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 stylish 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 GSLRef, a gradual security-typed higher-order language with references. As a gradual language, GSLRef supports the range of static-to-dynamic security checking exclusively driven by type annotations, without resorting to explicit casts. Additionally, GSLRef lets programmers use types to reason statically about termination-insensitive noninterference in all programs, even those that enforce security dynamically. We prove that GSLRef 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 SSL_{Ref}: Static semantics 3
  2.2 SSL_{Ref}: Dynamic semantics 3
  2.3 SSL_{Ref}: Noninterference definitions 4
  2.4 GSL_{Ref}: Static semantics 7
    2.4.1 Additional Definitions 7
  2.5 GSL_{Ref}^\varepsilon: Static semantics 9
  2.6 GSL_{Ref}^\varepsilon: Dynamic semantics 10
  2.7 GSL_{Ref}: Translation to GSL_{Ref} 11
  2.8 Noninterference definitions 13
3 Static Security Typing with References 20
  3.1 SSL_{Ref}: Static type safety 20
  3.2 SSL_{Ref}: 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 GSL_{Ref}^\varepsilon: 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 Intrinsic 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{Ref}ε. 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{Ref}ε. 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

\[
S ::= \text{Bool}_\ell \mid S \rightarrow_\ell S \mid \text{Ref}_\ell S \mid \text{Unit}_\ell \quad \text{(types)}
\]

\[
b ::= \text{true} \mid \text{false} \quad \text{(Booleans)}
\]

\[
r ::= b \mid \lambda x : S.t \mid \text{unit} \mid o \quad \text{(raw values)}
\]

\[
v ::= r \quad \text{(values)}
\]

\[
t ::= v \mid t \mid t \oplus t \mid \text{if } t \text{ then } t \text{ else } t \quad \text{(terms)}
\]

\[
\oplus ::= \land \mid \lor \quad \text{(operations)}
\]

Figure 3 presents the join and meet type functions.

Definition 2.1 (Valid Type Sets).

\[
\begin{align*}
\text{valid}(\{ \text{Bool}_\ell \}) & \quad \text{valid}(\{ S_{i1} \}) & \quad \text{valid}(\{ S_{i2} \}) & \quad \text{valid}(\{ \text{Unit}_\ell \}) \\
\text{valid}(\{ \text{Ref}_\ell S \}) & \quad \text{valid}(\{ S_{i1} \rightarrow_\ell S_{i2} \}) & \quad \text{valid}(\{ \text{Ref}_\ell S \})
\end{align*}
\]

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}.
In this section we present definitions and properties of noninterference for SSL\textsuperscript{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 \( \forall x \in \text{dom}(\Gamma), \forall \ell \in \Gamma[x], \Sigma; \ell_{c} + \rho(x) : S' \), where \( S' \models \Gamma(x) \).

**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 \( \Gamma; \Sigma \vdash \ell_1, \rho_1, \mu_1 \approx_{\ell_1} \ell_2, \rho_2, \mu_2 \), if \( \rho_1 \models \Gamma; \Sigma; \ell_1 \vdash \mu_1 \approx_{\ell_1} k \rho_2, \mu_2 \) and

\[
\forall x \in \Gamma; \Sigma; \ell_1(x), \mu_1 \approx_{\ell_1} k \ell_2(x), \mu_2 : \Gamma(x)
\]
\[ \forall : \text{TYPE} \times \text{TYPE} \rightarrow \text{TYPE} \]

\[ \text{Bool}_\ell \uplus \text{Bool}_{\ell'} = \text{Bool}_{(\ell \uplus \ell')} \]

\[ (S_{11} \rightarrow_{\ell} S_{12}) \lor (S_{21} \rightarrow_{\ell} S_{22}) = (S_{11} \uplus S_{21}) \rightarrow_{(\ell \uplus \ell')} (S_{12} \lor S_{22}) \]

\[ \text{Ref}_\ell \lor \text{Ref}_{\ell'} = \text{Ref}_{(\ell \uplus \ell')} \]

\[ S \lor S \text{ undefined otherwise} \]

