or (Rferr).

Case (I(if)). \(t^U = \text{if}^U \varepsilon_1 t^{U_1} \) then \(\varepsilon_2 t^{U_2}\) else \(\varepsilon_3 t^{U_3}\) and

\[
\begin{align*}
\phi &\vdash t^{U_1} \in T[U_1] \\
\varepsilon_1 &\vdash U_1 \ll \text{Bool}_g \\
\phi' &= ((\phi, e \ll \text{ilbl}(\varepsilon_1))(\phi, g_c \ll \text{label}(U_1)), g_c \ll g) \\
\phi' &\vdash t^{U_2} \in T[U_2] \\
\phi' &\vdash t^{U_3} \in T[U_3] \\
\varepsilon_2 &\vdash U_2 \ll (U_2 \ll U_3) \\
\varepsilon_3 &\vdash U_3 \ll (U_2 \ll U_3) \\
\phi &\vdash \text{if}^U \varepsilon_1 t^{U_1} \text{ then} \varepsilon_2 t^{U_2} \text{ else } \varepsilon_3 t^{U_3} \in T[(U_2 \ll U_3) \ll g]
\end{align*}
\]

By induction hypothesis on \(t^{U_1}\), one of the following holds:

1. \(t^{U_1}\) is a value \(u\), then by (R\(\rightarrow\)), \(t^U | \mu \rightarrow^\phi r\) and \(r \in \text{Config}_U \cup \{\text{error}\}\) by Prop 6.13.
2. \(t^{U_1}\) is an ascribed value \(v\), then, \(\varepsilon_1 t^{U_1} \rightarrow_c et'\) for some \(et' \in \text{EvTerm} \cup \{\text{error}\}\). Hence \(t^U | \mu \rightarrow r\) for some \(r \in \text{Config}_U \cup \{\text{error}\}\) by Prop 6.13 and either (Rg), or (Rgerr).
3. \(t^{U_1} | \mu \rightarrow r_1\) for some \(r_1 \in T[U_1] \cup \{\text{error}\}\). Hence \(t^U | \mu \rightarrow r\) for some \(r \in \text{Config}_U \cup \{\text{error}\}\) by Prop 6.13 and either (Rf), or (Rferr).
Case (lapp). $t^U = \epsilon_1 t^{U_1} \oplus_{\epsilon_2} t^{U_2}$

\[
\begin{align*}
\phi \triangleright t^{U_1} \in T[U_1] & \quad \epsilon_1 \vdash U_1 \leq U_{11} \overset{g_c}{\rightarrow} U_{12} \\
\phi \triangleright t^{U_2} \in T[U_2] & \quad \epsilon_2 \vdash U_2 \leq U_{11} \\
(\text{lapp}) & \quad \epsilon_\ell \vdash g_c \forall g \leq g_c' \quad \phi.\ell \vdash \phi.g_c \overset{c}{\sim} \phi.g_c \\
& \quad \phi \triangleright \epsilon_1 t^{U_1} \oplus_{\epsilon_2} t^{U_2} \in T[U_{12} \setminus g] \quad \times
\end{align*}
\]

By induction hypothesis on $t^{U_1}$, one of the following holds:

1. $t^{U_1}$ is a value $(\lambda x^{U_1}. t^{U_1'}_{g'})$ (by canonical forms Lemma 6.10), posing $U_1 = U_{11} \overset{g_c''}{\rightarrow} U_{12}$.

Then by induction hypothesis on $t^{U_2}$, one of the following holds:

2a. $t^{U_2}$ is a value $u$, then by $(\text{R\rightarrow})$, $t^U | \mu \overset{\phi}{\leftarrow} r$ and $r \in \text{CONFIG}_U \cup \{\text{error}\}$ by Prop 6.13.

2b. $t^{U_2}$ is an ascribed value $v$, then, $\epsilon_\ell t^{U_2} \rightarrow_c e \ell'$ for some $e \ell' \in \text{EvTerm} \cup \{\text{error}\}$. Hence $t^U | \mu \overset{\phi}{\rightarrow} r$ for some $r \in \text{CONFIG}_U \cup \{\text{error}\}$ by Prop 6.13 and either (Rg), or (Rgerr).

2c. $t^{U_2} | \mu \overset{\phi}{\rightarrow} r_2$ for some $r_2 \in \text{CONFIG}_U \cup \{\text{error}\}$. Hence $t^U | \mu \overset{\phi}{\rightarrow} r$ for some $r \in \text{CONFIG}_U \cup \{\text{error}\}$ by Prop 6.13 and either (Rf), or (Rferr). Also by Prop 6.13, if $r = t^{U_1} | \mu' \in T[U]$ then $\text{dom}(\mu) \subseteq \text{dom}(\mu')$.

Case (\text{Ideref}). Similar case to (lapp).

Case (\text{Iref}). $t^U = \text{ref}_{\ell_2} t^{U''}$ and

\[
\begin{align*}
\phi \triangleright t^{U''} \in T[U''] & \quad \epsilon \vdash \text{ref}_{\ell_2} t^{U''} \in T[\text{Ref}_U U''] \\
(\text{Iref}) & \quad \epsilon \vdash U'' \leq U' \quad \epsilon \vdash g_c \leq \text{label}(U') \\
& \quad \phi \triangleright \text{ref}_{\ell_2} t^{U''} \in T[\text{Ref}_U U''] \quad \times
\end{align*}
\]

By induction hypothesis on $t^{U''}$, one of the following holds:

1. $t^{U''}$ is a value $v$, then by $(\text{R\rightarrow})$, $t^{U''} | \mu \overset{\phi}{\rightarrow} r$ and $r \in \text{CONFIG}_U'$ by Prop 6.13. Also by Prop 6.13, if $r = t^{U''} | \mu' \in T[U]$ then $\text{dom}(\mu) \subseteq \text{dom}(\mu')$.

2. $t^{U''}$ is an ascribed value $v$, then, $\epsilon t^{U_1} \rightarrow_c e \ell'$ for some $e \ell' \in \text{EvTerm} \cup \{\text{error}\}$. Hence $t^U | \mu \overset{\phi}{\rightarrow} r$ for some $r \in \text{CONFIG}_U \cup \{\text{error}\}$ by Prop 6.13 and either (Rg), or (Rgerr).

3. $t^{U''} | \mu \overset{\phi}{\rightarrow} r_1$ for some $r_1 \in \text{CONFIG}_U' \cup \{\text{error}\}$. Hence $t^{U''} | \mu \overset{\phi}{\rightarrow} r$ for some $r \in \text{CONFIG}_U \cup \{\text{error}\}$ by Prop 6.13 and either (Rf), or (Rferr). Also by Prop 6.13, if $r = t^{U''} | \mu' \in T[U]$ then $\text{dom}(\mu) \subseteq \text{dom}(\mu')$.

Case (Ideref). $t^U = \text{ref}_{\ell_2} t^{U''}$

\[
\begin{align*}
\phi \triangleright t^{U''} \in T[U''] & \quad \epsilon \vdash U'' \leq \text{Ref}_U U' \\
(\text{Iref}) & \quad \phi \triangleright \text{ref}_{\ell_2} t^{U''} \in T[U' \setminus g] \quad \times
\end{align*}
\]
By induction hypothesis on \( t'' \), one of the following holds:

1. \( t'' \) is a value \( t''' \) (by canonical forms Lemma 6.10), where \( U''' = \text{Ref}_{g'} U''' \), then by (R→→),
   \[
   t'' \mid \mu \xrightarrow{\phi} r \quad \text{and} \quad r \in \text{CONFIG}_U
   \]
   by Prop 6.13.

2. \( t'' \) is an ascribed value \( v \), then, \( \epsilon t'' \rightarrow_c et' \) for some \( et' \in \text{EvTerm} \cup \{ \text{error} \} \). Hence
   \[
   t'' \mid \mu \xrightarrow{\phi} r \quad \text{for some} \quad r \in \text{CONFIG}_U \cup \{ \text{error} \}
   \]
   by Prop 6.13 and either (Rg), or (Rgerr).

3. \( t'' \mid \mu \xrightarrow{\phi} r_i \) for some \( r_i \in \text{CONFIG}_U \cup \{ \text{error} \} \). Hence \( t'' \mid \mu \xrightarrow{\phi} r \) for some \( r \in \text{CONFIG}_U \cup \{ \text{error} \} \) by Prop 6.13 and either (Rf), or (Rferr). Also by Prop 6.13, if \( r = t'' \mid \mu' \in T[U] \) then \( \text{dom}(\mu) \subseteq \text{dom}(\mu') \).

Case (Uassign). \( t'' = \epsilon_1 t''_{U_1} \xrightarrow{g_{U_1}} \epsilon_2 t''_{U_2} \) and

\[
\begin{align*}
\epsilon_1 + \text{Ref}_{g'} U_1' &\leq \text{Ref}_g U_1 \quad \phi \triangleright t''_{U_1} \in T[\text{Ref}_{g'} U_1'] \\
\epsilon_2 + U_2 &\leq U_1 \\
\phi \circ \epsilon + \phi \circ g_{c_1} &\leq \phi \circ g_{c_2} \\
\phi \circ \epsilon + \phi \circ g &\leq \text{label}(U_1)
\end{align*}
\]

By induction hypothesis on \( t''_{U_1} \), one of the following holds:

1. \( t''_{U_1} \) is a value \( U_{1''} \) (by canonical forms Lemma 6.10), where \( U_{1''} = \text{Ref}_{g'} U_{1'''} \). Then by induction hypothesis on \( t''_{U_2} \), one of the following holds:
   
   a) \( t''_{U_2} \) is a value \( u \), then by (R→→), \( t''_{U_2} \mid \mu \xrightarrow{\phi} r \) and \( r \in \text{CONFIG}_U \cup \{ \text{error} \} \) by Prop 6.13. Also by Prop 6.13, if \( r = t''_{U_2} \mid \mu' \in T[U] \) then \( \text{dom}(\mu) \subseteq \text{dom}(\mu') \).

   b) \( t''_{U_2} \) is an ascribed value \( v \), then, \( \epsilon_2 t''_{U_2} \rightarrow_c et' \) for some \( et' \in \text{EvTerm} \cup \{ \text{error} \} \). Hence
   \[
   t''_{U_2} \mid \mu \xrightarrow{\phi} r \quad \text{for some} \quad r \in \text{CONFIG}_U \cup \{ \text{error} \}
   \]
   by Prop 6.13 and either (Rg), or (Rgerr).

   c) \( t''_{U_2} \mid \mu \xrightarrow{\phi} r_2 \) for some \( r_2 \in \text{CONFIG}_U \cup \{ \text{error} \} \). Hence \( t''_{U_2} \mid \mu \xrightarrow{\phi} r \) for some \( r \in \text{CONFIG}_U \cup \{ \text{error} \} \) by Prop 6.13 and either (Rf), or (Rferr). Also by Prop 6.13, if \( r = t''_{U_2} \mid \mu' \in T[U] \) then \( \text{dom}(\mu) \subseteq \text{dom}(\mu') \).

2. \( t''_{U_1} \) is an ascribed value \( v \), then, \( \epsilon_1 t''_{U_1} \rightarrow_c et' \) for some \( et' \in \text{EvTerm} \cup \{ \text{error} \} \). Hence
   \[
   t''_{U_1} \mid \mu \xrightarrow{\phi} r \quad \text{for some} \quad r \in \text{CONFIG}_U \cup \{ \text{error} \}
   \]
   by Prop 6.13 and either (Rg), or (Rgerr).

3. \( t''_{U_1} \mid \mu \xrightarrow{\phi} r_1 \) for some \( r_1 \in \text{CONFIG}_U \cup \{ \text{error} \} \). Hence \( t''_{U_1} \mid \mu \xrightarrow{\phi} r \) for some \( r \in \text{CONFIG}_U \cup \{ \text{error} \} \) by Prop 6.13 and either (Rf), or (Rferr). Also by Prop 6.13, if \( r = t''_{U_1} \mid \mu' \in T[U] \) then \( \text{dom}(\mu) \subseteq \text{dom}(\mu') \).

\[ \square \]

**Proposition 6.15 (Static terms do not fail).** Let us define StaticTerm the set of evidence augmented terms with full static annotations. Then consider \( t_s \in \text{StaticTerm}, \phi = (\epsilon t_c', \ell_c) \), and \( \mu_s \), such that \( \epsilon = \gamma[\ell_c' \leq \ell_c], \phi \triangleright t_s \in T[S], \) and that \( \forall \epsilon_s \in \text{cod}(\mu_s), \epsilon_s \in \text{StaticTerm}. \) Then either \( t_s \) is a value, or

\[
\begin{align*}
t_s \mid \mu_s &\xrightarrow{\phi} t'_s \mid \mu'_s
\end{align*}
\]

**Proof.** We know that if you follow AGT to derive the dynamic semantics of a gradual language, then by construction the resulting language satisfy the dynamic conservative extension property. As we follow AGT to derive the dynamic semantics, we get this property by construction, save for the assignment elimination reduction rule. In this rule we add an extra check of the form \( \phi \circ \epsilon \leq \text{ilbl}(\epsilon) \). So if we prove that the extra check is always satisfied, then the result holds.
Let us consider a $t'_1$ fully static like so:

$$\begin{align*}
&\varepsilon_1 \vdash \text{Ref}_\ell S'_1 \leq \text{Ref}_\ell S_1 \\
&\varepsilon_2 \vdash S_2 \leq S_1 \\
&\phi \triangleright u \in \mathbb{T}[S_2] \\
\text{(lassn)} &\quad \phi, \varepsilon \vdash t'_2 \succeq \ell_c \\
&\phi \triangleright \varepsilon_1 o'_{S'_1} t'_2 \vdash t'_{S_i} : \varepsilon_1, \varepsilon_2 \vdash u \in \mathbb{T}[\text{Unit}_1]
\end{align*}$$

By inspection of the reduction rules we have to prove that $\phi, \varepsilon [\leq] \text{ilbl}(\ell)$. $\phi, \varepsilon [\leq] \text{ilbl}(\ell)$. We know by definition of interior between two static labels that $\varepsilon = \mathcal{G}[\ell'_c \preceq \ell_c] = \langle \ell'_c, \ell_c \rangle$. Also, if $\mu_s(o^{S_i}) = \varepsilon U' : S'_1$, as everything is static, $\text{ilbl}(\ell)$ have to have the form $\langle \ell, \ell' \rangle$, $\text{ilbl}(\ell')$, for some $\ell_u$. Then we have to prove that $\ell_c \preceq \text{label}(S'_1)$, but notice that as everything is static, $\ell'_c \preceq \ell$ is equivalent to $\ell_c \preceq \ell$, therefore we know that $\ell_c \preceq \text{label}(S'_1)$ and the result holds.

\[\square\]

### 6.5 Dynamic Gradual Guarantee

In this section we present the proof the Dynamic Gradual Guarantee for GSLRef without the specific check in rule (7).

**Definition 6.16 (Intrinsic term precision).** Let

$$\Omega = \mathcal{P}(\mathbb{V}[\star] \times \mathbb{V}[\star]) \cup \mathcal{P}(\mathbb{Loc}_s \times \mathbb{Loc}_s)$$

be defined as $\Omega := \{ x^{U_1} \sqsubseteq x^{U_2}, o^{U_1} \sqsubseteq o^{U_2} \}$ We define an ordering relation $(\cdot \vdash \cdot \sqsubseteq \cdot) \in (\mathcal{P}(\mathbb{V}[\star] \times \mathbb{V}[\star]) \cup \mathcal{P}(\mathbb{Loc}_s \times \mathbb{Loc}_s)) \times \mathbb{T}[\star] \times \mathbb{T}[\star]$ shown in Figure 29.

**Definition 6.17 (Well Formedness of $\Omega$).** We say that $\Omega$ is well formed iff $\forall \{ t^{U_1} \sqsubseteq t^{U_2} \} \in \Omega. U_1 \sqsubseteq U_2$

Before proving the gradual guarantee, we first establish some auxiliary properties of precision. For the following propositions, we assume Well Formedness of $\Omega$ (Definition 6.17).

**Proposition 6.18.** If $\Omega \vdash t^{U_1} \sqsubseteq t^{U_2}$ for some $\Omega \in \mathcal{P}(\mathbb{V}[\star] \times \mathbb{V}[\star]) \cup \mathcal{P}(\mathbb{Loc}_s \times \mathbb{Loc}_s)$, then $U_1 \sqsubseteq U_2$.

**Proof.** Straightforward induction on $\Omega \vdash t^{U_1} \sqsubseteq t^{U_2}$, since the corresponding precision on types is systematically a premise (either directly or transitively). \[\square\]

**Proposition 6.19.** Let $g_1, g_2 \in \text{EvFrame}$ such that $\phi_1 \triangleright g_1[\varepsilon_{11} t^{U_1}] \in \mathbb{T}[U'_1], \phi_2 \triangleright g_2[\varepsilon_{21} t^{U_2}] \in \mathbb{T}[U'_2]$, with $U'_1 \sqsubseteq U'_2$. Then if $g_1[\varepsilon_{11} t^{U_1}] \sqsubseteq g_2[\varepsilon_{21} t^{U_2}], \varepsilon_{12} \sqsubseteq \varepsilon_{22}$ and $t^{U_1} \sqsubseteq t^{U_2}$, then $g_1[\varepsilon_{12} t^{U_1}] \sqsubseteq g_2[\varepsilon_{22} t^{U_2}]$

**Proof.** We proceed by case analysis on $g_i$.

Case $(\square @^U \varepsilon t)$. Then for $i \in \{1, 2\}$ $g_i$ must have the form $\square @^U \varepsilon_i t^{U_i}$ for some $U''', \varepsilon_i$ and $t^{U_i}$. As $g_1[\varepsilon_{11} t^{U_1}] \sqsubseteq g_2[\varepsilon_{21} t^{U_2}]$ then by $\mathbb{APP} \varepsilon_1 \sqsubseteq \varepsilon_2, \varepsilon_1' \sqsubseteq \varepsilon_2', U'''' \sqsubseteq U''''$ and $t^{U_i} \sqsubseteq t^{U_i}$.

As $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$ and $t^{U_1} \sqsubseteq t^{U_2}$, then by $\mathbb{APP} \varepsilon_{12} t^{U_1} @^{U_{12}} \varepsilon_i' t^{U_i} \sqsubseteq \varepsilon_{22} t^{U_2} @^{U_{12}} \varepsilon_i' t^{U_i}$, and the result holds.

Case $(\square \triangleright g \varepsilon, ev \triangleright g \square, ev @^{U_1} \square, \square :: U, \sqsubset^{U_1} \square, \square \triangleright \sqsubseteq^{U_1} \varepsilon t, ev \triangleright \varepsilon \varepsilon_1 \varepsilon_1, \text{if} g \varepsilon \text{then et else et})$. Straightforward using similar argument to the previous case. \[\square\]
\[
\begin{array}{c}
\Omega \cup \{x^{U_1} \subseteq x^{U_2}\} \vdash x^{U_1} \subseteq x^{U_2} \\
g_1 \subseteq g_2 \\
\Omega \cup \{o^{U_1} \subseteq o^{U_2}\} \vdash o^{U_1} \subseteq o^{U_2} \\
g_1 \subseteq g_2 \\
\Omega \vdash \text{prot}_{\epsilon_1} \phi_1^{u_1}(\epsilon_1 t^{U_1}_1) \subseteq \text{prot}_{\epsilon_2} \phi_2^{u_2}(\epsilon_2 t^{U_2}_1) \\
g_{c_1} \subseteq g_{c_2} \\
\Omega \vdash t^{U_1_1} \subseteq t^{U_1_2} \\
\epsilon_{11} \subseteq \epsilon_{21} \\
\Omega \vdash t^{U_1_2} \subseteq t^{U_2_2} \\
\epsilon_{12} \subseteq \epsilon_{22} \\
\Omega \vdash t^{U_1_3} \subseteq t^{U_2_3} \\
\epsilon_{13} \subseteq \epsilon_{23} \\
\Omega \vdash \text{ref}_{\epsilon_1} t^{U_1}_1 \subseteq \text{ref}_{\epsilon_2} t^{U_2}_1 \\
\end{array}
\]

where $\phi_1 \equiv \phi_2 \iff \phi_1.\epsilon \subseteq \phi_2.\epsilon \land \phi_1.\!g_{c} \subseteq \phi_2.\!g_{c} \land \phi_1.\!g_{c} \subseteq \phi_2.\!g_{c}$

\[
\begin{array}{c}
g_1 \subseteq g_2 \\
\Omega + b_{g_1} \subseteq b_{g_2} \\
\Omega + \text{unit}_{g_1} \subseteq \text{unit}_{g_2} \\
\Omega + \text{unit}_{g_2} \\
\end{array}
\]

\[
\begin{array}{c}
U_{11} \subseteq U_{12} \\
g_{c_1} \subseteq g_{c_2} \\
g_1 \subseteq g_2 \\
\Omega \cup \{x^{U_{11}} \subseteq x^{U_{12}}\} \vdash t_{U_{12}} \subseteq t_{U_{22}} \\
\Omega \vdash (\lambda x^{U_{11}}.t^{U_{11}})_{g_1} \subseteq (\lambda x^{U_{21}}.t^{U_{21}})_{g_2} \\
\end{array}
\]

\[
\begin{array}{c}
U_{12} \subseteq U_{22} \\
\epsilon_{11} \subseteq \epsilon_{21} \\
\Omega \vdash t^{U_{11}} \subseteq t^{U_{21}} \\
(\epsilon_1 t^{U_{11}} \vdash U_{12}) \subseteq (\epsilon_2 t^{U_{21}} \vdash U_{22}) \\
\end{array}
\]

**Proposition 6.20.** Let $g_1, g_2 \in EvFrame$ such that $\phi_1 \triangleright g_1[\epsilon_1 t^{U_1}] \in T[U_1'], \phi_2 \triangleright g_2[\epsilon_2 t^{U_2}] \in T[U_2']$, with $U_1' \subseteq U_2'$. Then if $g_1[\epsilon_1 t^{U_1}] \subseteq g_2[\epsilon_2 t^{U_2}]$ then $t^{U_1} \subseteq t^{U_2}$ and $\epsilon_1 \subseteq \epsilon_2$.

**Proof.** We proceed by case analysis on $g_1$. Fig. 29. Intrinsic term precision
Proof. By definition of join and the definition of precision.

**Proposition 6.22.** Let \( f_1, f_2 \in \text{EvFrame} \) such that \( \phi_1 \triangleright f_1[t_1^{U_i}] \in T[U'_i], \phi_2 \triangleright t_2^{U_i} \in T[U'_i] \), with \( U'_i \subseteq U'_j \). Then if \( f_1[t_1^{U_i}] \sqsubseteq f_2[t_1^{U_i}] \) and \( t_2^{U_i} \sqsubseteq t_2^{U_j} \), then \( f_1[t_1^{U_i}] \sqsubseteq f_2[t_1^{U_j}] \).

**Proof.** Suppose \( f_1[t_1^{U_i}] = g_1[e_1 t_1^{U_i}] \). We know that \( \phi_1 \triangleright g_1[e_1 t_1^{U_i}] \in T[U'_i], \phi_2 \triangleright g_2[e_2 t_2^{U_i}] \in T[U'_i] \) and \( U'_i \subseteq U'_j \). Therefore if \( g_1[e_1 t_1^{U_i}] \sqsubseteq g_1[e_1 t_1^{U_j}] \), by Prop 6.20, \( e_1 \sqsubseteq e_2 \). Finally by Prop 6.19 we conclude that \( g_1[e_1 t_2^{U_i}] \sqsubseteq g_1[e_1 t_2^{U_j}] \).

**Proposition 6.23 (Substitution preserves precision).** If \( \Omega \cup \{ x^{U_i} \subseteq x^{U_i} \} \vdash t^{U_i} \sqsubseteq t^{U_j} \) and \( \Omega \vdash t^{U_i} \sqsubseteq t^{U_j} \), then \( \Omega \vdash [t^{U_i} / x^{U_i}]^{t^{U_i}} \sqsubseteq [t^{U_j} / x^{U_j}]^{t^{U_j}} \).

**Proof.** By induction on the derivation of \( t^{U_i} \sqsubseteq t^{U_j} \) and case analysis of the last rule used in the derivation. All cases follow either trivially (no premises) or by the induction hypotheses.

**Proposition 6.24 (Monotone precision for \( \triangleright \)).** If \( e_1 \sqsubseteq e_2 \) and \( e_2 \sqsubseteq e_3 \) then \( e_1 \triangleright e_3 \sqsubseteq e_2 \triangleright e_4 \).

**Proof.** By definition of consistent transitivity for \( \triangleright \) and the definition of precision.

**Proposition 6.25 (Monotone precision for \( \triangleright \)).** If \( e_1 \sqsubseteq e_2 \) and \( e_3 \sqsubseteq e_4 \) then \( e_1 \triangleright e_3 \sqsubseteq e_2 \triangleright e_4 \).

**Proof.** By definition of consistent transitivity for \( \triangleright \) and the definition of precision.

**Proposition 6.26 (Monotone precision for join).** If \( e_1 \sqsubseteq e_2 \) and \( e_3 \sqsubseteq e_4 \) then \( e_1 \triangleright e_3 \sqsubseteq e_2 \triangleright e_4 \).

**Proof.** By definition of join and the definition of precision.

**Proposition 6.27.** If \( \text{Ref } U_1 \sqsubseteq \text{Ref } U_2 \) then \( U_1 \sqsubseteq U_2 \).

**Proof.** By definition of precision we know that \( \{ \text{Ref } T \mid T \in \gamma(U_1) \} \sqsubseteq \{ \text{Ref } T \mid T \in \gamma(U_2) \} \). This relation is true only if \( \gamma(U_1) \subseteq \gamma(U_2) \) which is equivalent to \( U_1 \sqsubseteq U_2 \).

**Proposition 6.28.** If \( U_{11} \sqsubseteq U_{12} \) and \( U_{21} \sqsubseteq U_{22} \), then \( U_{11} \triangleright U_{12} \sqsubseteq U_{21} \triangleright U_{22} \).

**Proof.** By induction on the type derivation of the types and consistent join.
Lemma 6.29. If $e_1 \vdash \text{Ref}_{g_{11}} U_{11} \sqsubseteq \text{Ref}_{g_{12}} U_{12}$ and $e_2 \vdash \text{Ref}_{g_{21}} U_{21} \sqsubseteq \text{Ref}_{g_{22}} U_{22}$, and $e_1 \sqsubseteq e_2$, then $\text{iref}(e_1) \sqsubseteq \text{iref}(e_2)$.

Proof. By definition of precision and $\text{iref}$. □

Proposition 6.30 (Dynamic guarantee for $\concat$). Suppose $\Omega \vdash t_{1U_1} \sqsubseteq t_{2U_2}$, $\phi_1 \sqsubseteq \phi_2$, and $\Omega \vdash \mu_1 \sqsubseteq \mu_2$. If $t_{1U_1} \mid \mu_1 \xrightarrow{\phi_1} t_{1U_1} \mid \mu_1'$ then $t_{1U_2} \mid \mu_2 \xrightarrow{\phi_2} t_{2U_2} \mid \mu_2'$ where $\Omega' \vdash t_{1U_1} \sqsubseteq t_{2U_2}$ and $\Omega' \vdash \mu_1' \sqsubseteq \mu_2'$, for some $\Omega' \supseteq \Omega$.

Proof. By induction on the structure of $t_{1U_1} \sqsubseteq t_{2U_2}$. For simplicity we omit the $\Omega \vdash$ notation on precision relations when it is not relevant for the argument.

Case ($\concat$). We know that $t_{1U_1} = (e_{11}(b_{11})_{g_{11}} \oplus e_{12}(b_{12})_{g_{12}})$ then by ($\sqsubseteq\concat$) $t_{U_2} = (e_{21}(b_{11})_{g_{21}} \oplus e_{22}(b_{12})_{g_{22}})$ for some $e_{21}, e_{22}, g_{21}, g_{22}$ such that $e_{11} \subseteq e_{21}, e_{12} \subseteq e_{22}, g_{11} \sqsubseteq g_{21}$ and $g_{12} \sqsubseteq g_{22}$.

If $t_{1U_1} \mid \mu_1 \xrightarrow{\phi_1} b_3 \mid \mu_1$ where $b_3 = (e_{11} \lor e_{12})(b_1 \oplus b_2)_{(g_{11} \lor g_{21})} : \text{Bool}_{g_{11}}$, then

$t_{1U_1} \mid \mu_2 \xrightarrow{\phi_2} b'_3 \mid \mu_2$ where $b'_3 = (e_{21} \lor e_{22})(b_1 \oplus b_2)_{(g_{11} \lor g_{21})} : \text{Bool}_{g_{22}}$. By Lemma 6.26, $(e_{11} \lor e_{12}) \sqsubseteq (e_{21} \lor e_{22})$. Also $(g_{11} \lor g_{21}) \subseteq (g_{21} \lor g_{22})$.

\[
\begin{array}{c}
\Omega \sqsubseteq (b_1 \oplus b_2)_{(g_{11} \lor g_{21})} \subseteq (b_1 \oplus b_2)_{(g_{11} \lor g_{21})} \sqsubseteq (e_{21} \lor e_{22})
\end{array}
\]


Therefore $t_{1U_1} \sqsubseteq t_{2U_2}$. As $\Omega' = \Omega, \mu_1' = \mu_1$ and $\mu_2' = \mu_2$ then $\Omega' \vdash \mu_1' \sqsubseteq \mu_2'$.

Case ($\concat\text{prot}$). We know that $t_{1U_1} = \text{prot}_{\text{Ref}U_{11}} \phi_1'(e_1 u_1)$, then by ($\text{Ref}_{\sqsubseteq\text{prot}1}$) $t_{1U_2} = \text{prot}_{\text{Ref}U_{22}} \phi_2'(e_2 u_2)$, and therefore

\[
\begin{array}{c}
g_1 \sqsubseteq g_2
\end{array}
\]

\[
\begin{array}{cc}
\phi_1' \sqsubseteq \phi_2' & \epsilon_1 \sqsubseteq \epsilon_2
\end{array}
\]

\[
\begin{array}{c}
\Omega \sqsubseteq \text{prot}_{\text{Ref}U_{11}} \phi_1'(e_1 u_1) \sqsubseteq \text{prot}_{\text{Ref}U_{22}} \phi_2'(e_2 u_2)
\end{array}
\]

for some $\epsilon_2, u_2, U_2$ and $\epsilon_{\text{Ref}2}$, where $u_1 \in T[U_{1}]$ and $u_2 \in T[U_{2}]$. If

$t_{1U_1} \mid \mu_1 \xrightarrow{\phi_1} (e_{12} \lor e_{13})(u_1 \lor g'_1) \sqsubseteq U_1 \lor g_1 \mid \mu_1$. Therefore, $t_{1U_2} \mid \mu_2 \xrightarrow{\phi_2} (e_{22} \lor e_{23})(u_2 \lor g'_2) \sqsubseteq U_2 \lor g_2 \mid \mu_2$.

By Lemma 6.26, $(e_{13} \lor e_{12}) \sqsubseteq (e_{23} \lor e_{22})$, and as join is monotone $U_1 \lor g_1 \sqsubseteq U_2 \lor g_2$ and $(u_1 \lor g'_1) \sqsubseteq (u_2 \lor g'_2)$. Therefore by $\sqsubseteq\text{Ref}$, $(e_{13} \lor e_{12})(u_1 \lor g'_1) \sqsubseteq U_1 \lor g_1 \sqsubseteq (e_{22} \lor e_{23})(u_2 \lor g'_2) \sqsubseteq U_2 \lor g_2$. As $\Omega' = \Omega, \mu_1' = \mu_1$ and $\mu_2 = \mu_2'$ then $\Omega' \vdash \mu_1' \sqsubseteq \mu_2'$.

Case ($\concat\text{app}$). We know that

$t_{1U_1} = e_{11}(\lambda xU_{11}. t_{1U_{12}})_{g'_1} @ e_{12}U_{12} \epsilon_{12}u$ then by ($\sqsubseteq\text{app}$) $t_{1U_2}$ must have the form

\[
\begin{array}{c}
t_{1U_1} = e_{21}(\lambda xU_{21}. t_{2U_{12}})_{g'_2} @ e_{22}U_{22} \epsilon_{22}u_2
\end{array}
\]

for some $e_{21}, xU_{21}, t_{2U_{12}}, U_3, U_4, e_{22}, g'_{22}, g_2$ and $u_2$.

Let us pose $e_1 = e_{12} \oslash\text{idom}(e_{11})$ and $e_1' = (\phi_1, e \lor \text{ilbl}(e_{11})) \circ e_{12} \oslash\text{ilat}(e_{11})$, $\phi'_1 = (e'_1, g'_1 \lor \text{ilbl}(e_{11}), g_{11} \lor \text{ilat}(e))$. Then

$t_{1U_1} \mid \mu_1 \xrightarrow{\phi_1} \text{prot}_{\text{ilbl}(e_{11})} \phi'_1(\text{icod}(e_{11})) \mid \mu_1$ with $t'_1 = [(e_1 u_1 :: U_{11}) / xU_{11}]_l t_{1U_{12}}$.

Also, let us pose $e_2 = e_{22} \oslash\text{idom}(e_{21})$ and $e_2' = (\phi_2, e \lor \text{ilbl}(e_{21})) \circ e_{22} \oslash\text{ilat}(e_{21})$, $\phi'_2 = (e'_2, g'_2 \lor \text{ilbl}(e_{21}), g_{21} \lor \text{ilat}(e_{21})$, $\phi'_2 = (e'_2, g'_2 \lor \text{ilbl}(e_{21})$, $\phi'_2 \circ e_{22} \oslash\text{ilat}(e_{21})$.  \]
\langle \epsilon'_1 (g'_2 \simeq \phi_2.g_c), g_2 \simeq \phi_2.g_c \rangle. \text{ Then } t_{U_1}^1 \parallel \mu_2 \xrightarrow{\phi_2} \text{ prot}_{\text{libl}(\epsilon_1)} g'_2 (\text{icod}(\epsilon_2)) t'_2 \parallel \mu_2 \text{ with } t'_2 = [(\epsilon_2 u_2 : U_2) / x^{U_2}] t'^{U_2}.

As \Omega \vdash t_{U_1}^1 U_1 \notin t^{U_1}_1, \text{ then } u_1 \notin u_2, \epsilon_{12} \subseteq \epsilon_{22} \text{ and } \text{idom}(\epsilon_{11}) \subseteq \text{idom}(\epsilon_{21}) \text{ as well, then by Prop 6.24 } \epsilon_1 \subseteq \epsilon_2. \text{ Then } \epsilon_1 u_1 : U_1 \equiv \epsilon_2 u_2 : U_2 \text{ by } (\equiv_3).

We also know by \((\equiv_{\text{APP}})\) and \((\equiv_3)\) that \(\Omega \cup \{x^{U_1}, x^{U_2}\} \vdash t^{U_1}, t^{U_2} \). \text{ By Substitution preserves precision (Prop } 6.23 \text{) } t'_1 \equiv t'_2, \text{ therefore } \text{icod}(\epsilon_{11}) t'_1 \equiv U_2 \equiv \text{icod}(\epsilon_{21}) t'_2 \equiv U_4 \text{ by } (\equiv_3). \text{ Also } g_1 \subseteq g_2, \text{ libl}(\epsilon_{11}) \subseteq \text{libl} q_2, \epsilon'_{12} \subseteq \epsilon'_{22} \text{ and by Lemma } 6.24 \text{ and } 6.26, \epsilon'_{12} \subseteq \epsilon'_{22}. \text{ Also, as } \phi_1.g_c \subseteq \phi_2.g_c \text{ by monotonicity of the join } g_1 \simeq \phi_1.g_c \subseteq g_2 \simeq \phi_2.g_c \text{ and as } \phi_1.g_c \subseteq \phi_2.g_c \text{ also by monotonicity of the join } g'_1 \simeq \phi_1.g_c \subseteq g'_2 \simeq \phi_2.g_c. \text{ Then by } (\equiv_{\text{prot}(l)}), t_{U_1}^1 \equiv t_{U_2}^1. \text{ As } \Omega' = \Omega, \mu'_1 = \mu_1 \text{ and } \mu_2 = \mu'_2 \text{ then } \\
\Omega' + \mu'_1 \subseteq \mu'_2.

Case (\text{---if-true}). \text{ Then } t_{U_1}^1 = \text{if}^g \epsilon_{11} \text{true}_g \text{ then } \text{else } \epsilon_{12} t^{U_1} e_{13} t^{U_1} \text{ then by } (\equiv_{\text{f}}) t_{U_1}^1 \text{ has the form } \\
t_{U_1}^1 = \text{if}^g \epsilon_{21} \text{true}_g \text{ then } \text{else } \epsilon_{22} t^{U_2} e_{23} t^{U_2} \text{ for some } \\
\epsilon_{21}, \epsilon_{22}, \epsilon_{23}, \text{ and } t^{U_2}. \text{ Then } t_{U_1} \parallel \mu_1 \xrightarrow{\phi_1} \text{ prot}_{\text{libl}(\epsilon_{11})} g'_1 (\epsilon_{12} U^{U_1}_1) \parallel \mu_1, \text{ and } \\
t_{U_1} \parallel \mu_2 \xrightarrow{\phi_2} \text{ prot}_{\text{libl}(\epsilon_{22})} g'_2 (\epsilon_{22} U^{U_2}_2) \parallel \mu_2.

Where \( g'_1 = ((\phi_1.\epsilon \simeq \text{icod}(\epsilon_{12}))(g'_1 \simeq \phi_1.g_c), (\phi_1.g_c \simeq g_1)). \text{ Using the fact that } t_{U_1} \equiv \epsilon_{U_2} \text{ we know that } \\
\epsilon_{12} \subseteq \epsilon_{22}, t^{U_1} \equiv t^{U_2}, g'_1 \subseteq g'_2, \text{ as } \phi_1.g_c \subseteq \phi_2.g_c \text{ and } g_1 \subseteq g_2, \text{ and as join is monotone, } \phi_1.g_c \simeq \text{libl}(\epsilon_{12}) \subseteq \phi_2.g_c \simeq g_2. \text{ Also } \phi_1.g_c \subseteq \phi_2.g_c \text{ and } g'_1 \subseteq g'_2, \text{ as join is monotone, } \\
\phi_1.g_c \simeq \text{libl}(\epsilon_{22}) \subseteq \phi_2.g_c \simeq g_2. \text{ By } \text{Prop } 6.18, \text{ we know that } U_2 \equiv U_2 \text{ and } U_2 \equiv U_2. \text{ Therefore by Prop } 6.28 (U_2 \equiv U_2) \subseteq (U_2 \equiv U_2). \text{ Also } \phi_1.\epsilon \subseteq \phi_2.\epsilon \text{ and } \text{libl}(\epsilon_{12}) \subseteq \text{libl}(\epsilon_{22}) \text{ then by Lemma } 6.26 (\phi_1.\epsilon \simeq \text{libl}(\epsilon_{12})) \subseteq (\phi_2.\epsilon \simeq \text{libl}(\epsilon_{22})). \text{ Then using } (\equiv_{\text{prot}(l)}), t_{U_1} \equiv t_{U_2}. \text{ As } \Omega'' = \Omega, \mu'_1 = \mu_1 \text{ and } \mu_2 = \mu'_2 \text{ then } \\
\Omega'' + \mu'_1 \subseteq \mu'_2.

Case (\text{---if-false}). \text{ Same as case ---if-true, using the fact that } e_{13} \subseteq e_{23} \text{ and } t^{U_1} \equiv t^{U_2}.

Case (\text{---ref}). \text{ We know that } t_{U_1}^1 = \text{ref}_{e'_{12}} e_{11} U_{11} \text{, then by } (\equiv_{\text{ref}}) t_{U_2}^1 = \text{ref}_{e'_{22}} e_{22} U_{22} \text{, and therefore } \\
U_{11}'' U_{11} \equiv U_{22}'' U_{12} \equiv U_{12}'' e_{11} \equiv e_{12} || e_{11} = e_{12} \equiv e_{22} \equiv e_{21} = e_{22} \equiv e_{12} (e_1.\epsilon \simeq \text{libl}(\epsilon_{12})). \text{ By Lemma } 6.26 \text{ and } 6.24, e_{11} \equiv e_{12}, \text{ and } \phi_1.\epsilon \subseteq \phi_2.\epsilon \text{ and } \text{libl}(\epsilon_{12}) \subseteq \text{libl}(\epsilon_{22}) \text{ then by } (\equiv_{\text{prot}(l)}), t_{U_1} \equiv t_{U_2}. \text{ As } \Omega'' = \Omega, \mu'_1 = \mu_1 \text{ and } \mu_2 = \mu'_2 \text{ then } \\
\Omega'' + \mu'_1 \subseteq \mu'_2.