\[ \land : \text{TYPE} \times \text{TYPE} \rightarrow \text{TYPE} \]

\[ \text{Bool}_\ell \land \text{Bool}_{\ell'} = \text{Bool}_{(\ell \land \ell')} \]

\[ (S_{11} \rightarrow_{\ell} S_{12}) \land (S_{21} \rightarrow_{\ell} S_{22}) = (S_{11} \land S_{21}) \rightarrow_{(\ell \land \ell')} (S_{12} \land S_{22}) \]

\[ \text{Ref}_\ell \land \text{Ref}_{\ell'} = \text{Ref}_{(\ell \land \ell')} \]

\[ S \land S \text{ undefined otherwise} \]

Fig. 3. SSLRef: Join and meet type functions

\[ \begin{array}{c|c|c}
  t | \mu & \ell \rightarrow & t | \mu \\
\end{array} \]

**Notion of Reduction**

\[ b_{1,\ell_1} \oplus b_{2,\ell_2} | \mu \rightarrow_{\ell} (b_1 \oplus b_2)_{(\ell_1 \lor \ell_2)} | \mu \]

\[ (\ell' x : S.t) | \mu \rightarrow_{\ell} \text{prot}_\ell([v/x]t) | \mu \]

if true\(_\ell\) then \(_{t_1} \rightarrow_{\ell} \text{prot}_\ell(t_1) | \mu \)

if false\(_\ell\) then \(_{t_2} \rightarrow_{\ell} \text{prot}_\ell(t_2) | \mu \)

\[ \text{prot}_\ell(v) | \mu \rightarrow_{\ell} v \lor \ell | \mu \]

\[ \text{ref}^S v | \mu \rightarrow_{\ell} \alpha_o | \mu \rightarrow_{\ell} \alpha_o \rightarrow v \lor \ell \]

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

\[ !o_\ell | \mu \rightarrow_{\ell} v \lor \ell | \mu \]

\[ \lambda \rightarrow_{\ell} v : S | \mu \rightarrow_{\ell} v \lor \text{label}(S) | \mu \]

**Reduction**

\[ \begin{array}{c|c|c|c}
  (\text{R} \rightarrow) & t_1 | \mu_1 \rightarrow_{\ell} t_2 | \mu_2 \\
  t_1 | \mu_1 \rightarrow_{\ell} t_2 | \mu_2 \\
  \end{array} \]

\[ \begin{array}{c|c|c|c}
  (\text{Rf}) & t_1 | \mu_1 \rightarrow_{\ell} t_2 | \mu_2 \\
  f[t_1] | \mu_1 \rightarrow_{\ell} f[t_2] | \mu_2 \\
  \end{array} \]

\[ \begin{array}{c|c|c|c}
  \text{(Rprot)} & t_1 | \mu_1 \rightarrow_{\ell \lor \ell} t_2 | \mu_2 \\
  \text{prot}_\ell(t_1) | \mu_1 \rightarrow_{\ell} \text{prot}_\ell(t_2) | \mu_2 \\
  \end{array} \]

Fig. 4. SSLRef: Label Tracking Dynamic Semantics

**Definition 2.4 (Semantic Security Typing).**

\[ \Gamma; \Sigma; \ell_c \vdash t : S \iff \forall \ell_o \in \text{LABEL}, k \geq 0, \rho_1, \rho_2 \in \text{SUBST} \text{ and } \mu_1, \mu_2 \in \text{STORE} \text{ such that } \Sigma \vdash \mu_1 \text{ and } \Gamma; \Sigma \vdash \langle \ell_c, \rho_1, \mu_1 \rangle \approx_{\ell_o}^{k} \langle \ell_c, \rho_2, \mu_2 \rangle \text{, we have } \Sigma \vdash \langle \ell_c, \rho_1(t), \mu_1 \rangle \approx_{\ell_o}^{k} \langle \ell_c, \rho_2(t), \mu_2 \rangle : C(S) \]