Case (\text{---deref}). \text{ We know that } t_{U_1}^1 = (\text{Ref}_n U_{11}'' e_{11} U_{11}'' \equiv t_{U_1}^1 = (\text{Ref}_n U_{11}'' e_{11} U_{11}'' \text{, and so } \text{Ref}_n U_{11}'' e_{11} U_{11}'' \equiv \text{Ref}_n U_{22}'' e_{22} U_{22}'' \text{. As } \Omega + \mu_1 \equiv \mu_2, \text{ using } (\equiv_\mu) \text{ then } \Omega + \mu_1 (U_{11}'' \equiv \mu_2 (U_{22}'' \text{. Then } \\
(\text{Ref}_n U_{11}'') \equiv \mu_1 \xrightarrow{\phi_1} \text{ prot}_{\text{libl}(\epsilon_{11})} g'_1 (\text{icod}(\epsilon_1) \mu_1 (o_{1''})) \text{ where } c_{11} = \text{libl}(\epsilon_1). \text{ Therefore } \\
(\text{Ref}_n U_{22}'') \equiv \mu_2 \xrightarrow{\phi_2} \text{ prot}_{\text{libl}(\epsilon_2)} g'_2 (\text{icod}(\epsilon_2) \mu_2 (o_{2''})) \text{ where } c_{22} = \text{libl}(\epsilon_2).
Where $\phi'_1 = (\phi_1, \epsilon \wedge \epsilon'_1)(\phi_1, \epsilon \wedge g'_1), \phi_1, \epsilon \wedge g'_1).$ By monotonicity of the join $\phi_1, \epsilon \wedge g_1 \sqsubseteq \phi_2, \epsilon \wedge g_2, \phi_1, \epsilon \wedge g'_1 \sqsubseteq \phi_2, \epsilon \wedge g_2'$ and $(\phi_1, \epsilon \wedge \epsilon'_1) \sqsubseteq (\phi_2, \epsilon \wedge \epsilon'_2).$ As $e_1 \sqsubseteq e_2,$ then by Lemma 6.29, $\text{iref}(e_1) \sqsubseteq \text{iref}(e_2).$

Then using $\subseteq_{\text{prot}(1)}$ we can conclude that $\Omega + t_2^U_1 \sqsubseteq t_2^U_2.$ As $\Omega' = \Omega, \mu_1 = \mu'_1$ and $\mu_2 = \mu'_2$ then also $\Omega' + \mu'_1 \sqsubseteq \mu'_2.$

\begin{proposition}[Dynamic Guarantee]
Suppose $t_1^U \subseteq t_2^U,$ $\phi_1 \sqsubseteq \phi_2,$ and $\mu_1 \sqsubseteq \mu_2.$ If $t_1^U \sqsubseteq \epsilon_11 \sqsubseteq \epsilon_12,$ $t_2^U \sqsubseteq \epsilon_21 \sqsubseteq \epsilon_22,$ and $\mu_1 \sqsubseteq \mu_2$.

Proof. We prove the following property instead: Suppose $\Omega + t_1^U \sqsubseteq t_1^U, \phi_1 \sqsubseteq \phi_2,$ and $\mu_1 \sqsubseteq \mu_2.$ If $t_1^U \sqsubseteq \phi_1 \rightarrow t_2^U \sqsubseteq \phi_2 \rightarrow t_2^U$ where $t_1^U \subseteq t_2^U$ and $\mu_1 \subseteq \mu_2.$

By induction on the structure of a derivation of $t_1^U \subseteq t_2^U.$ For simplicity we omit the $\Omega$ notation on precision relations when it is not relevant for the argument.

Case (R$\rightarrow$). $\Omega + t_1^U \subseteq t_2^U, \Omega + \mu_1 \sqsubseteq \mu_2$ and

$t_1^U \sqsubseteq \phi_1 \rightarrow t_2^U \sqsubseteq \phi_2 \rightarrow t_2^U \sqsubseteq \phi_2 \rightarrow t_2^U \sqsubseteq \phi_2 \rightarrow t_2^U, \Omega' + \mu'_1 \sqsubseteq \mu'_2$ for some $\Omega' \supseteq \Omega.$ And the result holds immediately.

Case (Rf). $t_1^U = f_1[t_1^U] \subseteq t_2^U = f_1[t_1^U].$ We know that $\Omega + \phi_1[t_1^U] \subseteq f_2[t_1^U].$ By using Prop 6.18, $U_1 \subseteq U_2.$ By Prop 6.22, we also know that $\Omega + t_1^U \subseteq t_1^U.$ By induction hypothesis, $t_1^U \sqsubseteq t_1^U \sqsubseteq t_1^U \sqsubseteq t_1^U, \Omega' + \mu'_1 \sqsubseteq \mu'_2$ for some $\Omega' \supseteq \Omega.$

Then by Prop 6.21 then $\Omega' \sqsubseteq f_1[t_2^U] \subseteq f_2[t_2^U]$ and the result holds.

Case (Rprot). Then $t_1^U \sqsubseteq \text{prot}_{\epsilon_12}[\phi_1(\epsilon_11) \rightarrow t_2^U \sqsubseteq \text{prot}_{\epsilon_12}[\phi_1(\epsilon_11) \rightarrow t_2^U \sqsubseteq \text{prot}_{\epsilon_12}[\phi_1(\epsilon_11) \rightarrow t_2^U,$ $\Omega' + \mu'_1 \sqsubseteq \mu'_2$ for some $\Omega' \supseteq \Omega.$

But then by $(\subseteq_{\text{prot}(1)},)$

$\Omega' + \text{prot}_{\epsilon_12}[\phi_1(\epsilon_11) \rightarrow t_2^U \sqsubseteq \text{prot}_{\epsilon_12}[\phi_1(\epsilon_11) \rightarrow t_2^U.$

Case (Rg). $t_1^U = g_1[et_1], t_1^U = g_2[et_2], \Omega + g_1[et_1] \subseteq g_2[et_2].$ Also $et_1 \rightarrow_{e} et'_1$ and $et_2 \rightarrow_{e} et'_2.$
Then there exists $U_1, \varepsilon_1, \varepsilon_12$ and $v_1$ such that $et_1 = \varepsilon_11(\varepsilon_12v_1 :: U_1)$. Also there exists $U_2, \varepsilon_21, \varepsilon_22$ and $v_2$ such that $et_2 = \varepsilon_21(\varepsilon_22v_2 :: U_2)$. By Prop 6.20, $\varepsilon_11 \sqsubseteq \varepsilon_21$, and by $(\sqsubseteq ::) \varepsilon_12 \sqsubseteq \varepsilon_22$, $v_1 \sqsubseteq v_2$ and $U_1 \sqsubseteq U_2$. Then as $et_1 \rightarrow c (\varepsilon_12 \circ \sqsubseteq: \varepsilon_11)v_1$ and $et_2 \rightarrow c (\varepsilon_22 \circ \sqsubseteq: \varepsilon_21)v_2$ then, by Prop 6.24 we know that $\varepsilon_12 \circ \sqsubseteq \varepsilon_11 \sqsubseteq \varepsilon_22 \circ \sqsubseteq \varepsilon_21$. Then using this information, and the fact that $v_1 \sqsubseteq v_2$, by Prop 6.19, it follows that $\Omega \vdash \mu_1[et_1'] \sqsubseteq g_1[et_2']$. As $\Omega' = \Omega, \mu_1' = \mu_1$ and $\mu_2 = \mu_2'$ then $\Omega' \vdash \mu_1' \sqsubseteq \mu_2'$.

Case (Rpro). Analogous to (Rprot) case but using $\rightarrow c$ instead.

□
6.6 Noninterference

In this section we present the proof of noninterference for GSL_{Ref}. We use a logical relation that is more general than the one presented in the paper. The main difference (beside using intrinsic terms), is that the logical relation is no longer indexed by a static security effect. As $\phi$ embeds the static security effect information, we generalize the logical relation to also relate two different static security effects as well. Section 6.6.1 present some auxiliary definitions. Section 6.6.2 presents the proof of Noninterference (Prop 6.65), which implies Security Type Soundness (Prop 2.24) presented in the paper.

6.6.1 Definitions. We introduce a function $uval$, which strips away ascriptions from a simple value:

$$uval : \text{GType} \to \text{SimpleValue}$$

$$uval(u) = u$$

$$uval(\ell u :: U) = u.$$ 

In order to compare the observable results of program, we introduce the $rval(\nu)$ operator, which strips away any checking-related information like labels or evidence-carrying ascriptions:

$$rval : \text{Value} \to \text{RawValue}$$

$$rval(b_g) = b$$

$$rval(\epsilon b_g :: U) = b$$

$$rval(\epsilon \text{unit}_g) = \text{unit}$$

$$rval(\epsilon \text{unit}_g :: U) = \text{unit}$$

$$rval(\epsilon o^U_g) = o$$

$$rval(\epsilon o^U_g') :: U) = o$$

$$rval((\lambda^x U :: t^U_1)_g) = (\lambda^x U :: t^U_1)$$

$$rval((\lambda^x U :: t^U_2)_g :: U) = (\lambda^x U :: t^U_2)$$

**Definition 6.32 (Gradual security logical relations).** For an arbitrary element $\ell_o$ of the security lattice, the $\ell_o$-level gradual security relations are step-indexed and type-indexed binary relations on tuples of security effect, closed terms and stores defined inductively as presented in Figure 30. The notation $\langle \phi_1, \nu_1, \mu_1 \rangle \approx_{\ell_o}^k \langle \phi_2, \nu_2, \mu_2 \rangle : U$ indicates that the tuple of security effect $\phi_1$, value $\nu_1$ and store $\mu_1$ is related to the tuple of security effect $\phi_2$, value $\nu_2$ and store $\mu_2$ at type $U$ for $k$ steps when observed at the security level $\ell_o$. Similarly, the notation $\langle \phi_{\ell o}^t, t_{\ell o}, \mu_{\ell o} \rangle \approx_{\ell_o}^k \langle \phi_{\ell o}^t, t, \mu \rangle C(U)$ indicates that the tuple of security effect $\phi_1$, term $t_1$ and store $\mu_1$, and the tuple of security effect $\phi_2$, term $t_2$ and store $\mu_2$ are related computations for $k$ steps, that produce related values and related stores at type $U$ when observed at the security level $\ell_o$. Notation $\mu_1 \approx_{\ell_o}^k \mu_2$ relates stores $\mu_1$ and $\mu_2$ for $k$ steps when observed at security level $\ell_o$. Finally, notation $\phi_1 \approx_{\ell_o} \phi_2$, relates security effect $\phi_1$ and $\phi_2$ for any number of steps at security level $\ell_o$.

We say that a value is observable at level $\ell_o$ if, given a security effect $\phi$, the value is typeable, the security effect is observable, and the label of the value is sublabel of $\ell_o$. Also, as value $\nu$ can be a casted value, we need to analyze if its underlying evidence justifies that the security level of the bare value is also subsumed by the observer security level. We do this by demanding that the underlying evidence and label is also observable. We say that a security effect is observable if its underlying evidence and static label is also observable. We say that an evidence and label
\[ \langle \phi_1, v_1, \mu_1 \rangle \approx_{k, \ell_o}^k \langle \phi_2, v_2, \mu_2 \rangle : U \iff \phi_1 \approx_{\ell_o} \phi_2 \land \mu_1 \approx_{k, \ell_o} \mu_2 \land \phi_1 \triangleright v_1 \in \mathbb{T}[U] \land \text{obsEq}_{\ell_o}(\phi_1 \triangleright v_1, \phi_2 \triangleright v_2) \land (\text{obs}_{\ell_o}(\phi_1 \triangleright v_1) \implies \text{obsRel}_{U, k, \ell_o}^U(\phi_1, v_1, \mu_1, \phi_2, v_2, \mu_2)) \]

\[ \text{obsRel}_{U, k, \ell_o}^U(\phi_1, v_1, \mu_1, \phi_2, v_2, \mu_2) \iff (rval(v_1) = rval(v_2)) \quad \text{if} \ U \in \{ \text{Bool}_g, \text{Unit}_g, \text{Ref}_g \} \]

\[ \text{obsRel}_{U_1, \ell_o}^U(\phi_1, v_1, \mu_1, \phi_2, v_2, \mu_2) \iff \forall j \leq k. \forall U' = U'' \triangleright_{\ell_1}^g \phi_1 \triangleright_{\ell_2}^g U_1', \phi_2 \triangleright_{\ell_2}^g \phi_2', \phi_1 \approx_{\ell_o} \phi_2' \text{ s.t. } \phi_1 \leq_{\ell_o} \phi_1', \epsilon_1 \vdash_{\ell_1} U_1, \epsilon_2 \vdash_{\ell_2} U_2 \leq U', \text{ and } \epsilon_2 \vdash_{\ell_2} U'_1 \leq U''_1, \epsilon_2 \vdash_{\ell_2} \phi_2 \triangleright_{\ell_2}^g \phi_2', \text{ we have:} \]

\[ \forall U'_1, \epsilon_1', \langle \phi_1', v_2, \mu_1' \rangle \approx_{\ell_o} \langle \phi_2, v_2, \mu_2' \rangle : U'_1, \text{dom}(\mu_1) \subseteq \text{dom}(\mu_2), \]

\[ \langle \phi_1, (\epsilon_1' v_1 \triangleright_{\ell_1}^U \epsilon_2' v_2'), \mu_1' \rangle \approx_{\ell_o} \langle \phi_2, (\epsilon_1' v_2 \triangleright_{\ell_2}^U \epsilon_2' v_2'), \mu_2' \rangle : C(U''_2 \triangleright g_2) \]

\[ \langle \phi_1, t_1, \mu_1 \rangle \approx_{k, \ell_o}^k \langle \phi_2, t_2, \mu_2 \rangle : C(U) \iff \phi_1 \approx_{\ell_o} \phi_2 \land \mu_1 \approx_{k, \ell_o} \mu_2 \land \forall \phi_1', \phi_2', \phi_3 \approx_{\ell_o} \phi_2' \text{ s.t. } \phi_1 \leq_{\ell_o} \phi_1' \text{ and } \phi_1' \triangleright t_1 \in \mathbb{T}[U] \text{ we have } \forall j < k \]

\[ \left( t_1 \mid \mu_1 \xrightarrow{j} t_1' \mid \mu_1' \implies \mu_1' \approx_{k-j} \mu_2' \land (\text{irred}(t_1') \implies \langle \phi_1, t_1', \mu_1' \rangle \approx_{k-j} \langle \phi_2, t_2', \mu_2' \rangle : U) \right) \]

\[ \mu_1 \approx_{k, \ell_o} \mu_2 \iff \forall \phi_1, \phi_1 \approx_{\ell_o} \phi_2, j < k, \forall \phi_1 \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2) \]

\[ \langle \phi_1 \triangleright_{\ell_2} \phi_2 \triangleright_{\ell_2} \text{ grown } \rangle \quad \text{if defined} \]

\[ \text{obs}_{\ell_o}(\phi_1 \triangleright_{\ell_2} \phi_2 \triangleright_{\ell_2} \text{ grown }) \quad \text{if defined} \]

\[ \text{obs}_{\ell_o}(\epsilon(U)) = \begin{cases} \text{obs}_{\ell_o}(\epsilon(U)) & \epsilon \text{ is defined, where } \epsilon' = \text{ grown}(\epsilon, \ell_o) \\ \text{obs}_{\ell_o}(\epsilon(U)) & \epsilon \text{ is defined, where } \epsilon' = \text{ grown}(\epsilon, \ell_o) \end{cases} \]

\[ \text{ev}(\epsilon u : U) = \epsilon \]

\[ \text{ev}(u) = \text{ grown}(\epsilon, \ell_o) \]

Fig. 30. Gradual security logical relations

are observable, if any value with that underlying evidence and static label, can be used used as argument of a function that expects a value with security level \( \ell_o \). If the consistent transitivity check of the reduction of the application does not hold, then it is not plausible that the security level of
the value is subsumed by $\ell_o$, and therefore is not observable. For instance, consider $\ell_o = L$, evidence $e = ([H, T], [L, T])$ and static label $g = \?$. We can construct any value such as $v = e_{\text{true}} : \text{Bool}$. The level of the value and the bare value are sublabel of $\ell_o$. But the evidence describes that at some point during reduction, the security level of the bare value was required to be at least as high as $H$. Therefore, $v$ is not observable at level $L$ (considering $L \leq H$), because as $\eta_s(\?, \ell_o) = ([L, L], [L, L])$, the consistent transitivity operation $([H, T], [L, T]) \circ < : ([L, L], [L, L])$ does not hold.

Two stores are related at $k$ steps if each value in the heap of the locations they have in common, are related at $j < k$ steps for any related security effects. We say that store $\mu_2$ is the evolution of store $\mu_1$, annotated $\mu_1 \rightarrow \mu_2$ if the domain of $\mu_1$ is a subset of $\mu_2$.

Two tuples of security effects, values and stores are related for $k$ steps at type $\text{Bool}$ if the security effects are related, the stores are related for $k$ steps, the values can be typed as $\text{Bool}$ using the security effects as context (any security effect will do, given that the typing of values do not depend on the security effect). Additionally, both security effect and values must both be either observable or not observable. If the security effect and values are observable then the raw values are the same. Two tuples are observables at type $\text{Unit}$ and $\text{Ref}$ analogous to booleans.

Pairs are related at function types similarly to booleans. The difference is that functions cannot be compared as booleans. Two functions are related if, given two related values and stores for $j < k$ steps at the argument type, the application of those function to the related values are also related for $j$ steps at the return type.

Two tuples of terms and stores are related computations for $k$ steps at type $U$, first, if the security effects are related, and the stores are related for $k$ steps. Second the terms must be typed as $U$ using a observationally higher security effect. Third, if for any $j < k$ both terms can be reduced for at least $j$ steps, then the resulting stores are related for the remaining $k - j$ steps Finally, if after at least $j$ steps the resulting terms are irreducible, then the resulting terms are also related values for the remaining $k - j$ steps at type $U$. Notice that the logical relation also relates programs that do not terminate as long as after $k$ steps the new stores are also related.

To define the fundamental property of the step-indexed logical relations we first define how to relate substitutions:

**Definition 6.33.** Let $\rho$ be a substitution and $\Gamma$ a type substitution. We say that substitution $\rho$ satisfy environment $\Gamma$, written $\rho \models \Gamma$, if and only if $\text{dom}(\rho) = \Gamma$.

**Definition 6.34 (Related substitutions).** Tuples $\langle \phi_1, \rho_1, \mu_1 \rangle$ and $\langle \phi_2, \rho_2, \mu_2 \rangle$ are related on $k$ steps under $\Gamma$, notation $\Gamma \vdash \langle \phi_1, \rho_1, \mu_1 \rangle \approx^k_{\ell_o} \langle \phi_2, \rho_2, \mu_2 \rangle$, if $\rho_1 \models \Gamma$, $\mu_1 \approx^k_{\ell_o} \mu_2$ and

$$\forall x^U \in \Gamma, \langle \phi_1, \rho_1(x^U), \mu_1 \rangle \approx^k_{\ell_o} \langle \phi_2, \rho_2(x^U), \mu_2 \rangle : U$$

6.6.2 Proof of noninterference.

**Lemma 6.35 (Noninterference for booleans).** Suppose $k > 0$, and

- an open term $\phi \triangleright t^U \in T[\text{Bool}_{\ell_o}]$ where $\text{FV}(t) = \{ x^{U_i} \}$ with label($U_i$) $\neq \ell_o$
- two compatible valid stores $t^U \triangleright \mu_1, \mu_1 \approx^k_{\ell_o} \mu_2$

Then for any $j < k$, $v_1, v_2 \in T[U_1]$, if both

- $t^U[v_1/x^{U_1}] \mid \mu_1 \xrightarrow{\phi} j v'_1 \mid \mu'_1$
- $t^U[v_2/x^{U_1}] \mid \mu_2 \xrightarrow{\phi} j v'_2 \mid \mu'_2$

we have that $\text{rval}(v'_1) = \text{rval}(v'_2)$, and $\mu'_1 \approx^k_{\ell_o} \mu'_2$.

**Proof.** The result follows as a special case of Proposition 6.65 below. □
In this theorem, we treat \( t^U \) as a program that takes \( x^U_i \) as its input. Furthermore, the security level \( g' = \text{label}(U_i) \) of the input is not subsumed by the security level \( \ell_o \) of the observer. As such, noninterference dictates that changing non-observable input must not change the observable value of the output (i.e., change true to false or vice-versa). However, this theorem is technically \textit{termination-insensitive} in that it is vacuously true if a change of inputs changes a program that terminates with a value into one that either terminates with an \textbf{error}, or does not terminate at all.