**Proposition 2.5 (Security Type Soundness).** If \(\Gamma; \Sigma; \ell_c \vdash t : S' \)

\[ \forall S, S' :: S, \Gamma; \Sigma; \ell_c \vdash t : S \]
\[
\Sigma \vdash (\ell_1, v_1, \mu_1) \approx^{k}_{\ell_o} (\ell_2, v_2, \mu_2) : S \iff \\
\ell_1 \approx_{\ell_o} \ell_2 \land \Sigma \vdash \mu_1 \approx^{k}_{\ell_o} \mu_2 \land \Sigma; \ell_1 \vdash v_1 : S_1', S'_1 \ll : S,
\]

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

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

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

\[
\Sigma' \vdash (\ell_1, v_1, \mu_1) \approx^{j}_{\ell_o}(\ell_2, v_2, \mu_2) : C(S_2 \rightarrow g)
\]

\[
\Sigma \vdash (\ell_1, t_1, \mu_1) \approx^{k}_{\ell_o}(\ell_2, t_2, \mu_2) : C(S) \iff \\
\ell_1 \approx_{\ell_o} \ell_2 \land \Sigma \vdash \mu_1 \approx^{k}_{\ell_o} \mu_2 \land \Sigma; \ell_1 \vdash t_1 : S_1', S'_1 \ll : S, \forall j < k
\]

\[
\left( t_1 \mid \mu_i \xrightarrow{\ell_1} t_1' \mid \mu_1' \implies \Sigma \subseteq \Sigma', \Sigma' \vdash \mu_1' \approx^{k-j}_{\ell_o} \mu_2' \land \right)
\]

\[
\text{(irred}(t_1') \implies \Sigma' \vdash (\ell_1, t_1', \mu_1') \approx^{k-j}_{\ell_o}(\ell_2, t_2', \mu_2') : S)
\]

\[
\Sigma \vdash \mu_1 \approx^{k}_{\ell_o} \mu_2 \iff \Sigma \vdash \mu_1 \land \forall \ell_1, \ell_1 \approx_{\ell_o} \ell_2, j < k, \forall o \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)
\]

\[
\Sigma \vdash (\ell_1, \mu_1(0), \mu_1) \approx^{j}_{\ell_o}(\ell_2, \mu_2(0), \mu_2) : S(o)
\]

\[
\ell_1 \approx_{\ell_o} \ell_2 \iff \text{obs}_{\ell_o}(\ell_1) \lor \lnot \text{obs}_{\ell_o}(\ell_1)
\]

\[
\mu_1 \rightarrow \mu_2 \iff \text{dom}(\mu_1) \subseteq \text{dom}(\mu_2)
\]

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

\[
\text{obs}_{\ell_o}(\ell) \iff \ell \ll \ell_o
\]

Fig. 5. Security logical relations
Type-Driven Gradual Security with References: Complete Definitions and Proofs

\[ g, g_c, g_r \in \text{GLABEL}, \quad U \in \text{GTYPE}, \quad x \in \text{VAR}, \quad b \in \text{BOOL}, \quad \oplus \in \text{BOOLOP} \]
\[ l \in \text{LOC}, \quad t \in \text{GTERM}, \quad r \in \text{RAWVALUE}, \quad v \in \text{VALUE} \]
\[ \Gamma \in \text{VAR} \xrightarrow{\text{fin}} \text{GTYPE}, \quad \Sigma \in \text{LOC} \xrightarrow{\text{fin}} \text{GTYPE} \]

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

\[ \text{Fig. } 6. \text{ GSLRef: Syntax} \]

\section{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.

\subsection{2.4.1 Additional Definitions.}

\textbf{Definition 2.6 (Type Concretization).} \( \gamma_S : \text{GTYPE} \rightarrow \mathcal{P}(\text{TYPE}) \)
\[ \gamma_S(\text{Bool}_g) = \{ \text{Bool}_\ell | \ell \in \gamma(g) \} \]
\[ \gamma_S(U_1 \overset{g}{\rightarrow} U_2) = \gamma_S(U_1) \overset{\gamma(g)}{\rightarrow} \gamma_S(U_2) \]
\[ \gamma_S(\text{Unit}_g) = \{ \text{Unit}_\ell | \ell \in \gamma(g) \} \]
\[ \gamma_S(\text{Ref}_g U) = \{ \text{Ref}_\ell S | \ell \in \gamma(g), S \in \gamma_S(U) \} \]

Type concretization induces notions of precision and abstraction.

\textbf{Definition 2.7 (Type Precision).} \( U_1 \subseteq U_2 \), if and only if \( \gamma_S(U_1) \subseteq \gamma_S(U_2) \).

\textbf{Definition 2.8 (Type Abstraction).} \( \alpha_S : \mathcal{P}(\text{TYPE}) \rightarrow \text{GTYPE} \)
\[ \alpha_S([\text{Bool}_\ell]) = \text{Bool}_\alpha(\overline{\alpha(\ell)}) \]
\[ \alpha_S([\text{Unit}_\ell]) = \text{Unit}_{\alpha(\overline{\alpha(\ell)})} \]
\[ \alpha_S([\text{Ref}_\ell S]) = \text{Ref}_{\alpha(\overline{\alpha(\ell)})} \alpha_S([S]) \]

\[ \alpha_S(S) \text{ is undefined otherwise} \]

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

\textbf{Definition 10 (Gradual label meet).} \( g_1 \ll g_2 = \alpha([\ell_1 \land \ell_2 | (\ell_1, \ell_2) \in \gamma(g_1) \times \gamma(g_2)]). \)

Algorithmically:
\[ \bot \ll ? = ? \bot = \bot \]
\[ g \ll ? = ? g \ll = ? \text{ if } g \neq \bot \quad \ell_1 \ll \ell_2 = \ell_1 \land \ell_2 \]
\(Ux\) \[x : U \in U; \Gamma; \Sigma; \Gamma \vdash x : U\]
\((Ub)\) \[\Gamma; \Sigma; \Gamma \vdash \text{Bool} : \text{Bool}\]
\((Uo)\) \[\Gamma; \Sigma; \Gamma \vdash U \in \Sigma\]
\((U\text{prot})\) \[\Gamma; \Sigma; g_\text{c} \vdash g + t : U\]
\((U\text{app})\) \[\Gamma; \Sigma; g_\text{c} \vdash t : U_1\]
\((U\vdash)\) \[\Gamma; \Sigma; g_\text{c} \vdash t : U_2\]
\((U\text{ref})\) \[\Gamma; \Sigma; g_\text{c} \vdash t : U'\]
\((U\text{asgn})\) \[\Gamma; \Sigma; g_\text{c} \vdash t_1 : \text{Ref}_U\]

Fig. 7. GSL\text{Ref}: Static Semantics

\[U \triangleright U, U \triangleright U\]

\(\triangleright\) : TYPE \(\times\) TYPE \(\to\) TYPE
\(\text{Bool}_U \triangleright \text{Bool}_U = \text{Bool}_U(\gamma_U g'\triangleright g')\)
\((U_1 \triangleright g_\text{c}(U_2)) \triangleright (U_2 \triangleright g_\text{c}(U_2)) = (U_1 \triangleright \text{g}_\text{c}(U_2)) \triangleright (\gamma_U g'\triangleright (g_\text{c}(U_2)))\)
\(\text{Ref}_U \triangleright \text{Ref}_U \triangleright U' = \text{Ref}_U(\gamma_U g'\triangleright U)\)
\(U\triangleright U\) undefined otherwise

\(\triangleright\) : TYPE \(\times\) TYPE \(\to\) TYPE
\(\text{Bool}_U \triangleright \text{Bool}_U = \text{Bool}_U(\gamma_U g'\triangleright g')\)
\((U_1 \triangleright g_\text{c}(U_2)) \triangleright (U_2 \triangleright g_\text{c}(U_2)) = (U_1 \triangleright \text{g}_\text{c}(U_2)) \triangleright (\gamma_U g'\triangleright (g_\text{c}(U_2)))\)
\(\text{Ref}_U \triangleright \text{Ref}_U \triangleright U' = \text{Ref}_U(\gamma_U g'\triangleright U)\)
\(U\triangleright U\) undefined otherwise