If a program does not terminate after any number of steps, then at least the stores are related at observation level \( \ell_o \).

Note that we compare equality of \textit{raw} values at first-order type. Restricting attention to first-order types (i.e., \text{Bool}) is common when investigating observational equivalence of typed languages. We strip away security information because a person or client who uses the program ultimately observes only the raw value that the program produces.

Also, gradual security \textit{dynamically} traps some information leaks, so a change in equivalent inputs may cause a program that previously yielded a value or diverged to now produce an \textbf{error}. This change in behavior falls under the notion of \textit{termination-insensitive}, since yielding an error is simply a third form of termination behavior (in addition to producing a value and diverging).

Finally, we use notation \( t^S \mid \mu \xrightarrow{ k } t^S \mid \mu' \) to describe that configuration \( t^S \mid \mu \) reduces, in at most \( k \) steps, to configuration \( t^S \mid \mu' \).

**Lemma 6.36.** Consider \( \epsilon_1 \vdash g \triangleleft g' \). If \( \forall \epsilon_2 \text{ such that } \epsilon_2 \vdash g' \triangleleft \ell_o, \epsilon_1 \circ \epsilon_2 \vdash g \triangleleft \ell_o \) is not defined. Then if \( \epsilon_3 \vdash g' \triangleleft g'' \), then \( \forall \epsilon_4 \text{ such that } \epsilon_4 \vdash g'' \triangleleft \ell_o, \epsilon_1 \circ \epsilon_3 \circ \epsilon_4 \vdash g \triangleleft \ell_o \) is not defined.

**Proof.** Applying associativity: \( (\epsilon_1 \circ \epsilon_3) \circ \epsilon_4 = \epsilon_1 \circ (\epsilon_3 \circ \epsilon_4) \), but \( (\epsilon_3 \circ \epsilon_4) \vdash g' \triangleleft g_o \), and we know that \( \epsilon_1 \circ \epsilon_3 \) is not defined \( \forall \epsilon_2 \text{ such that } \epsilon_2 \vdash g' \triangleleft \ell_o \). Therefore \( (\epsilon_1 \circ \epsilon_3) \circ \epsilon_4 \vdash g \triangleleft \ell_o \) is not defined and the result holds.

**Lemma 6.37.** Consider \( \epsilon_1 \vdash g \triangleleft g' \). If \( \forall \epsilon_2 \text{ such that } \epsilon_2 \vdash g' \triangleleft \ell_o, \epsilon_1 \circ \epsilon_2 \vdash g \triangleleft \ell_o \) is not defined. Also \( \epsilon_0 \vdash g_1 \triangleleft g_2 \). If \( \epsilon_3 \vdash g_2 \\vee g' \triangleleft \ell_o \), then \( (\epsilon_0 \vee \epsilon_1) \circ \epsilon_3 \vdash \epsilon_1 \circ \epsilon_2 \vdash g \triangleleft \ell_o \) is not defined.

**Proof.** Let us prove that if \( (\epsilon_0 \vee \epsilon_1) \circ \epsilon_3 \vdash g_1 \\vee g \triangleleft \ell_o \) is defined, then \( (\epsilon_0 \vee \epsilon_1) \circ \epsilon_2 \) is defined.

As join is monotone \( \exists \epsilon'_0 \) such that \( \epsilon'_0 \vdash g' \\triangleleft g_1 \vee g' \).

Suppose \( \epsilon_0 = ([\ell_{11}, \ell_{12}], [\ell_{21}, \ell_{22}]), \epsilon_1 = ([\ell_{31}, \ell_{32}], [\ell_{41}, \ell_{42}]), \epsilon'_0 = ([\ell_{51}, \ell_{52}], [\ell_{61}, \ell_{62}]) \), and \( \epsilon_3 = ([\ell_{71}, \ell_{72}], [\ell_{81}, \ell_{82}]) \).

As \( \epsilon_0 \vee \epsilon_1 = ([\ell_{11} \vee \ell_{31}, \ell_{12} \vee \ell_{32}], [\ell_{21} \vee \ell_{41}, \ell_{22} \vee \ell_{42}]) \) is defined, then \( \ell_{11} \vee \ell_{31} \triangleleft \ell_{12} \vee \ell_{32} \) and \( \ell_{21} \vee \ell_{41} \\triangleleft \ell_{22} \vee \ell_{42} \). Also as

\[
(\epsilon_0 \vee \epsilon_1) \circ \epsilon_3 = ([\ell_{11} \vee \ell_{31}, (\ell_{12} \vee \ell_{32}) \wedge ((\ell_{22} \vee \ell_{42}) \wedge \ell_{72}) \wedge \ell_{82}],
\]

\[
[\ell_{11} \vee \ell_{31} \vee \ell_{21} \vee \ell_{41} \vee \ell_{72} \vee \ell_{81}, \ell_{82}]
\]

is defined then \( \ell_{21} \vee \ell_{41} \vee \ell_{71} \triangleleft (\ell_{22} \vee \ell_{42}) \wedge \ell_{72}, \ell_{11} \vee \ell_{31} \triangleleft (\ell_{22} \vee \ell_{42}) \wedge \ell_{72}, \ell_{11} \vee \ell_{31} \triangleleft \ell_{82}, \) and \( \ell_{21} \vee \ell_{41} \vee \ell_{71} \triangleleft \ell_{82} \).

If we choose \( \epsilon'_0 = \) the interior of the judgment, then we do not get new information, therefore \( [\ell_{21}, \ell_{22}] \subseteq [\ell_{51}, \ell_{52}] \), \textit{i.e.} \( \ell_{51} \triangleleft \ell_{21} \) and \( \ell_{22} \triangleleft \ell_{52} \). Using the same argument \( \ell_{61} \triangleleft \ell_{71} \) and \( \ell_{72} \triangleleft \ell_{62} \).

Then

\[
\epsilon'_0 \circ \epsilon_3
= \Delta([\ell_{51}, \ell_{52}], [\ell_{61}, \ell_{62}] \cap [\ell_{71}, \ell_{72}], [\ell_{81}, \ell_{82}])
= \Delta([\ell_{51}, \ell_{52}], [\ell_{61} \vee \ell_{71}, \ell_{62} \wedge \ell_{72}], [\ell_{81}, \ell_{82}])
= \Delta([\ell_{51}, \ell_{52}], [\ell_{71}, \ell_{72}], [\ell_{81}, \ell_{82}])
\]
which is defined if \( \ell_{51} \leq \ell_{72}, \ell_{71} \leq \ell_{82} \) and \( \ell_{51} \leq \ell_{82} \). But \( \ell_{51} \leq \ell_{21} \leq \ell_{21} \lor \ell_{41} \lor \ell_{71} \leq \ell_{82} \) and \( \ell_{71} \leq \ell_{21} \lor \ell_{41} \lor \ell_{71} \leq \ell_{82} \).

Therefore

\[
e_0 \circ \tau \ = \ (\ell_{51}, \ell_{52} \land \ell_{72} \land \ell_{82}, [\ell_{51} \lor \ell_{71} \lor \ell_{81}, \ell_{82}])
\]

Using the same method, \( e_1 \circ \tau \ = \ (e_0 \circ \tau \circ \tau) \) is defined if \( \ell_{21} \lor \ell_{51} \leq \ell_{22} \land (\ell_{52} \land \ell_{72} \land \ell_{82}), \ell_{11} \leq \ell_{22} \land (\ell_{52} \land \ell_{72} \land \ell_{82}), \) and \( \ell_{11} \leq \ell_{82} \).

But by definition of \( \ell \), \( \ell_{21} \leq \ell_{22}, \) also \( \ell_{21} \leq \ell_{22} \leq \ell_{52}, \) \( \ell_{21} \leq \ell_{21} \lor \ell_{41} \lor \ell_{71} \leq \ell_{22} \land \ell_{42} \land \ell_{72} \leq \ell_{72} \), \( \ell_{21} \leq \ell_{41} \lor \ell_{71} \leq \ell_{82} \), \( \) and \( \ell_{51} \leq \ell_{71} \leq \ell_{72} \), therefore \( \ell_{21} \lor \ell_{51} \leq \ell_{22} \land (\ell_{52} \land \ell_{72} \land \ell_{82}) \).

Also \( \ell_{11} \leq \ell_{22} \leq \ell_{52}, \ell_{11} \leq \ell_{11} \lor \ell_{31} \leq (\ell_{22} \lor \ell_{42}) \land \ell_{72} \leq \ell_{72}, \) and \( \ell_{11} \leq \ell_{11} \lor \ell_{31} \leq \ell_{82} \), therefore \( \ell_{11} \leq \ell_{22} \land (\ell_{52} \land \ell_{72} \land \ell_{82}), \) and \( \ell_{11} \leq \ell_{82} \).

Then as \( e_1 \circ \tau \ (e_0 \circ \tau \circ \tau) \) is defined then if we choose \( e_2 = (e_0 \circ \tau \circ \tau) + \eta \leq \ell_0 \), the result holds.

\[
\square
\]

**Lemma 6.38 (Associativity).** Consider \( e_1, e_2 \) and \( e_3 \), such that \( e_1 \uplus e_2 \leq e_1 \uplus e_3 \) and \( e_3 \uplus e_2 \leq e_3 \uplus e_3 \) and \( e_3 \uplus e_3 \leq e_4 \). \( (e_1 \circ \tau \circ \tau) \circ \tau \circ \tau = e_1 \circ \tau \circ \tau \circ \tau \)

**Proof.** Suppose \( e_1 = ([e_{11}, e_{12}], [e_{21}, e_{22}]), e_2 = ([e_{31}, e_{32}], [e_{41}, e_{42}]), \) and \( e_3 = ([e_{51}, e_{52}], [e_{61}, e_{62}]) \)

Then

\[
(e_1 \circ \tau \circ \tau) \circ \tau \circ \tau = \Delta \circ ([e_{11}, e_{12}], [e_{21}, e_{22}] \lor [e_{31}, e_{32}], [e_{41}, e_{42}]) \circ \tau \circ \tau \circ \tau
\]

where \( \ell_{21}' = e_{12} \land (e_{22} \land e_{32}) \land e_{42} \land e_{52} \land e_{62} \) and \( \ell_{61}' = e_{11} \lor (e_{21} \lor e_{31}) \lor e_{41} \lor e_{51} \lor e_{61} \). But

\[
e_1 \circ \tau \ (e_2 \circ \tau \circ \tau) = e_1 \circ \tau \ \Delta \circ ([e_{31}, e_{32}], [e_{41}, e_{42}] \lor [e_{51}, e_{52}], [e_{61}, e_{62}])
\]

\[
e_1 \circ \tau \ \Delta \circ ([e_{31}, e_{32}], [e_{41}, e_{42}] \lor [e_{51}, e_{52}], [e_{61}, e_{62}])
\]

\[
= \Delta \circ ([e_{11}, e_{12}], [e_{21}, e_{22}] \lor [e_{31}, e_{32}], [e_{41}, e_{42}]) \circ \tau \circ \tau \circ \tau \circ \tau
\]

\[
= \Delta \circ ([e_{11}, e_{12}], [e_{21}, e_{22}] \lor [e_{31}, e_{32}], [e_{41}, e_{42}]) \circ \tau \circ \tau \circ \tau \circ \tau
\]

where \( \ell_{21}' = e_{12} \land (e_{22} \land e_{32}) \land e_{42} \land e_{52} \land e_{62} \) and \( \ell_{61}' = e_{11} \lor (e_{21} \lor e_{31}) \lor e_{41} \lor e_{51} \lor e_{61} \), and the result holds.

\[
\square
\]

**Lemma 6.39.** Consider \( e_1, e_2 \) and \( e_3 \) such that \( e_1 \uplus e_2 \leq e_2 \uplus e_3 \) and \( e_3 \uplus e_3 \leq e_4 \). If \( e_1 \uplus (e_2 \circ \tau \circ \tau) \) is defined, then \( (e_1 \uplus e_2) \circ \tau \circ \tau \circ \tau \) is defined.

**Proof.** By definition of join and consistent transitivity, using the property that the join operator is monotone.
LEMMA 6.40. If \( \bar{\varepsilon}_1 \vdash g_1 \leq g_2 \), then \( \bar{\varepsilon}_2 \vdash g_1 \vee g_1 \leq g_2 \).

Proof. By definition of join and consistent transitivity, using the property that the join operator is monotone.

□

LEMMA 6.41. Consider stores \( \mu_1, \mu_2, \mu'_1, \mu'_2 \) such that \( \mu_1 \rightarrow \mu'_1 \), and substitutions \( \rho_1 \) and \( \rho_2 \), such that \( \Gamma \vdash (\phi_1, \rho_1, \mu_1) \approx^k_{\ell_0} (\phi_2, \rho_2, \mu_2) \), then if \( \forall j \leq k, \mu'_1 \approx^k_{\ell_0} \mu'_2 \) then \( \Gamma \vdash (\phi_1, \rho_1, \mu'_1) \approx^k_{\ell_0} (\phi_2, \rho_2, \mu'_2) \).

Proof. By definition of related computations and related stores. The key argument is that given that \( \mu_1 \rightarrow \mu'_1 \) then \( \mu'_1 \) have at least the same locations of \( \mu_i \) and the values still are related as well given that they still have the same type.

□

LEMMA 6.42 (SUBSTITUTION PRESERVES TYPING). If \( \phi \triangleright t_U \in T[U] \) and \( \rho \models \text{FV}(t_U) \) then \( \phi \triangleright \rho(t_U) \in T[U] \).

Proof. By induction on the derivation of \( \phi \triangleright t_U \in T[U] \)

□

LEMMA 6.43 (REDUCTION PRESERVES RELATIONS). Consider \( \phi_i \leq_{\bar{\ell}_0} \phi_i', \phi_i' \triangleright t_i \in T[U], \mu_i \in \text{STORE}, t_i \triangleright \mu_i \), and \( \mu_1 \approx^k_{\bar{\ell}_0} \mu_2 \). Consider \( j < k \), posing \( t_i \triangleright \mu_i \xrightarrow{\phi_i'} \rho_j \triangleright t \triangleright \mu_i' \), we have
\[
(\phi_1, t_1, \mu_1) \approx^k_{\bar{\ell}_0} (\phi_2, t_2, \mu_2) : C(U) \quad \text{if and only if} \quad (\phi_1, t_1', \mu_1') \approx^{k-j}_{\bar{\ell}_0} (\phi_2, t_2', \mu_2') : C(U)
\]

Proof. Direct by definition of
\[
(\phi_1, t_1, \mu_1) \approx^k_{\bar{\ell}_0} (\phi_2, t_2, \mu_2) : C(U) \quad \text{and transitivity of } \xrightarrow{\phi'} j.
\]

□

LEMMA 6.44 (ASCRPTION PRESERVES RELATION). Suppose \( \varepsilon \vdash U' \leq U \).

(1) If \( \langle \phi_1, v, \mu \rangle \approx^k_{\bar{\ell}_0} \langle \phi_2, v, \mu \rangle \triangleright 2 : U' \) then 
\[
\langle \phi_1, \varepsilon v_1 :: U, \mu_1 \rangle \approx^{k+1}_{\bar{\ell}_0} \langle \phi_2, \varepsilon v_2 :: U, \mu_2 \rangle : C(U).
\]

(2) If \( \langle \phi_1, t, \mu \rangle \approx^k_{\bar{\ell}_0} \langle \phi_2, t, \mu \rangle \triangleright 2 : C(U') \) then 
\[
\langle \phi_1, \varepsilon t_1 :: U, \mu_1 \rangle \approx^{k}_{\bar{\ell}_0} \langle \phi_2, \varepsilon t_2 :: U, \mu_2 \rangle : C(U).
\]

Proof. Following Zdancewic [2002], the proof proceeds by induction on the judgment \( \varepsilon \vdash U' \leq U \). The difference here is that consistent subtyping is justified by evidence, and that the terms have to be ascribed to exploit subtyping. In particular, case 1 above establishes a computation-level relation because each ascribed term \( (\varepsilon v_i :: U) \) may not be a value: each value \( v_i \) is either a bare value \( u_i \) or a casted value \( \varepsilon v_i :: U_i \), with \( \varepsilon_i + S_i \leq U \). In the latter case, \( (\varepsilon(\varepsilon_i u_i :: U_i)) :: U \) either steps to \textbf{error} (in which case the relation is vacuously established), or steps to \( \varepsilon' u_i :: U \), which is a value. Next if both values were originally observables, then whatever the label of \( U \) both values are going to be related. If both values were originally not observables, then by Lemma 6.44 both values are going to be still non observables.

□

LEMMA 6.45. Consider \( \varepsilon_{11} \triangleright U_1 \leq U_2, \varepsilon_{21} \triangleright U_2 \leq U_3 \), and \( \varepsilon_{31} = \varepsilon_{11} \circ \varepsilon_{21} \) such that \( \varepsilon_{31} \vdash U_1 \leq U_3 \). Then if \( \varepsilon_{11} \approx_{\bar{\ell}_0} \varepsilon_{12} \) and \( \varepsilon_{21} \approx_{\bar{\ell}_0} \varepsilon_{22} \), then \( \varepsilon_{31} \approx_{\bar{\ell}_0} \varepsilon_{32} \).

Proof. By induction on \( \varepsilon_{11} \approx_{\bar{\ell}_0} \varepsilon_{12} \).

□
Lemma 6.46. If $\langle \phi_1, v_1, \mu_1 \rangle \approx^k_t \langle \phi_2, v_2, \mu_2 \rangle : U$ and, $\phi_1 \models \text{uval}(v_1) \in T[U_i]$ where $U_i \subseteq U$, then $\forall U', \epsilon_1 \approx_{\epsilon_1} \epsilon_2, \epsilon_1 \circ U \subseteq U', v_1 = \epsilon'_1 U_i :: U, \epsilon_1 = \epsilon' \circ \epsilon_1$, we know that $\langle \phi_1, \epsilon''_1 \text{uval}(v_1) :: U', \mu_1 \rangle \approx^k_t \langle \phi_2, \epsilon''_2 \text{uval}(v_2) :: U', \mu_1 \rangle : U'$.

Proof. The result follows by induction on relation $\langle \phi_1, v_1, \mu_1 \rangle \approx^k_t \langle \phi_2, v_2, \mu_2 \rangle : U$ using Lemmas 6.43, 6.45, and observational monotonicity of the transitivity (Lemma 6.52).

Lemma 6.47 (Downward Closed / Monotonicity). If

(1) $\langle \phi_1, v_1, \mu_1 \rangle \approx^k_{\epsilon_0} \langle \phi_2, v_2, \mu_2 \rangle : U$ then
\[ \forall j \leq k, \langle \phi_1, v_1, \mu_1 \rangle \approx^j_{\epsilon_0} \langle \phi_2, v_2, \mu_2 \rangle : U \]

(2) $\langle \phi_1, t^U_1, \mu_1 \rangle \approx^k_{\epsilon_0} \langle \phi_2, t^U_2, \mu_2 \rangle : C(U)$ then
\[ \forall j \leq k, \langle \phi_1, t^U_1, \mu_1 \rangle \approx^j_{\epsilon_0} \langle \phi_2, t^U_2, \mu_2 \rangle : C(U) \]

(3) $\mu_1 \approx^k_{\epsilon_0} \mu_2$ then $\forall j \leq k, \mu_1 \approx^j_{\epsilon_0} \mu_2$

Proof. By induction on type $U$ and the definition of related stores.

Lemma 6.48. Consider $\epsilon_1 \vdash g'_1 \preceq g_1$ and $\epsilon_2 \vdash g'_2 \preceq g_2$. Then $\neg \text{obs}_{\epsilon_0}(\epsilon_1 g_1) \land \epsilon_1 [\preceq] \epsilon_2 \implies \neg \text{obs}_{\epsilon_0}(\epsilon_2 g_2)$.

Proof. Suppose $\epsilon_1 = \langle [\ell_{11}, \ell_{12}], [\ell_{13}, \ell_{14}] \rangle$ and $\epsilon_2 = \langle [\ell_{21}, \ell_{22}], [\ell_{23}, \ell_{24}] \rangle$.

Also consider $\epsilon'_1 = \delta_{\epsilon_0}(g_1, \ell_o) = \langle [\ell'_{11}, \ell'_{12}], [\ell_o, \ell_o] \rangle$ and $\epsilon'_2 = \delta_{\epsilon_0}(g_2, \ell_o) = \langle [\ell'_{21}, \ell'_{22}], [\ell_o, \ell_o] \rangle$.

If $\epsilon_1 \sqsubseteq \epsilon'_1 = \Delta_{\epsilon_0}(\epsilon_1, \ell_o)$, $\epsilon'_1 = \delta_{\epsilon_0}(\ell_{11} \vee \ell_{12}, [\ell_{11} \cup \ell_{12}, \ell_{13} \cup \ell_{12}, [\ell_o, \ell_o])$ is not defined then

1. $\ell_{13} \cup \ell_{13}' \preceq \ell_{14} \cup \ell_{12}$,
2. $\ell_{13}' \preceq \ell_{14} \cup \ell_{12}$, or
3. $\ell_{13} \cup \ell_{13}' \preceq \ell_o$ or
4. $\ell_{13}' \preceq \ell_o$.

By construction we know that $\ell_{11} \preceq \ell_{14}$. By $\epsilon_1 [\preceq] \epsilon_2$ we know that $\ell_{13} \preceq \ell_{23}$.

If $g_1 = \ell$, then $[\ell'_{11}, \ell'_{12}] = [\ell_{11}, \ell_{14}] = [\ell, \ell]$, therefore $\ell \preceq \ell_{23}$. If $\ell \preceq \ell_o$, then $\ell_{23} \vee \ell_{23}' \preceq \ell_o$ and the result holds immediately. If $\ell \preceq \ell_o$, by construction of evidence we know that it must be the case that $\ell_{11} \preceq \ell_{13}$, then either

1. $\ell \not\preceq \ell_o \land \ell_o \land \ell$ (which is impossible),
2. $\ell_{11} \preceq \ell_o \land \ell$ (which is a contradiction by construction of evidence), or
3. $\ell \not\preceq \ell_o \land \ell_o \preceq \ell_o$ (which contradicts $\ell \preceq \ell_o$) or
4. $\ell_{11} \preceq \ell_o$,

so the only possibility is that $\ell_{11} \preceq \ell_o$, but we know that $\ell_{11} \preceq \ell_{13}$, i.e. $\ell_{11} \preceq \ell$ and that $\ell \preceq \ell_o$, then by transitivity $\ell_{11} \preceq \ell_o$ which is a contradiction so $\ell \preceq \ell_o$ case cannot happen.

If $g_1 = ?$, then $[\ell'_{11}, \ell'_{12}] = [\ell, \ell_o]$. If (1) holds, i.e. $\ell_{11} \preceq \ell_{14} \land \ell_o$, by construction we know that $\ell_{13} \preceq \ell_{14}$, therefore it must be the case that $\ell_{13} \preceq \ell_o$, but $\ell_{13} \preceq \ell_{23}$ and the result holds because (3) does not hold for $\epsilon_2$.

If (2) holds, i.e. $\ell_{11} \preceq \ell_{14} \land \ell_o$, by construction we know that $\ell_{11} \preceq \ell_{14}$, therefore it must be the case that $\ell_{11} \preceq \ell_o$. We also know by construction that $\ell_{11} \preceq \ell_{13}$, then $\ell_{13} \preceq \ell_{23}$. As $\ell_{13} \preceq \ell_{23}$, then $\ell_{23} \preceq \ell_o$ and therefore (3) does not hold for $\epsilon_2$, i.e. $\ell_{23} \not\preceq \ell_{23}' \preceq \ell_o$. If (3) holds, i.e. $\ell_{13} \not\preceq \ell_o$, then $\ell_{13} \preceq \ell_o$, but $\ell_{13} \preceq \ell_{23}$ and the result holds because (3) does not hold for $\epsilon_2$.

If (4) holds, i.e. $\ell_{11} \preceq \ell_o$, as $\ell_{11} \preceq \ell_{13} \preceq \ell_{23}$ then $\ell_{23} \preceq \ell_o$, and therefore (3) does not hold for $\epsilon_2$, i.e. $\ell_{23} \not\preceq \ell_{23}' \preceq \ell_o$. □
Lemma 6.49. Consider $\varepsilon_1 \vdash g_1 \sim g_1$, $\varepsilon_2 \vdash g_2 \sim g_2$, and $\varepsilon_3 = \varepsilon_1 \lor \varepsilon_2$ such that $\varepsilon_3 \vdash g_1 \lor g_2 \sim g_1 \lor g_2$. Then $(\text{obs}_{\varepsilon_1}(\varepsilon_1 g_1)) \land \text{obs}_{\varepsilon_2}(\varepsilon_2 g_2)) \Rightarrow \text{obs}_{\varepsilon_3}(\varepsilon_3 g_2)$.

Proof. Suppose $\varepsilon_1 = ([\ell_{11}, \ell_{12}], [\ell_{13}, \ell_{14}])$ and $\varepsilon_2 = ([\ell_{21}, \ell_{22}], [\ell_{23}, \ell_{24}])$. Then $\varepsilon_1 \lor \varepsilon_2 = \varepsilon_3 = ([\ell_{11} \lor \ell_{21}, \ell_{12} \lor \ell_{22}], [\ell_{13} \lor \ell_{23}, \ell_{14} \lor \ell_{24}])$. Also consider $\varepsilon'_1 = g_{\bowtie}(g_1, \ell_o) = ([\ell_{11}', \ell_{12}], [\ell_{13}, \ell_{14}]), \varepsilon'_2 = g_{\bowtie}(g_2, \ell_o) = ([\ell_{21}', \ell_{22}], [\ell_{23}, \ell_{24}]),$ and $\varepsilon'_3 = g_{\bowtie}(g_2 \bowtie g_3, \ell_o) = ([\ell_{11}', \ell_{12}'], [\ell_{13}, \ell_{14}])$.

If $g_1 = \ell_1$ and $g_2 = \ell_2$, then $\ell_{12}' = \ell_1 \lor \ell_2, \ell_{22}' = \ell_2$ and $\ell_{12}' = \ell_1$. Also $\ell_{11}' = \ell_1 \lor \ell_2, \ell_{21}' = \ell_2$ and $\ell_{11}' = \ell_1$.

If $g_1 \neq \ell_1$ or $g_2 \neq \ell_2$ (the other case is analogous) then $\ell_{32}' = \ell_o$ and $\ell_{12}' = \ell_o$ and $\ell_{22}' = \ell_o$ such that $\ell_2 \bowtie \ell_o$. Also $\ell_{11}' = \bot, \ell_{21}' = \ell_2$, but $\ell_{11}' = \bot$. Therefore $\ell_{32}' = \ell_o \lor \ell_{22}'$ and $\ell_{31}' = \ell_1 \lor \ell_{22}'$.

We know that

1. $\ell_{13} \lor \ell_{11}' \bowtie \ell_{14} \land \ell_{12}'$
2. $\ell_{11} \bowtie \ell_{14} \land \ell_{12}'$, or
3. $\ell_{13} \lor \ell_{11}' \bowtie \ell_o$ or
4. $\ell_{11} \bowtie \ell_o$.
5. $\ell_{23} \lor \ell_{21}' \bowtie \ell_{24} \land \ell_{22}'$
6. $\ell_{21} \bowtie \ell_{24} \land \ell_{22}'$, or
7. $\ell_{23} \lor \ell_{21}' \bowtie \ell_o$ or
8. $\ell_{21} \bowtie \ell_o$.

We have to prove

10. $(\ell_{13} \lor \ell_{23}) \lor \ell_{11}' \bowtie (\ell_{14} \lor \ell_{24}) \land \ell_{32}'$
11. $(\ell_{11} \lor \ell_{21}) \bowtie (\ell_{14} \lor \ell_{24}) \land \ell_{32}'$, or
12. $(\ell_{13} \lor \ell_{23}) \lor \ell_{11}' \bowtie \ell_o$
13. $(\ell_{11} \lor \ell_{21}) \bowtie \ell_o$.

(13) follows directly by (4) and (8).
(12) follows from (3) and (7) and monotonicity of the join.

By definition of evidence and interior, $\ell_{32}' \bowtie \ell_o$ and $\ell_{31}' \bowtie \ell_{32}'$. Therefore, from (1) $\ell_{13} \bowtie \ell_{14}$, from (5) $\ell_{23} \bowtie \ell_{24}$ and therefore $\ell_{13} \lor \ell_{23} \bowtie \ell_{14} \lor \ell_{24}$. Also as $\ell_{13} \bowtie \ell_{12}'$ and $\ell_{23} \bowtie \ell_{12}'$, then $\ell_{13} \lor \ell_{23} \bowtie \ell_{12}' \lor \ell_{22}' = \ell_{32}'$. By similar argument $\ell_{31}' \bowtie (\ell_{14} \lor \ell_{24})$, and also $\ell_{11}' \lor \ell_{21}' \bowtie \ell_{32}'$. But then $\ell_{11}' \bowtie \ell_{11} \lor \ell_{21}' \bowtie \ell_{32}'$ and (10) holds.

□

Lemma 6.50. Consider $\varepsilon_1 \vdash g_1 \bowtie g_2$, $\varepsilon_2 \vdash g_2 \bowtie g_3$, and $\varepsilon_3 = \varepsilon_1 \circ \bowtie \varepsilon_2$ such that $\varepsilon_3 \vdash g_1 \bowtie g_3$. Then $(\text{obs}_{\varepsilon_1}(\varepsilon_1 g_2)) \Rightarrow (\text{obs}_{\varepsilon_2}(\varepsilon_2 g_2) \Rightarrow (\text{obs}_{\varepsilon_3}(\varepsilon_3 g_3))$.

Proof. Suppose $\varepsilon_1 = ([\ell_1, \ell_2], [\ell_3, \ell_4]), \varepsilon_2 = ([\ell_5, \ell_6], [\ell_7, \ell_8])$.

$\varepsilon_1 \circ \bowtie \varepsilon_2 = \Delta^\bowtie([\ell_1, \ell_2], [\ell_3 \lor \ell_5, \ell_4 \land \ell_6], [\ell_7, \ell_8]) = ([\ell_1, \ell_2 \land \ell_4 \land \ell_6 \lor \ell_8], [\ell_1 \lor \ell_3 \lor \ell_5 \lor \ell_7, \ell_8])$.

Notice that as $\ell_3 \leq \ell_1 \lor \ell_3 \lor \ell_5 \lor \ell_7$ then $\varepsilon_1 \bowtie \ell_3$ and as $\ell_7 \leq \ell_1 \lor \ell_3 \lor \ell_5 \lor \ell_7$ then $\varepsilon_2 \bowtie \varepsilon_3$.

What we have to prove is equivalent to prove that

$$(\neg \text{obs}_{\varepsilon_1}(\varepsilon_1 g_2) \lor \neg \text{obs}_{\varepsilon_2}(\varepsilon_2 g_3)) \Rightarrow \neg \text{obs}_{\varepsilon_3}(\varepsilon_3 g_3)$$

If $\neg \text{obs}_{\varepsilon_1}(\varepsilon_1 g_2)$ and as $\varepsilon_1 \bowtie \varepsilon_3$, then by Lemma 6.48 $\neg \text{obs}_{\varepsilon_3}(\varepsilon_3 g_3)$ and the result holds. Similarly, if $\neg \text{obs}_{\varepsilon_2}(\varepsilon_2 g_3)$ and as $\varepsilon_2 \bowtie \varepsilon_3$, then by Lemma 6.48 $\neg \text{obs}_{\varepsilon_3}(\varepsilon_3 g_3)$ and the result holds.

□

Lemma 6.51. Consider $\varepsilon_1 \vdash g_1 \bowtie g_2$, $\varepsilon_2 \vdash g_2 \bowtie g_3$, and $\varepsilon_3 = \varepsilon_1 \circ \bowtie \varepsilon_2$ such that $\varepsilon_3 \vdash g_1 \bowtie g_3$. Then $(\text{obs}_{\varepsilon_1}(\varepsilon_1 g_2) \land \text{obs}_{\varepsilon_2}(\varepsilon_2 g_3)) \Rightarrow \text{obs}_{\varepsilon_3}(\varepsilon_3 g_3))$. 


Proof. Suppose \( \varepsilon_1 = \langle [\ell_1, \ell_2], [\ell_3, \ell_4] \rangle, \varepsilon_2 = \langle [\ell_5, \ell_6], [\ell_7, \ell_8] \rangle \).
\( \varepsilon_1 \circ \varepsilon_2 = \Delta^\prec([\ell_1, \ell_2], [\ell_3 \vee \ell_5, \ell_4 \wedge \ell_6], [\ell_7], [\ell_8]) = \langle [\ell_1, \ell_2 \wedge \ell_4 \wedge \ell_6 \wedge \ell_8], [\ell_1 \wedge \ell_3 \vee \ell_5 \vee \ell_7, \ell_8] \rangle \)

By definition of the transitivity operator, \( \ell_1 \preceq \ell_8 \), \( \ell_1 \preceq \ell_4 \wedge \ell_6 \), and \( \ell_3 \vee \ell_5 \succeq \ell_8 \).
Let us consider \( \varepsilon'_1 = \mathcal{G}_\prec(g_2, \ell_o) = \langle [\ell_1', \ell_2'], [\ell_o, \ell_o] \rangle \), \( \varepsilon'_2 = \mathcal{G}_{\approx}(g_3, \ell_o) = \langle [\ell_5', \ell_6'], [\ell_o, \ell_o] \rangle \) We know that

\begin{enumerate}
  \item \( \ell_3 \vee \ell_1' \preceq \ell_4 \wedge \ell_2' \),
  \item \( \ell_1 \preceq \ell_4 \wedge \ell_2' \), or
  \item \( \ell_3 \vee \ell_1' \preceq \ell_o \) or
  \item \( \ell_1 \preceq \ell_o \).
  \item \( \ell_7 \vee \ell_5' \preceq \ell_8 \wedge \ell_6' \),
  \item \( \ell_5 \preceq \ell_8 \wedge \ell_6' \), or
  \item \( \ell_7 \vee \ell_5' \preceq \ell_o \) or
  \item \( \ell_5 \preceq \ell_o \).
\end{enumerate}

We have to prove

\begin{enumerate}
  \item \( \ell_1 \vee \ell_3 \vee \ell_5 \vee \ell_7 \vee \ell_1' \preceq \ell_8 \wedge \ell_6' \),
  \item \( \ell_1 \preceq \ell_8 \wedge \ell_6' \), or
  \item \( \ell_1 \vee \ell_3 \vee \ell_5 \vee \ell_7 \vee \ell_1' \preceq \ell_o \) or
  \item \( \ell_1 \preceq \ell_o \).
\end{enumerate}

Notice that if \( g_3 = \) \then \( \ell_1' = \ell_o \) and therefore by (4) \( \ell_1 \preceq \ell_o ' \), and by (3), \( \ell_3 \preceq \ell_o ' \). Also \( \ell_o ' = \bot \) and therefore \( \ell_o ' \preceq \ell_7 \) \preceq \ell_8 \). If \( g_2 = \), then \( \ell_o ' \preceq \ell_o ' = \ell_1 \) and \( \ell_7 = \ell_8 = \ell_o \), but we know that \( \ell_1 \preceq \ell_8 \), and therefore \( \ell_1 \preceq \ell_6 ' \) and \( \ell_o ' \preceq \ell_8 \). Also as \( \ell_3 \preceq \ell_8 \) then \( \ell_3 \preceq \ell_6 ' \).

We also know that \( \ell_3 \vee \ell_5 \preceq \ell_8 \) and by definition of intervals \( \ell_7 \preceq \ell_8 \). We know that \( \ell_1 \leq \ell_6 ' \). By (5) \( \ell_7 \vee \ell_5 ' \preceq \ell_o ' \). By (6) \( \ell_5 \preceq \ell_6 ' \). Also \( \ell_3 \preceq \ell_6 ' \) and (10) follows.

We know that \( \ell_1 \leq \ell_8 \) and that \( \ell_1 \leq \ell_o ' \) therefore (11) holds. By (4), (3), (7), (8) and because \( \ell_o ' \preceq \ell_o \) by definition of interior, (12) holds. Finally (13) holds by (4).

\[ \square \]

Lemma 6.52. Consider \( \varepsilon_1 \vdash g_1 \preceq g_2, \varepsilon_2 \vdash g_2 \preceq g_3, \) and \( \varepsilon_3 = \varepsilon_1 \circ \varepsilon_2 \) such that \( \varepsilon_1 \vdash g_1 \preceq g_3. \)
Then \( \neg \mathsf{obs}_{\varepsilon_2}(\varepsilon_1 g_2) \vee \neg \mathsf{obs}_{\varepsilon_3}(\varepsilon_2 g_3) \iff \neg \mathsf{obs}_{\varepsilon_1}(\varepsilon_3 g_3). \)

Proof. Direct by Lemmas 6.50 and 6.51.

\[ \square \]

Lemma 6.53. Consider \( \varepsilon_1 \) and \( \varepsilon'_1 = \varepsilon_2 \triangleright (\varepsilon_1 \circ \varepsilon_3) \), for some \( \varepsilon_2 \) and \( \varepsilon_3 \). Then \( \varepsilon_1 \preceq \varepsilon'_1 \)

Proof. Suppose \( \varepsilon_2 = \langle [\ell_1, \ell_2], [\ell_3, \ell_4] \rangle, \varepsilon_1 = \langle [\ell_5, \ell_6], [\ell_7, \ell_8] \rangle, \) and \( \varepsilon_3 = \langle [\ell_9, \ell_{10}], [\ell_{11}, \ell_{12}] \rangle. \)
\( \varepsilon_1 \circ \varepsilon_3 = \Delta^\prec([\ell_5, \ell_6], [\ell_7 \vee \ell_9, \ell_8 \wedge \ell_{10}], [\ell_{11}, \ell_{12}]) = \langle [\ell_5, \ell_6 \wedge \ell_9 \wedge \ell_{10} \wedge \ell_{11} \wedge \ell_{12}], [\ell_5 \vee \ell_7 \vee \ell_9 \vee \ell_{11} \vee \ell_{12}] \rangle \)
\( \varepsilon'_2 \triangleright (\varepsilon_1 \circ \varepsilon_3) = \langle [\ell_1 \vee \ell_5, \ell_2 \vee (\ell_6 \wedge \ell_9 \wedge \ell_{10} \wedge \ell_{12})], [\ell_3 \vee \ell_5 \vee \ell_7 \vee \ell_9 \vee \ell_{11} \vee \ell_{12}] \rangle \).

But \( \ell_7 \preceq \ell_5 \vee \ell_7 \preceq \ell_9 \vee \ell_{11} \) and therefore, \( \varepsilon_1 \preceq \varepsilon'_1 \).

\[ \square \]

Lemma 6.54. Consider \( \varepsilon_1 \vdash g_1 \preceq g_1 \) and \( \varepsilon'_1 = \varepsilon_2 \triangleright (\varepsilon_1 \circ \varepsilon_3) \) such that \( \varepsilon_1 \vdash g_2 \preceq g_2. \)
Then \( \neg \mathsf{obs}_{\varepsilon_1}(\varepsilon_1 g_1) \Rightarrow \neg \mathsf{obs}_{\varepsilon_1}(\varepsilon'_1 g_2). \)

Proof. By Lemma 6.53 and Lemma 6.48 the result holds immediately.

\[ \square \]
Lemma 6.55. Consider $\varepsilon_1 \vdash g_1' \lessdot g_1, g_2' \lessdot g_2$, and $\varepsilon_3 = \varepsilon_1 \nu \varepsilon_2$ such that $\varepsilon_3 \vdash g_1' \nu g_2' \lessdot g_1 \nu g_2$. Then $\varepsilon_1 \preceq \varepsilon_3$.

Proof. Suppose $\varepsilon_1 = \{[\ell_1, \ell_2], [\ell_3, \ell_4]\}, \varepsilon_2 = \{[\ell_5, \ell_6], [\ell_7, \ell_8]\}$, then $\varepsilon_3 = \{[\ell_1 \nu \ell_5, \ell_2 \nu \ell_6], [\ell_3 \nu \ell_7, \ell_4 \nu \ell_8]\}$. As $\varepsilon_3 \preceq \varepsilon_3 \nu \varepsilon_7$ therefore, $\varepsilon_1 \preceq \varepsilon_3$ and the result holds.

Lemma 6.56. Consider $\varepsilon_1 \vdash g_1' \lessdot g_1, g_2' \lessdot g_2$, and $\varepsilon_3 = \varepsilon_1 \nu \varepsilon_2$ such that $\varepsilon_3 \vdash g_1' \nu g_2' \lessdot g_1 \nu g_2$. Then $(\neg \text{obs}_{\varepsilon_0}(\varepsilon_1 g_1) \land \neg \text{obs}_{\varepsilon_0}(\varepsilon_2 g_2)) \iff \neg \text{obs}_{\varepsilon_0}(\varepsilon_3 g_1 \nu g_2)$.

Proof. First we prove the $\Rightarrow$ direction. By Lemma 6.55, $\varepsilon_1 \preceq \varepsilon_3$. Suppose $\text{obs}_{\varepsilon_0}(\varepsilon_1 g_1)$ does not hold (the other case is analogous). Then by Lemma 6.48 the result holds immediately. Then for the $\Leftarrow$ we use Lemma 6.49 and the result holds immediately.

Lemma 6.57. Consider $\phi' \triangleright i^U \in \mathbb{T}[U]$, and $\mu$, such that $i^U \vdash \mu$ and $\neg \text{obs}_{\varepsilon_0}(\phi')$, and $\forall k > 0$, such that $i^U \vdash \mu \xrightarrow{\phi'} k_{i^U} | \mu'$, then $\forall \phi'$.

1. $\forall o^U \in \text{dom}(\mu) \setminus \text{dom}(\mu)$, $\neg \text{obs}_{\varepsilon_0}(\phi \triangleright o^U)$.
2. $\forall o^U \in \text{dom}(\mu) \cap \text{dom}(\mu) \land \mu'(o^U) \not\equiv \mu(o^U)$, $\neg \text{obs}_{\varepsilon_0}(\phi \triangleright o^U)$.

Proof. We use induction on the derivation of $i^U$. The interest cases are the last step of reduction rules for references and assignments.

Case ($t = \varepsilon_1 o^U \triangleright \varepsilon_2 u$). We are only updating the heap so we only have to prove (a) and (b). Then

$$\varepsilon_1 o^U \triangleright \varepsilon_2 u \xrightarrow{\phi'} \text{unit} \cdot | \mu[o^U \triangleright \phi' \nu \varepsilon_1 u \triangleright (\phi' \nu g \nu g') :: U']$$

where $\varepsilon' = (\varepsilon_2 \circ \leq \text{iref}(\varepsilon_1) \nu (\phi' \nu g \nu g'))$. For simplicity let us call $\varepsilon_2' = (\varepsilon_2 \circ \leq \text{iref}(\varepsilon_1))$ and $\varepsilon_3' = \varepsilon_3 \circ \leq \text{ilbl}(\varepsilon_1)(\phi' \nu g)$). Then by Lemma 6.54, $\neg \text{obs}_{\varepsilon_0}(\varepsilon' \nu \text{label}(U'))$. Next we have to prove that (a) is only not defined. Consider that $\mu(o^U) \equiv \varepsilon u :: U'$. We know that $\text{obs}_{\varepsilon_0}(\phi' \nu g \nu g')$ is not defined, and that $\phi' \varepsilon' [\leq \varepsilon] \varepsilon$, therefore by Lemma 6.48, $\text{obs}_{\varepsilon_0}(\phi' \nu g)$. Then we will show that $\text{obs}_{\varepsilon_0}(\phi' \nu g \nu g')$ is not defined as well and the result holds.

Case ($t = \text{ref}_{\varepsilon_2} \varepsilon_3 u$). We are extending the heap, so we need to only prove (1). Then

$$\text{ref}_{\varepsilon_2} \varepsilon_3 u \cdot | \mu \xrightarrow{\phi'} \phi' \nu \text{ref}_{\varepsilon_2} \varepsilon_3 u \rightarrow \phi' \nu \phi' \nu \phi' \nu g :: U']$$

where $o^U \not\in \text{dom}(\mu)$, $\nu = \varepsilon_3 \nu (\phi' \nu g \nu \nu)$). We need to prove that $\text{obs}_{\varepsilon_0}(\phi' \nu \phi' \nu g :: U')$ does not hold. In order to do so, we will show that $\text{obs}_{\varepsilon_0}(\nu \text{label}(U'))$ does not hold, which follows directly by Lemma 6.54.

Lemma 6.58. Consider $\phi'$, such that $\text{obs}_{\varepsilon_0}(\phi' \nu g)$ does not hold, then $\forall k > 0$, such that $t^U | \mu_i \xrightarrow{\phi'} k t^U | \mu_i'$, then if $\mu_i \nsucceq \varepsilon_0 \mu_2$, then $\mu_1' \succeq^k \varepsilon_0 \mu_2'$

Proof. By Lemma 6.57 we know three things:
(1) $\forall o' \in dom(\mu') \setminus dom(\mu_i)$, $\text{obs}_{\varepsilon_o} (\phi \triangleright \mu'_i (o'U'))$ does not hold, i.e. new locations are not observable.

(2) $\forall o' \in dom(\mu'_i) \cap dom(\mu_i) \land \mu'_i (o'U') \neq \mu (o'U')$,

(a) $\text{obs}_{\varepsilon_o} (\phi \triangleright \mu_i (o'U'))$ does not hold, and

(b) $\text{obs}_{\varepsilon_o} (\phi \triangleright \mu'_i (o'U'))$ does not hold.

i.e. for all updated references they have to be previously not observable, and by definition therefore related, and second they are still non observable after the update, and by definition those locations are still related under $\phi$.

Therefore $\mu'_i \approx_{\varepsilon_o} \mu'_2$ and the result holds. \hfill \Box

**Lemma 6.59.** Consider simple values $u_i \in T[U_i]$ and

$\langle \phi_1, \varepsilon_i u_1 :: U, \mu_1 \rangle \approx_{\varepsilon_o} \langle \phi_2, \varepsilon_i u_2 :: U, \mu_2 \rangle : U$.

If $\varepsilon_i \approx_{\varepsilon_o} \varepsilon_2 : g'$ where $\varepsilon_1 \triangleright g \leq g'$, then

$\langle \phi_1, (\varepsilon_i' \triangleright \varepsilon_1)(u_1 \triangleright g) : U \triangleright g', \mu_1 \rangle \approx_{\varepsilon_o} \langle \phi_2, (\varepsilon_i'' \triangleright \varepsilon_2)(u_2 \triangleright g) : U \triangleright g', \mu_2 \rangle : U \triangleright g'$

**Proof.** By induction on relation $\langle \phi_1, \varepsilon_i u_1 :: U, \mu_1 \rangle \approx_{\varepsilon_o} \langle \phi_2, \varepsilon_i u_2 :: U, \mu_2 \rangle : U$ and Lemma 6.60 (observational-monotonicity of the join), considering that the label stamping can make the new values non observable and that join of evidences does not introduce impresicion. \hfill \Box

**Lemma 6.60.** Suppose that $\phi_1 \leq_{\varepsilon_o} \phi'_1, \phi'_1 \triangleright \text{prot}_{\varepsilon_i \phi'_i}(\varepsilon_i t U_i) \in T[U \triangleright g]$, for $i \in \{1, 2\}$, where $\neg \text{obs}_{\varepsilon_o} (\phi'_1 \triangleright \varepsilon_i \phi'_i, g_i)$, and either $\neg \text{obs}_{\varepsilon_o} (\phi_1 \triangleright \phi_1, g_i)$ or $\neg \text{obs}_{\varepsilon_o} (\varepsilon_i g_i)$. Also consider two stores $\mu$ such that $\mu_1 \approx_{\varepsilon_o} \mu_2$.

Then $\langle \phi_1, \text{prot}_{\varepsilon_i \phi'_i}(\varepsilon_i t U_i), \mu_1 \rangle \approx_{\varepsilon_o} \langle \phi_2, \text{prot}_{\varepsilon_i \phi'_i}(\varepsilon_i t U_i), \mu_2 \rangle$.

**Proof.** Suppose that after at least $j$ more steps, where $j < k$, both subterms reduce to a value (let us assume no cast errors are produced, otherwise the lemma vacuously holds):

$t U_i | \mu_i \triangleright \varepsilon_i u_i | \mu'_i$

Therefore:

$\text{prot}_{\varepsilon_i \phi'_i}(\varepsilon_i t U_i) | \mu'_i$

$\triangleright \varepsilon_i u_i | \mu'_i$

$\triangleright 1 (\varepsilon_i' \triangleright \varepsilon_i')(u_1 \triangleright g_i') : U \triangleright g | \mu'_i$

As the values can be radically different we have to make sure that both values are not observables. If $\text{obs}_{\varepsilon_o} (\phi_i \triangleright \phi_i, g_i)$ does not hold then the values are not observables because the security context is not observable. Let us assume that $\text{obs}_{\varepsilon_o} (\phi_i \triangleright \phi_i, g_i)$ holds, but $\text{obs}_{\varepsilon_o} (\varepsilon_i g_i)$ not. Then by Lemma 6.56, $\text{obs}_{\varepsilon_o} ((\varepsilon_i' \triangleright \varepsilon_i')(\text{label}(U \triangleright g))$ does not hold, and therefore $\text{obs}_{\varepsilon_o} (\phi_i \triangleright (\varepsilon_i' \triangleright \varepsilon_i')(u_1 \triangleright g_i') : U \triangleright g)$ does not hold, and by definition of related evidences $(\varepsilon_i' \triangleright \varepsilon_i') \approx_{\varepsilon_o} (\varepsilon_i'' \triangleright \varepsilon_i'')$.

Now we have to prove that the resulting stores are related. But by Lemma 6.58 the result immediately. \hfill \Box

**Lemma 6.61.** Suppose that $\phi_1 \leq_{\varepsilon_o} \phi'_1, \phi_1 \leq_{\varepsilon_o} \phi''_1, \langle \phi_1, t_1, \mu_1 \rangle \approx_{\varepsilon_o} \langle \phi_2, t_2, \mu_2 \rangle : C(U')$, and that $\phi'_1 \triangleright \text{prot}_{\varepsilon_i \phi'_i}(\varepsilon_i t U_i) \in T[U \triangleright g]$, for $i \in \{1, 2\}$, if $\varepsilon_i \approx_{\varepsilon_o} \varepsilon_2 : U, \phi_1 \triangleright_{\varepsilon_o} \phi_2, \phi'_1 \triangleright_{\varepsilon_o} \phi'_2, \phi''_1 \triangleright_{\varepsilon_o} \phi''_2$, then

$\langle \phi_1, t_1, \mu_1 \rangle \approx_{\varepsilon_o} \langle \phi_2, t_2, \mu_2 \rangle$.
and $\epsilon'_1 \approx_{\ell_o} \epsilon'_2 : g$, 
then $\langle \phi_1, \text{prot}_{\ell_o}^{U' \epsilon_1} (\epsilon_1 t' U'), \mu_1 \rangle \approx^k_{\ell_o} \langle \phi_2, \text{prot}_{\ell_o}^{U' \epsilon_2} (\epsilon_2 t' U'), \mu_2 \rangle : C(U \triangledown g)$

**Proof.** In case that combining evidence may fail, then the Lemma vacuously holds. Let us assume that combining evidence always succeeds. Consider $j < k$, we know by definition of related computations that $$t'_{U'} | \mu_i \xrightarrow{j} t'_{U'} | \mu'_i$$
then $\mu'_1 \approx_{\ell_o} \mu'_2$, and by Lemma 6.62 $\mu_i \rightarrow \mu'_i$. If $t'_{U'}$ are reducible after $k - 1$ steps, then the result holds immediately by (Rprot ($i$)). The interest case if $t'_{U'}$ are irreducible after $j < k$ steps:
Suppose that after $j$ steps $t'_{U'} = v_i$, then $\langle \phi_1, v_1, \mu'_1 \rangle \approx_{k-j} \langle \phi_2, v_2, \mu'_2 \rangle : U'$.
Therefore:
$$\text{prot}_{\ell_o}^{U' \epsilon_1} (\epsilon_1 t' U') | \mu'_i \xrightarrow{j} \text{prot}_{\ell_o}^{U' \epsilon_1} (\epsilon_1' u_1) | \mu'_i$$
$$\text{prot}_{\ell_o}^{U' \epsilon_1} (\epsilon_1' u_1) | \mu'_i \xrightarrow{1} (\epsilon'_1 \triangledown \epsilon'_1)(u_1 \triangledown g_i') :: U \triangledown g | \mu'_i$$

We know by Lemma 6.46 that $\langle \phi_1, \epsilon'_1' u_1 :: U, \mu'_1 \rangle \approx^k_{\ell_o} \langle \phi_2, \epsilon'_2' u_2 :: U, \mu'_2 \rangle : U$.

If $\neg\text{obs}_{\ell_o}(\phi_i \triangleright v_i)$ or $\neg\text{obs}_{\ell_o}(\epsilon_i \triangleright \text{label}(U))$, then by Lemma 6.64, $\text{obs}_{\ell_o}(\phi_i \triangleright \epsilon_i' u_i :: U)$ also does not hold. Finally by Lemma 6.56 $\text{obs}_{\ell_o}(\phi_i \triangleright \epsilon_i'' \triangleright \epsilon_i')(\text{label}(U) \triangledown g)$ does not hold and therefore the final values are related.

Let us consider that $\text{obs}_{\ell_o}(\phi_i \triangleright v_i)$, $\text{obs}_{\ell_o}(\epsilon_i \triangleright \text{label}(U))$, and that $\text{obs}_{\ell_o}(\phi_i \triangleright \epsilon_i' u_i :: U)$ holds (otherwise we follow by the previous argument).

Let us assume that $\neg\text{obs}_{\ell_o}(\epsilon_i')$. Then by Lemma 6.56, $\neg\text{obs}_{\ell_o}(\phi_i \triangleright (\epsilon''_i \triangledown \epsilon'_i)(\text{label}(U) \triangledown g))$, and therefore $\neg\text{obs}_{\ell_o}(\phi_i \triangleright (\epsilon''_i \triangledown \epsilon'_i)(u_i \triangledown g'_i) :: U \triangledown g)$.

If $\text{obs}_{\ell_o}((\epsilon''_i \triangledown \epsilon'_i)(\text{label}(U) \triangledown g))$ hold, then the result follows by Lemma 6.59, and by backward preservation of the relations (Lemma 6.43).

\[\square\]

**Lemma 6.62.** Consider term $\phi \triangleright t' U \in \mathbb{T}[U]$, store $\mu$ and $j > 0$, such that $t' U | \mu \xrightarrow{j} t'_{U'} | \mu'$. Then $\mu \rightarrow \mu'$.

**Proof.** Trivial by induction on the derivation of $t' U$. The only rules that change the store are the ones for reference and assignment, neither of which remove locations. \[\square\]

**Lemma 6.63.** If $\phi \leq_{\ell_o} \phi'$ and $\phi' \leq_{\ell_o} \phi''$, then $\phi \leq_{\ell_o} \phi''$.

**Proof.** Trivial because if $\phi$ is not observable, then $\phi'$ is not observable as well by definition of $\leq_{\ell_o}$, and therefore $\phi''$ must also be not observable. \[\square\]

**Lemma 6.64.** Consider $\phi_1 \triangleright u \in \mathbb{T}[U]$, and $\epsilon \triangleright U \leq U'$. Suppose $\epsilon u :: U' \xrightarrow{1} \epsilon' u :: U'$. If $\neg\text{obs}_{\ell_o}(\phi_i \triangleright v)$ or $\neg\text{obs}_{\ell_o}(\epsilon U')$ if $\neg\text{obs}_{\ell_o}(\phi_i \triangleright \epsilon' u :: U')$.

**Proof.** Direct by Lemma 6.52. \[\square\]

Next, we present the Noninterference proposition, which naturally implies the Security Type Soundness proposition (Prop 2.24) presented in the paper.
PROPOSITION 6.65 (Noninterference). If \( \phi' \vdash \Gamma \in \mathcal{T}[U] \), \( \mu_i \in \text{Store} \), \( \Gamma \vdash \mu_i, \Gamma \vdash (\phi_1, \rho_1, \mu_1) \approx^k_{\ell_0} (\phi_2, \rho_2, \mu_2) \), then \( \langle \phi_1, \rho_1(\hat{i}), \mu_1 \rangle \approx^k_{\ell_0} \langle \phi_2, \rho_2(\hat{i}), \mu_2 \rangle : C(U) \).

Proof. By induction on the derivation of term \( \hat{i} \in \mathcal{T}[U] \). Let us take an arbitrary index \( k \geq 0 \).

Case (x). \( \hat{i} = x^U \) so \( \Gamma = \{x^U\} \). \( \Gamma \vdash (\phi_1, \rho_1, \mu_1) \approx^k_{\ell_0} (\phi_2, \rho_2, \mu_2) \) implies by definition that \( \langle \phi_1, \rho_1(x^U), \mu_1 \rangle \approx^k_{\ell_0} \langle \phi_2, \rho_2(x^U), \mu_2 \rangle : U \), and the result holds immediately.

Case (b). \( \hat{i} = b_g \). By definition of substitution, \( \rho_1(b_g) = \rho_2(b_g) = b_g \). By definition, \( \langle \phi_1, b_g, \mu_1 \rangle \approx^k_{\ell_0} \langle \phi_2, b_g, \mu_2 \rangle : \text{Boo}_g \) as required.

Case (a). \( \hat{i} = o^U_{\mu_i} \) where \( U = \text{Ref}_{g_i} U_1 \). By definition of substitution, \( \rho_1(o^U_{\mu_i}) = \rho_2(o^U_{\mu_i}) = o^U_{\mu'_i} \). We know that \( \phi_1 \vdash o^U_{\mu_i} \in \mathcal{T}[\text{Ref}_{g_i} U_1] \). By definition of related stores, \( \langle \phi_1, o^U_{\mu_i}, \mu_1 \rangle \approx^k_{\ell_0} \langle \phi_2, o^U_{\mu_i}, \mu_2 \rangle : \text{Ref}_{g_i} U_1 \) as required, and the result holds.

Case (λ). \( t^U = (\lambda \sigma^U_{x^U}. t^U_{\mu_i})g \). Then \( U = U_1 \frac{g'}{g} U_2 \).

By definition of substitution, assuming \( x^U_i \notin \text{dom}(\rho_1) \), and Lemma 6.42:

\[ \phi'_1 \vdash \rho_1(t^U) = \phi'_1 \cdot (\lambda \sigma^U_{x^U}. \rho_1(t^U))g \in \mathcal{T}[U] \]

Consider \( j \leq k, \mu'_1, \mu'_2 \) such that \( \mu_i \rightarrow \mu'_i \) and \( \mu'_1 \approx^j_{\ell_0} \mu'_2 \), and assume two values \( v_1 \) and \( v_2 \) such that \( \langle \phi_1, v_1, \mu'_1 \rangle \approx^j_{\ell_0} \langle \phi_2, v_2, \mu'_2 \rangle : U' \). Consider \( U' = U_1' \frac{g'_1}{g'} U_2' \), \( \epsilon_{11}, \epsilon_{12}, \epsilon_{21}, \epsilon_{22} \approx_{\ell_0} \epsilon_{11}, \epsilon_{22} \), such that \( \epsilon_{11} \vdash U_1' \frac{g'_1}{g'} U_2 \leq U' \), that \( \epsilon_{21} \vdash U_1' \leq U''_1 \), and that \( \epsilon_{12} \vdash \phi'_1, g'_1 \cdots g''_{1 \leq j} \).

For simplicity, let us annotate \( U'_2 = U''_2 \frac{g''_2}{g''} \). We need to show that:

\[ \langle \phi_1, \epsilon_{11} (\lambda \sigma^U_{x^U}. \rho_1(t^U))g \rangle_{\epsilon_{12} \epsilon_{21} U_1, \mu'_1} \approx^j_{\ell_0} \langle \phi_2, \epsilon_{22} (\lambda \sigma^U_{x^U}. \rho_2(t^U))g \rangle_{\epsilon_{12} \epsilon_{22} U_2, \mu'_2} : C(U'_2) \]

Each \( v_i \) is either a bare value \( u_0 \) or a casted value \( \epsilon_{ai} u_1 :: U'_i \). In the latter case, the application expression combines evidence, which may fail with \textbf{error}. If it succeeds, we call the combined evidence \( \epsilon''_{21} \). The application rule then applies: it may fail with \textbf{error} if the evidence \( \epsilon''_{21} \) cannot be combined with the evidence for the function parameter. Every time a failure is produced product of evidence combination, then the relation vacuously holds. We therefore consider the only interesting case, where reductions always succeed. Then:

\[ \frac{\epsilon_{11} (\lambda \sigma^U_{x^U}. \rho_1(t^U))g \at \epsilon_{12} \epsilon_{21} U_1, \mu'_1}{\phi'_1} \frac{\text{prot}^\epsilon_{11} \epsilon_{12} \epsilon_{21} U_1, \mu'_1}{\phi''_1} \]

\[ \frac{\epsilon_{22} (\lambda \sigma^U_{x^U}. \rho_2(t^U))g \at \epsilon_{12} \epsilon_{22} U_2, \mu'_2}{\phi''_1} \frac{\text{prot}^\epsilon_{12} \epsilon_{22} U_2, \mu'_2}{\phi''_1} \]

where \( \phi''_1 = (\phi''_1, (\phi''_1 g'_{1 \leq j} g''_{1 \leq j}) \cdot g''_{1 \leq j}) \).

Notice that \( \text{ilat}(\epsilon_{11}) g'_{1 \leq j} \approx_{\ell_0} \text{ilat}(\epsilon_{11}) g''_{1 \leq j} \), also \( \epsilon_{21} g''_{1 \leq j} \approx_{\ell_0} \epsilon_{21} g''_{1 \leq j} \). If \( \epsilon_{ai} \phi''_i \) do not hold, then by Lemma 6.56, \( \epsilon_{ai} \phi''_i \) do not hold. Then \( \phi'_1 \leq_{\ell_0} \phi''_1 \), and by Lemma 6.63, \( \phi'_1 \leq_{\ell_0} \phi''_1 \). Also by Lemmas 6.52 and 6.56, \( \phi''_1 \approx_{\ell_0} \phi''_2 \).

\( \epsilon_{ai}, \epsilon_{ai} \) and \( \epsilon_{ai} \) are the new evidences for the label, return value and argument, respectively. We then extend the substitutions to map \( x^U_i \) to the casted arguments:

\[ \rho'_1 = \rho_1[x^U_i \mapsto \epsilon_{ai} u_1 :: U_1] \]
We know that $\langle \phi_1, v_1, \mu_1 \rangle \approx^j_{\ell_0} \langle \phi_2, v_2, \mu_2 \rangle$ and consider $\rho \cdot u_i \in \mathbb{T}[U_{ai}]$ then $\varepsilon_{ai} \vdash U_{ai} \not\leq U_1$ and $\varepsilon_{ai} = (\varepsilon_{ai} \circ^{<} \varepsilon_{ai} \circ^{<} idom(\varepsilon_1))$. As $\varepsilon_{ai} \approx_{\ell_0} \varepsilon_{ai} \circ^{<} idom(\varepsilon_1)$, therefore using Lemma 6.46 $\langle \phi_1, (\varepsilon_{ai} u_1 :: U_1), \mu_1 \rangle \approx^j_{\ell_0} \langle \phi_2, (\varepsilon_{ai} u_2 :: U_1), \mu_2 \rangle : U_1$

So as $\mu_1 \rightarrow \mu'_1$ then by Lemma 6.41, $\Gamma, x^U_1 \vdash \langle \phi_1, \rho_1', \mu_1 \rangle \approx^j_{\ell_0} \langle \phi_2, \rho_2', \mu_2 \rangle$.

We also know that $\phi''_1 \triangleright \rho_1'(t^{U_2}) \in \mathbb{T}[U_2]$. Then by induction hypothesis:

$$\langle \phi_1, \rho_1'(t^{U_2}), \mu_1' \rangle \approx^j_{\ell_0} \langle \phi_2, \rho_2'(t^{U_2}), \mu_2' \rangle : C(U_2)$$

Finally, as $\varepsilon_{p_i} = icod(\varepsilon_{i1})$, we know that $icod(\varepsilon_{i1}) \approx_{\ell_0} icod(\varepsilon_{i2})$, also $\varepsilon_{i1} = lbll(\varepsilon_{i1})$, we know that $lbll(\varepsilon_{i1}) \approx_{\ell_0} lbll(\varepsilon_{i2})$ then by Lemma 6.61:

$$\langle \phi_1, \rho_1'(t^{U_2}), \mu_1' \rangle \approx^j_{\ell_0} \langle \phi_2, \rho_2'(t^{U_2}), \mu_2' \rangle : C(U_2)$$

and finally the result holds by backward preservation of the relations (Lemma 6.43).

---

**Case (!).** $t^U = \!_{\text{Ref}_g} U_1 \varepsilon \text{Ref}_U$. Then $U = U_1 \vDash g$.

By definition of substitution:

$$\rho_1(t^U) = \!_{\text{Ref}_g} U_1 \varepsilon \text{Ref}_U$$

We have to show that

$$\langle \phi_1, \rho_1(t^{U_1}), \mu_1 \rangle \approx^k_{\ell_0} \langle \phi_2, \rho_2(t^{U_1}), \mu_2 \rangle : C(U_1)$$

By Lemma 6.42:

$$\phi' \triangleright \!_{\text{Ref}_g} U_1 \varepsilon \text{Ref}_U \cdot \rho_1(t^{U_1}) \in \mathbb{T}[U_1 \vDash g]$$

By induction hypotheses on the subterm:

$$\langle \phi_1, \rho_1(t^{U_1}), \mu_1 \rangle \approx^k_{\ell_0} \langle \phi_2, \rho_2(t^{U_1}), \mu_2 \rangle : C(U_1)$$

Consider $j < k$, then by definition of related computations

$$\rho_1(t_i^{U_i}) \mid \mu_i \xrightarrow{\phi_i'} t_i^{U_i} \mid \mu_i' \implies \mu'_i \approx^j_{\ell_0} \mu_2' \wedge (\text{irred}(t_i^{U_i}) \implies \langle \phi_1, t_i^{U_1}, \mu_i \rangle \approx^k_{\ell_0} \langle \phi_2, t_i^{U_1}, \mu_2 \rangle : U_i')$$

Where $U_i' = \text{Ref}_{\phi_i'} U_i^{\prime'}$. If terms $t_i^{U_i'}$ are reducible after $j = k - 1$ steps, then

$$\!_{\text{Ref}_g} U_1 \varepsilon \text{Ref}_U \cdot \rho_1(t_i^{U_i'}) \mid \mu_i \xrightarrow{\phi_i'} t_i^{U_i'} \mid \mu_i' \text{ and the result holds.}$$

If after at most $j$ steps $t_i^{U_i'}$ is irreducible it means that for some $j' \leq j$, $t_i^{\text{Ref}_g U_1 \varepsilon \text{Ref}_U \rho_1(t_i^{U_i'})} \mid \mu_i \xrightarrow{\phi_i'} t_i^{U_i'} \mid \mu_i'$. If $j' = j$ then we use the same same argument for reducible terms and the result holds.

Let us consider now $j' < j$. Then $\langle \phi_1, v_1, \mu_1 \rangle \approx^{k-j}_{\ell_0} \langle \phi_2, v_2, \mu_2 \rangle : U_i'$. By Lemma 6.10, each $v_1$ is either a location $(o_i g_i^{U_i'})$ or a casted location $\varepsilon_1(o_i g_i^{U_i'}) :: U_i'$. Let us assume they both are a casted location (the other cases are analogous). In case a value $v_1$ is a casted value, then the whole term $\rho_1(t^U)$ can take a step by (Rg), combining $\varepsilon$ with $\varepsilon_1$. Such a step either fails, or succeeds with a new combined evidence. Therefore, either:

$$\rho_1(t^U) \mid \mu_i \xrightarrow{\phi_i'} \text{error}$$
in which case we do not care since we only consider termination-insensitive noninterference, or:
\[
\rho_i(t^U) \mid \mu \xrightarrow{\phi_i^j} f^j + 1 \xrightarrow{!\text{Ref}_j \ e_i t^U_{ij} \ | \ \mu_i^j} \rho_i^j \xrightarrow{\phi_i^j} 1 \xrightarrow{\text{pro}^U_{\text{iref}(\epsilon_i) \phi_i^j(\text{iref}(\epsilon_i) \nu_i^j) \mid \mu_i^j}}
\]
with \(\nu_i^j = \mu_i^j(\alpha_{g_i} g_i^j) = \epsilon_e t_i^U \vdash U_i^\nu \), \(\phi_i^\nu = \langle (\phi_i^\nu \epsilon_e \leadsto \text{ilbl}(\epsilon_i)) (\phi_i^\nu \epsilon_e \leadsto g_i^j), \phi_i^\nu \epsilon_e \leadsto g \rangle\). Notice that as \(\langle \phi_1, \nu_1, \mu_1 \rangle \approx_{\epsilon_o}^{k-j} \langle \phi_2, \nu_2, \mu_2 \rangle : U_1^j\) and as \(\epsilon \approx_{\epsilon_o} \epsilon\), then by Lemma 6.46,
\[
\langle \phi_1, \epsilon_1^i \alpha_{g_i} g_i^j, \mu_1 \rangle \approx_{\epsilon_o} \langle \phi_2, \epsilon_2^j \alpha_{g_i} g_i^j, \mu_2 \rangle : \text{Ref}_g U_1 \text{, therefore } \epsilon_1^j \approx_{\epsilon_o} \epsilon_2^j, \text{i.e. } \text{ilbl}(\epsilon_i) \approx_{\epsilon_o} \text{ilbl}(\epsilon_i) \text{ and } \text{iref}(\epsilon_i) \approx_{\epsilon_o} \text{iref}(\epsilon_i).
\]
By Lemma 6.56, if \(\neg \text{obs}_{\epsilon_o}(\phi_i^j)\) then \(\neg \text{obs}_{\epsilon_o}(\phi_i^j)\). Then by Lemma 6.63, \(\phi_i \leq_{\epsilon_o} \phi_i^j\). Also by Lemma 6.56, either \(\text{obs}_{\epsilon_o}(\phi_i^j)\) or \(\neg \text{obs}_{\epsilon_o}(\phi_i^j)\), therefore \(\phi_i^j \approx_{\epsilon_o} \phi_i^j\).

If both locations are not observable because \(\neg \text{obs}_{\epsilon_o}(\phi_i^j)\), then the resulting values also are not related and the result hold immediately. If both locations are related but not observable because \(\neg \text{obs}_{\epsilon_o}(\text{ilbl}(\epsilon_i))\), then by Lemma 6.56 \(\neg \text{obs}_{\epsilon_o}(\phi_i^j)\), and the result holds by Lemma 6.60.

If both locations are observables, then as \(\langle \phi_1, \nu_1, \mu_1 \rangle \approx_{\epsilon_o}^{k-j} \langle \phi_2, \nu_2, \mu_2 \rangle : U_1^j\), by Lemma 6.61,
\[
\langle \phi_1, \text{prot}_{\text{ilbl}(\epsilon_i)}\mu_i^j \cdot \phi_i^j(\text{iref}(\epsilon_i) \nu_i^j), \mu_i \rangle \approx_{\epsilon_o}^j \langle \phi_2, \text{prot}_{\text{ilbl}(\epsilon_i)}\mu_i^j \cdot \phi_i^j(\text{iref}(\epsilon_i) \nu_i^j), \mu_i \rangle : C(U_2^j)
\]
and finally the result holds by backward preservation of the relations (Lemma 6.43).

---

**Case (\(=\)).** \(t^U = \epsilon_1 t_1^U \epsilon_2 t_2^U\). Then \(U = \text{Unit}_L\).

By definition of substitution:
\[
\rho_i(t^U) = \epsilon_1 \rho_i(t_1^U) \epsilon_2 \rho_i(t_2^U)
\]
and Lemma 6.42:
\[
\phi_i \approx_{\epsilon_i} \epsilon_1 \rho_i(t_1^U) \epsilon_2 \rho_i(t_2^U) \in T[\text{Unit}_L]
\]

We have to show that
\[
\langle \phi_1, \epsilon_1 \rho_i(t_1^U) \rangle \approx_{\epsilon_i} \epsilon_2 \rho_i(t_2^U) : C(U_1^j)
\]

By induction hypotheses
\[
\langle \phi_1, \rho_i(t_1^U), \mu_1 \rangle \approx_{\epsilon_o}^{k-j} \langle \phi_2, \rho_2(t_1^U), \mu_2 \rangle : C(U_1^j)
\]

Suppose \(j_1 < k\), and that \(\rho_i(t_1^U)\) are irreducible after \(j_1\) steps (otherwise, similar to case \(!\), the result holds immediately). Then by definition of related computations:
\[
\rho_i(t_1^U) \mid \mu_1 \xrightarrow{\phi_i^j} j_1 \nu_1 \mid \mu_1^j \implies \mu_1^j \approx_{\epsilon_o}^{k-j} \mu_2^j \land \langle \phi_1, \nu_1, \mu_1^j \rangle \approx_{\epsilon_o}^{k-j} \langle \phi_2, \nu_2, \mu_2^j \rangle : U_1
\]

By Lemma 6.62 \(\mu_1 \rightarrow \mu_i^j\), and \(\mu_1^j \approx_{\epsilon_o}^{k-j} \mu_2^j\) then by Lemma 6.41, \(\langle \phi_1, \mu_i^j \rangle \approx_{\epsilon_o}^{k-j} \langle \phi_2, \mu_2^j \rangle\). By induction hypotheses:
\[
\langle \phi_1, \rho_i(t_1^U), \mu_1^j \rangle \approx_{\epsilon_o}^{k-j} \langle \phi_2, \rho_2(t_1^U), \mu_2^j \rangle : C(U_2^j)
\]
Again, consider \( j_2 = k - j_1 \), if after \( j_2 \) steps \( \rho_1(t^{U_2}) \) is reducible or is a value, the result holds immediately. The interest case if after \( j_2' < j_2 \) steps \( \rho_1(t^{U_2}) \) reduces to values \( v'_i \):

\[ \rho_1(t^{U_2}) \mid \mu'_i \xrightarrow{\phi'_i} j'_2 v'_i \mid \mu''_i \implies \mu'_1 \approx_{\ell_o} \mu''_1 \wedge (\phi_1, v'_1, \mu'_1) \approx_{\ell_o} (\phi_2, v'_2, \mu''_2) : U_2 \]

Then

\[ \rho_1(t^{U}) \mid \mu_i \xrightarrow{\phi_i} h + j'_2 \varepsilon_i v_i := \varepsilon \varepsilon_2 v'_i \mid \mu''_i \approx_{\ell_o} \phi_2, \varepsilon_2, \mu'_2 \]

Now \( v_i \) and \( v'_i \) can be bare values or casted values. In the case of casted values we can combine evidence, which may fail with error. We assume that all evidence combinations succeed, otherwise the relation vacuously holds. As both values \( v_i \) are related at some reference type, then by canonical forms (Lemma 6.10) they both must be locations \( o_i^{U_i'} \) for some \( U_i' \leq U_i \).

\[
\begin{align*}
\varepsilon_1 v_1 &:= \varepsilon \varepsilon_2 v'_1 \mid \mu''_i \\
\phi'_1 &\xrightarrow{2} \varepsilon_1 o_i^{U_i'} := \varepsilon \varepsilon_2 U_i' \mid \mu''_i \\
\phi'_1 &\xrightarrow{1} \text{unit}_{\bot} \mid \mu''_i
\end{align*}
\]

Where \( \mu''_i = \mu''_i[o_i^{U_i'} \mapsto \varepsilon_1'(u_i' \triangleright (\phi'_i g \triangleright g)) \supset U_i''] \). Notice that \( \varepsilon_1 \approx_{\ell_o} \varepsilon_1 \) and \( \varepsilon_2 \approx_{\ell_o} \varepsilon_2 \). As \( \langle \phi_1, v'_1, \mu'_1 \rangle \approx_{\ell_o} \langle \phi_2, v'_2, \mu'_2 \rangle : U_2 \) then by Lemma 6.46,

\[
\langle \phi_1, v'_1, \mu'_1 \rangle \approx_{\ell_o} \langle \phi_2, v'_2, \mu'_2 \rangle : U_1 \quad \text{Similarly as} \quad \langle \phi_1, v_1, \mu_1 \rangle \approx_{\ell_o} \langle \phi_2, v_2, \mu_2 \rangle : U_1,
\]

then by Lemma 6.46

\[
\langle \phi_1, (\varepsilon_2')_{o_i^{U_i'}} : \text{Ref}_{\ell_i} U_i', \mu'_1 \rangle \approx_{\ell_o} \langle \phi_2, (\varepsilon_2')_{o_i^{U_i'}} : \text{Ref}_{\ell_i} U_i', \mu'_2 \rangle : U_i'.
\]

We consider first when the values are observable and the locations are identical: As \( \text{iref}(\varepsilon'_1) \approx_{\ell_o} \text{iref}(\varepsilon'_2) \) then either \( \text{obs}_{\ell_o}(\text{iref}(\varepsilon'_1)U'_1) \) or \( \neg \text{obs}_{\ell_o}(\text{iref}(\varepsilon'_1)U'_1) \). Also as \( \phi'_1 \varepsilon \approx_{\ell_o} \phi'_2 \varepsilon \), then either \( \text{obs}_{\ell_o}(\phi'_1) \) or \( \neg \text{obs}_{\ell_o}(\phi'_1) \).

Notice that \( \varepsilon''_1 = (\varepsilon'_2' \circ \varepsilon_1') \triangleright (\phi'_1 g \triangleright g) \), where \( \varepsilon'_1 = ((\phi'_1 \varepsilon \triangleright \text{ilb}(\varepsilon'_1)) \circ \varepsilon_1) \circ \varepsilon_1 \triangleright \text{ilb}(\varepsilon'_1) \). By Lemma 6.46, \( \langle \phi_1, (\varepsilon'_2')_i^{U_i'} : \text{iref}(\varepsilon'_1)u'_1 :: U'_1', \mu'_1 \rangle \approx_{\ell_o} \langle \phi_2, (\varepsilon'_2')_i^{U_i'} : \text{iref}(\varepsilon'_1)u'_2 :: U'_1', \mu'_1 \rangle : U_i' \).

- Suppose \( \text{obs}_{\ell_o}(\phi'_1 \phi'_1 g \triangleright g) \wedge \text{obs}_{\ell_o}(\text{ilb}(\varepsilon'_1)g) \), \( \varepsilon_{s1l} = \phi'_1 \varepsilon_1' \triangleright \text{ilb}(\varepsilon'_1) \), then by Lemma 6.56, \( \text{obs}_{\ell_o}(\phi'_1 \phi'_1 g \triangleright g) \).
  - If \( \text{obs}_{\ell_o}(\phi'_1 \phi'_1 g \triangleright g) \wedge \text{obs}_{\ell_o}(\text{ilb}(\varepsilon'_1)g) \), \( \varepsilon_{s2l} = (\varepsilon_{s1l}) \circ \varepsilon'_1 \) then by Lemma 6.52 \( \text{obs}_{\ell_o}(\phi'_1 \phi'_1 g \triangleright g) \).
- Suppose \( \text{obs}_{\ell_o}(\phi'_1 \phi'_1 g \triangleright g) \wedge \text{obs}_{\ell_o}(\text{ilb}(\varepsilon'_1)g) \), \( \varepsilon_{s1l} = \phi'_1 \varepsilon_1' \triangleright \text{ilb}(\varepsilon'_1) \), then by Lemma 6.56, \( \text{obs}_{\ell_o}(\phi'_1 \phi'_1 g \triangleright g) \).

Also if \( \neg \text{obs}_{\ell_o}(\phi'_1) \Rightarrow \neg \text{obs}_{\ell_o}(\phi'_1) \) and therefore by monotonicity of the join \( \neg \text{obs}_{\ell_o}(\phi'_1) \).

Therefore \( \varepsilon'_1 \approx_{\ell_o} \varepsilon'_2 \), then by Lemma 6.59,
to be not observable as well, independently of the context. Then \( \forall, \phi'' \approx^k_{\ell_\phi} \phi''_2 \),
\[ \approx^k_{\ell_\phi} \langle \phi''_1, \mu''_1 \rangle \\
\approx^k_{\ell_\phi} \langle \phi''_2, \mu''_2 \rangle \\
\langle \phi''_1, \mu''_1 \rangle \approx^k_{\ell_\phi} \langle \phi''_2, \mu''_2 \rangle \]
As every values are related at type Unit, we only have to prove that \( \mu'' \approx^k_{\ell_\phi} \mu''_1 \), but using monotonicity (Lemma 6.47), it is trivial to prove that because both both stores update the same location \( \phi'' \) to values that are related, therefore the result holds.

We consider now the values that are not observable and the locations are not different:
Suppose that \( \mu''_1 \approx^k_{\ell_\phi} \mu''_2 \) such that \( \langle \phi_1, \mu''_1 \rangle \approx^k_{\ell_\phi} \langle \phi_2, \mu''_2 \rangle \).
Then by definition of \( \mu''_1 \) and \( \mu''_2 \), we use the same same argument for reducible terms and the result holds.

---

**Case (ref).** \( t = \text{ref}_{t_1} \). Then \( U = \text{Ref}_U t_1 \).
By definition of substitution:
\[ \rho_1(t_U) = \text{ref}_{t_1} \rho_1(t_U) \]
and Lemma 6.42:
\[ \phi''_1 \approx^k_{\ell_\phi} \text{ref}_{t_1} \phi''_2 \in T[\text{Ref}_U t_1] \]
We have to show that
\[ \langle \phi_1, \text{ref}_{t_1} \rangle \approx^k_{\ell_\phi} \langle \phi_2, \text{ref}_{t_1} \rangle \]
By induction hypotheses:
\[ \langle \phi_1, \rho_1(t_U), \mu \rangle \approx^k_{\ell_\phi} \langle \phi_2, \rho_2(t_U), \mu \rangle \]
Consider \( j < k \), by definition of related computations
\[ \rho_j(t_U) | \mu \xrightarrow{\phi_j} t_j U | \mu \xrightarrow{\phi_j} t_j U | \mu \xrightarrow{\phi_j} t_j U | \mu \xrightarrow{\phi_j} t_j U | \mu \]
If terms \( t_j U \) are reducible after \( j = k - 1 \) steps, then
\[ \text{ref}_{t_1} \rho_j(t_U) | \mu \xrightarrow{\phi_j} \text{ref}_{t_1} \rho_j(t_U) | \mu \]
If after at most \( j \) steps \( t_j U \) is irreducible, it means that for some \( j' \leq j \) \( \text{ref}_{t_1} \rho_j(t_U) | \mu \xrightarrow{\phi_j} \text{ref}_{t_1} \rho_j(t_U) | \mu \]
If \( j' = j \) then we use the same same argument for reducible terms and the result holds.
Let us consider now \( j' < j \). By Lemma 6.10, each \( v_i \) is either a base value \( u_i \) or a casted base value \( e_i u_i :: U_i' \). In case a value \( v_{ij} \) is a casted value, then the whole term \( \rho_i(t^U) \) can take a step by \((Rg)\), combining \( e \) with \( e_i \). Such a step either fails, or succeeds with a new combined evidence. Therefore, either:

\[
\rho_i(t^U) | \mu_i \xrightarrow{\phi_i'} \text{error}
\]

in which case we do not care since we only consider termination-insensitive noninterference, or:

\[
\rho_i(t^U) | \mu_i \xrightarrow{\phi_i'} \rho_i^{e_i} v_i | \mu_i'
\]

By definition of substitution:

\[
\rho_i(t^U) | \mu \xrightarrow{\phi_i'} \rho_i^{e_i} v_i | \mu_i'
\]

with, \( \mu_i' = \mu_i'[o^U_i \mapsto \epsilon_i'(u_{ij} \triangleright \phi_i' g_{ec}) :: U_i] \). Where \( \epsilon_i'' = \epsilon_i' \triangleright (\phi_i' e \triangleleft \epsilon_i) \). Notice that \( \phi_i' e \approx \epsilon_i \), \( \phi_i'' \), and \( \epsilon_i \triangleright \epsilon_i \) therefore by Lemma 6.52. We know that if \( u_i \in T[U_i] \), then \( \epsilon_i + U_i \subseteq U_i. \) Also, as

\[
\langle \phi_1, \epsilon_{i1} u_1 :: U_i, \mu_1' \rangle \approx_{\epsilon_o} \phi_2, \epsilon_{i2} u_2 :: U_i, \mu_2' \rangle : U_i' \]

and as \( \langle \phi_1, \epsilon_{i1} u_1 :: U_i, \mu_1' \rangle \approx_{\epsilon_o} \phi_2, \epsilon_{i2} u_2 :: U_i, \mu_2' \rangle : U_i' \) and as \( \phi_1' e \triangleleft \epsilon_i \) + \( \phi_2' g_{ec} \triangleleft \text{label}(U_1) \), then by Lemma 6.56, Lemma 6.54, and Lemma 6.47.

Also if \( \neg \text{obs}_{\epsilon_o} (\phi_1') \Rightarrow \neg \text{obs}_{\epsilon_o} (\phi_1) \) and therefore by monotonicity of the join \( \neg \text{obs}_{\epsilon_o} (\epsilon_i', \text{label}(U_1)) \). Therefore the values where different but context not observables, now the new values are going to be not observable as well, independently of the context. Then

\[
\forall, \phi_1'' \approx_{\epsilon_o} \phi_2'', \langle \phi_1', \epsilon_{i1}' (u_1 \triangleright \phi_1' g_{ec}) :: U_i, \mu_1'' \rangle \approx_{\epsilon_o} \langle \phi_2', \epsilon_{i2}' (u_2 \triangleright \phi_2' g_{ec}) :: U_i, \mu_2'' \rangle : U_i'.
\]

By definition of related stores \( \mu_1'' \approx_{\epsilon_o} \mu_2'' \). Then by Monotonicity of the relation (Lemma 6.47) \( \mu_1'' \approx_{\epsilon_o} \mu_2'' \) and the result holds.

---

**Case (\( \circ \)).** \( t^U = \epsilon_1 t^{U_1} \oplus^\theta \epsilon_2 t^{U_2} \)

By definition of substitution:

\[
\rho_i(t^U) = \epsilon_1 \rho_i(t^{U_1}) \oplus^\theta \epsilon_2 \rho_i(t^{U_2})
\]

and Lemma 6.42:

\[
\phi_i' \triangleright \epsilon_1 \rho_i(t^{U_1}) \oplus^\theta \epsilon_2 \rho_i(t^{U_2}) \in T[U]
\]

We use a similar argument to case := for reducible terms. The interest case is when we suppose some \( j_1 \) and \( j_2 \) such that \( j_1 + j_2 < k - 3 \) where:

\[
\rho_i(t^{U_1}) | \mu_i \xrightarrow{\phi_i'} j_i v_{i1} | \mu_i' \implies \mu_i' \approx_{\epsilon_o} \mu_2' \wedge \langle \phi_1, \epsilon_{i1} u_1 :: U_i, \mu_1'' \rangle \approx_{\epsilon_o} \langle \phi_2, \epsilon_{i2} u_2 :: U_i, \mu_2'' \rangle : U_i
\]

\[
\rho_i(t^{U_2}) | \mu_i' \xrightarrow{\phi_i'} j_2 v_{i2} | \mu_i'' \implies \mu_i'' \approx_{\epsilon_o} \mu_2'' \wedge \langle \phi_1, \epsilon_{i1} u_2 :: U_i, \mu_1'' \rangle \approx_{\epsilon_o} \langle \phi_2, \epsilon_{i2} u_2 :: U_i, \mu_2'' \rangle : U_2
\]

By Lemma 6.10, each \( v_{ij} \) is either a boolean \((b_{ij})_{g_{ec}}\), or a casted boolean \( \epsilon_{ij} (b_{ij})_{g_{ec}} :: U_j \). In case a value \( v_{ij} \) is a casted value, then the whole term \( \rho_i(t^U) \) can take a step by \((Rg)\), combining \( e_i \) with \( \epsilon_{ij} \). Such a step either fails, or succeeds with a new combined evidence. Therefore, either:

\[
\rho_i(t^U) | \mu_i \xrightarrow{\phi_i'} j_1 + j_2 \text{error}
\]
in which case we do not care since we only consider termination-insensitive noninterference, or:

\[ \vdash \rho_i(t^U) \mid \mu'' \]

\[ \vdash \epsilon_1'(b_{1i})_{g_{1i}'} \vartriangleleft^g \epsilon_2'(b_{12})_{g_{12}'} \mid \mu'' \]

\[ \vdash 1 \]

\[ \epsilon_i'(b_{1i})_{g_{1i}'} : \text{Bool}_g \mid \mu'' \]

with \( b_i = b_{1i} \oplus b_{12} \), \( \epsilon_i' = \epsilon_{i1} \lor \epsilon_{i2} \), and \( g_{i}' = g_{i1}' \lor g_{i2}' \). It remains to show that:

\[ \langle \phi_1, \epsilon_1'(b_{1i})_{g_{1i}'} : \text{Bool}_g, \mu'' \rangle \approx^{k-j-i-3} \langle \phi_2, \epsilon_2'(b_{12})_{g_{12}'} : \text{Bool}_g, \mu'' \rangle : \text{Bool}_g \]

If \( \neg \text{obs}_{\epsilon_o}(\phi_i) \), then the result is trivial because the resulting booleans are also related as they are not observable.

If \( \text{obs}_{\epsilon_o}(\phi_i) \), then by Lemma 6.46, \( \langle \phi_1, \epsilon_1'(b_{1i})_{g_{1i}'} : \text{Bool}_g, \mu'' \rangle \approx_{\epsilon_o} \langle \phi_2, \epsilon_2'(b_{12})_{g_{12}'} : \text{Bool}_g, \mu'' \rangle \). If \( \neg \text{obs}_{\epsilon_o}(\text{ibl}(\epsilon_{i1}))g \) or \( \neg \text{obs}_{\epsilon_o}(\text{ibl}(\epsilon_{i2}))g \), then by Lemma 6.56, \( \neg \text{obs}_{\epsilon_o}(\phi_i)g \) and the result holds. If both \( \text{obs}_{\epsilon_o}(\text{ibl}(\epsilon_{i2}))g \) then \( b_{1i} = b_{21} \) and \( b_{12} = b_{22} \), so \( b_1 = b_2 \), and the result holds.

---

**Case** (app). \( t^U = \epsilon_1 t_{U1} \bowtie_{\epsilon_r} \epsilon_{U2} \epsilon_2 \epsilon_{U2} \)

with \( \epsilon_1 + U_1 \leq S_{i1} \rightarrow \gamma S_{i2} ; \epsilon_2 \rightarrow U_2 \leq U_{i1} \), and \( U = U_{i2} \sim \gamma g \).

We omit the \( \bowtie_{\epsilon_r} \) operator in applications below.

By definition of substitution:

\[ \rho_i(t^U) = \epsilon_1 \rho_i(t^U_1) \epsilon_2 \rho_i(t^U_2) \]

and Lemma 6.42:

\[ \phi_i' \bowtie \epsilon_1 \rho_i(t^U_1) \epsilon_2 \rho_i(t^U_2) \in T[U] \]

We use a similar argument to case := for reducible terms. The interest case is when we suppose some \( j_1 \) and \( j_2 \) such that \( j_1 + j_2 < k \) where by induction hypotheses and the definition of related computations:

\[ \rho_i(t^U_1) \mid \mu_i \xrightarrow{\phi_i'} j_1 v_{11} \mid \mu_i' \implies \mu_i' \approx_{\epsilon_{1o}}^{k-j_1} \mu_1' \land \langle \phi_1, v_{11}, \mu_i' \rangle \approx_{\epsilon_{1o}}^{k-j_1} \langle \phi_2, v_{21}, \mu_2' \rangle : U_1 \]

\[ \rho_i(t^U_2) \mid \mu_i' \xrightarrow{\phi_i'} j_2 v_{12} \mid \mu_i'' \implies \mu_i'' \approx_{\epsilon_{2o}}^{k-j_2} \mu_2'' \land \langle \phi_1, v_{12}, \mu_i'' \rangle \approx_{\epsilon_{2o}}^{k-j_2} \langle \phi_2, v_{22}, \mu_2'' \rangle : U_2 \]

Then

\[ \rho_i(t^U) \mid \mu_i \xrightarrow{\phi_i'} j_1 + j_2 \epsilon_1 v_{11} \epsilon_2 v_{12} \mid \mu_i'' \]

If \( \text{obs}_{\epsilon_o}(\phi_i \bowtie v_{11}) \) then, by definition of \( \approx_{\epsilon_{1o}} \) at values of function type, using \( \epsilon_1 \) and \( \epsilon_2 \) to justify the subtyping relations, we have:

\[ \approx_{\epsilon_{1o}}^{k-j_1-j_2} \langle \phi_1, (\epsilon_1 v_{11} \epsilon_2 v_{12}), \mu_i'' \rangle \]

\[ \approx_{\epsilon_{2o}}^{k-j_1-j_2} \langle \phi_2, (\epsilon_1 v_{21} \epsilon_2 v_{22}), \mu_2'' \rangle : C(U_{12} \sim \gamma g) \]

Finally, by backward preservation of the relations (Lemma 6.43) the result holds.

If \( \neg \text{obs}_{\epsilon_o}(\phi_i \bowtie v_{11}) \), and we assume by canonical forms that \( v_{11} = \epsilon_{11}(\lambda \phi_{1x} t_{1i})_{g_{1i}} : U_1 \) and that \( v_{12} = \epsilon_{12} u_{12} : U_2 \) (and that evidence combination always succeed or the result holds immediately), then,
We use a similar argument to case ε. Finally, by backward preservation of the relations (Lemma 6.43) the result holds.

As ¬obs₁(φ₁ ∗ u₁1), then either ¬obs₁(φ₁) or ¬obs₁(illbl(ε₁1(label(U₁))). If ¬obs₁(φ₁) then ¬obs₁(φ₁′) and by Lemma 6.52, either both ilbl(ε₁1) are observable or not (the latter when ¬obs₁(illbl(ε₁1(label(U₁))))). If ¬obs₁(illbl(ε₁1(label(U₁)))) then similar to the context case, ¬obs₁(φ₁′). Also by Lemma 6.52, ¬obs₁(illbl(ε₁1(label(U₁))))

Finally by Lemma 6.60,

\[ \langle φ₁, \text{prot}_{\text{illbl}(ε₁1)}^{g₁} \text{φ₁}(\text{icod}(ε₁1)τ₁'), μ₁'' \rangle ≈^{k-h-j₁} \langle φ₂, \text{prot}_{\text{illbl}(ε₁1)}^{g₁} \text{φ₂}(\text{icod}(ε₁1)τ₂'), μ₂'' : C(U₁2, g) \]

Finally, by backward preservation of the relations (Lemma 6.43) the result holds.

---

Case (if). \( t^U = \text{if}^g ε₁₁ t^U₁ \) then \( ε₂₁ t^U₂ \) else \( ε₃₁ t^U₃ \), with \( φ₁' ∗ t^U₁ \in T[U₁], g' = \text{label}(U₁), ε₁ₙ₁ = (φ₁, e \text{ ilbl}(ε₁₁)), φ₁'' = (ε₁'₁(ϕ₂'g_c \text{ v } g'), (φ₁'g_c \text{ v } g)), φ₁' ∗ t^U₂ \in T[U₂], φ₁' ∗ t^U₃ \in T[U₃], ε₁₁ U₁ \leq \text{Bool}_g \), and \( U = (U₂ \text{ v } U₃) \text{ v } g \)

By definition of substitution,

\[ ρ₁(t^U) = \text{if}^g ε₁₁ ρ₁(t^U₁) \text{ then } ε₂₁ ρ₁(t^U₂) \text{ else } ε₃₁ ρ₁(t^U₃) \]

We use a similar argument to case := for reducible terms. The interest case is when we suppose some \( j₁ \) and \( j₂ \) such that \( j₁ + j₂ < k \) by where induction hypotheses and related computations we have that:

\[ ρ₁(t^U₁) | μ₁ \xrightarrow{φ₁'} j₁ v₁₁ | μ₁' \implies μ₁' ≈^{k-j₁} μ₂' ∧ \langle φ₁' ∗ v₁₁, μ₁' ≈^{k-j₁} \langle φ₂' ∗ v₁₂, μ₂' : U₁ \]

By Lemma 6.10, each \( v₁₁ \) is either a boolean \( (b₁₁)₁₁ \) or a casted boolean \( ε₁₁(b₁₁)₁₁ : U₁ \). In either case, \( U₁ \leq \text{Bool}_g \), implies \( U₁ = \text{Bool}_g' \). In case a value \( v₁₁ \) is a casted value, then the whole term \( ρ₁(t^U₁) \) can take a step by (Rg), combining \( ε₁₁ \) with \( ε₁₁ \). Such a step either fails, or succeeds with a new combined evidence. Therefore, either:

\[ ρ₁(t^U) | μ₁ \xrightarrow{φ₁'} j₁+1 \text{ error} \]

in which case we do not care since we only consider termination-insensitive noninterference, or:

\[ ρ₁(t^U) | μ₁ \xrightarrow{φ₁'} j₁+1 \text{ if } ε₁₁'(b₁₁)₁₁, then } ε₂₁ ρ₁(t^U₂) \text{ else } ε₃₁ ρ₁(t^U₃) | μ₁' \]

If ¬obs₁(φ₁ ∗ v₁₁), then by Lemma 6.64 ¬obs₁(φ₁ ∗ b₁₁ : Bool_g). Without loosing generality, let us assume the worst case scenario and that both execution reduce via different branches of the conditional.

Consider \( φ₁'' = ((φ₁' \text{ v } g)_1, (φ₁'g_c \text{ v } g)), (φ₁'g_c \text{ v } g)) \). It is easy to see that if \( φ₁' \) is not observable, then as \( φ₁ \leq ε₁₁ φ₁' \text{ obs}_g(φ₁) \), and therefore by Lemma 6.56, ¬obs₁(φ₁'(φ₁'g_c)). Then \( φ₁ \leq ε₁₁ φ₁'' \).

If ¬obs₁(ε₁₁ Bool_g), then also by Lemma 6.56, ¬obs₁(φ₁'(φ₁'g_c)). Then

\[ ρ₁(t^U) | μ₁ \xrightarrow{φ₁'} j₁+2 \text{ prot}_{\text{illbl}(ε₁₁)}^{g₁} φ₁''(ε₂₁ ρ₁(t^U₁)) | μ₁' \]
\[ \rho_2(t^U) \mid \mu_2 \xrightarrow{\phi_i^j} j_{i+2} \text{prot}_{\text{ilbl}(e_1^j)\text{g}_2}^{g,U} \phi_2''(\varepsilon_3\rho_2(t^{U_1})) \mid \mu_2' \]

But because \( \neg \text{obs}_{\ell_o}(\phi \triangleright e_1^j b_{11} :: \text{Bool}_g) \) then either \( \neg \text{obs}_{\ell_o}(\phi \cdot \varepsilon \cdot \text{g}_c) \) or \( \neg \text{obs}_{\ell_o}(\text{ilbl}(e_1^j g)) \). Then as \( \phi_i \leq_{\ell_o} \phi_i'' \) by Lemma 6.60,

\[ \langle \phi_1, \text{prot}_{\text{ilbl}(e_1^j)\text{g}_1}^{g,U} \phi_1''(\varepsilon_2\rho_1(t^{U_1})), \mu_1' \rangle \approx_{\ell_o} \langle \phi_2, \text{prot}_{\text{ilbl}(e_1^j)\text{g}_2}^{g,U} \phi_2''(\varepsilon_3\rho_2(t^{U_1})), \mu_2' \rangle : \text{C}(U) \]

and the result holds by backward preservation of the relations (Lemma 6.43).

Now consider if \( \text{obs}_{\ell_o}(\phi \triangleright e_1^j b_{11} :: \text{Bool}_g) \) may hold or not. If its not observable we proceed like we just did for the non-observable case. Let us consider that \( \text{obs}_{\ell_o}(\phi \triangleright e_1^j b_{11} :: \text{Bool}_g) \) holds.

Then by definition of \( \approx_{\ell_o} \) on boolean values, \( b_{11} = b_{21} \). Because \( b_{11} = b_{21} \), both \( \rho_1(t^{U_1}) \) and \( \rho_2(t^{U_2}) \) step into the same branch of the conditional. Let us assume the condition is true (the other case is similar):

Then by induction hypotheses \( \langle \phi_1, \rho_1(t^{U_2}), \mu_1' \rangle \approx_{\ell_o} \langle \phi_2, \rho_2(t^{U_2}), \mu_2' \rangle : \text{C}(U_2) \). Also we know that \( \text{ilbl}(e_1^j) \approx_{\ell_o} \text{ilbl}(e_2^j) \), and as \( \phi_i' \approx_{\ell_o} \phi_i'' \), by Lemma 6.56, \( \phi_i' \approx_{\ell_o} \phi_i'' \), then as \( \phi_i \leq_{\ell_o} \phi_i'' \), by Lemma 6.61,

\[ \langle \phi_1, \text{prot}_{\text{ilbl}(e_1^j)\text{g}_1}^{g,U} \phi_1''(\varepsilon_2\rho_1(t^{U_2})), \mu_1' \rangle \approx_{\ell_o} \langle \phi_2, \text{prot}_{\text{ilbl}(e_1^j)\text{g}_2}^{g,U} \phi_2''(\varepsilon_3\rho_2(t^{U_1})), \mu_2' \rangle : \text{C}(U) \]

and the result holds by backward preservation of the relations (Lemma 6.43).

Case \((\text{prot}(\cdot))\). Direct by using Lemma 6.61.

\( \square \)

REFERENCES

Alonzo Church. 1940. A Formulation of the Simple Theory of Types. J. Symbolic Logic 5, 2 (06 1940), 56–68.