Fig. 8. GSL\text{Ref}: consistent join and consistent meet

**Definition 2.11 (Gradual label join).** \(g_1 \triangleright g_2 = \alpha(\{\ell_1 \lor \ell_2 | (\ell_1, \ell_2) \in g_1 \times g_2\})\)
Algorithmically:
\[T \triangleright \neg \gamma = \neg \gamma \triangleright T = T\]
\[g \triangleright \neg \gamma = \neg \gamma \triangleright g = \gamma \text{ if } g \neq T\]
\[\ell_1 \triangleright \ell_2 = \ell_1 \lor \ell_2\]
Definition 2.12 (Label Meet). \( g_1 \cap g_2 = \alpha(\gamma(g_1) \cap \gamma(g_2)) \).
Algorithmically:
\[
g \cap g \vdash g \quad g \cap ? \vdash ? \cap g = g
\]

Definition 2.13 (Type Meet). \( U_1 \cap U_2 = \alpha_S(\gamma_S(U_1) \cap \gamma_S(U_2)) \).
Algorithmically:
\[
\begin{array}{c}
g \cap g' \\
\text{Bool}_g \cap \text{Bool}_{g'}
\end{array} \quad 
\begin{array}{c}
g \cap g' \\
\text{Unit}_g \cap \text{Unit}_{g'}
\end{array} \quad 
\begin{array}{c}
g \cap g' \\
\text{Ref}_g U_1 \cap \text{Ref}_{g'} U_2
\end{array}
\]
\[
\begin{array}{ccc}
U_1 \cap U_1' & U_2 \cap U_2' & g_1 \cap g_1' \quad g_2 \cap g_2'
\end{array}
\]

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

Definition 2.14 (Type Precision (inductive definition)).
\[
\begin{array}{c}
g_1 \subseteq g_2 \\
\text{Bool}_{g_1} \subseteq \text{Bool}_{g_2}
\end{array} \quad 
\begin{array}{c}
g_1 \subseteq g_2 \\
\text{Unit}_{g_1} \subseteq \text{Unit}_{g_2}
\end{array} \quad 
\begin{array}{c}
U_{11} \subseteq U_{21} \quad U_{12} \subseteq U_{22} \quad g_1 \subseteq g_2 \quad g_{c1} \subseteq g_{c2}
\end{array}
\]
\[
\begin{array}{c}
U_{11} \xrightarrow{g_{c1}} g_1 U_{12} \subseteq U_{21} \xrightarrow{g_{c2}} g_2 U_{22}
\end{array}
\]
\[
\begin{array}{c}
g_1 \subseteq g_2 \\
\text{Ref}_{g_1} U_1 \subseteq \text{Ref}_{g_2} U_2
\end{array}
\]

Definition 2.15 (Consistent label ordering (inductive definition)).
\[
\begin{array}{c}
? \sim g \\
? \sim g
\end{array}
\quad 
\begin{array}{c}
g \sim ? \\
\ell_1 \sim \ell_2
\end{array}
\]

Definition 2.16 (Consistent subtyping (inductive definition)).
\[
\begin{array}{c}
g \sim g' \\
\text{Bool}_g \sim \text{Bool}_{g'}
\end{array} \quad 
\begin{array}{c}
g \sim g' \\
\text{Unit}_g \sim \text{Unit}_{g'}
\end{array} \quad 
\begin{array}{c}
g \sim g' \quad U_1 \subseteq U_2 \quad U_2 \subseteq U_1
\end{array}
\]
\[
\begin{array}{c}
U_1 \xrightarrow{g_2} g_1 U_2 \subseteq U_1 \xrightarrow{g_2'} g_1' U_2'
\end{array}
\]

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

Definition 2.18 (Evidence for labels).
\[
\varepsilon ::= \langle \ell, \ell \rangle
\]
In this section we present the full definition of the dynamic semantics of GSL. We extend the syntax of GSL with frames defined as follows:

Fig. 9. GSLRef: Example typing derivations

Fig. 10. GSLRef: Syntax

Definition 2.19 (Type Evidence). An evidence type is a gradual type labeled with an interval:

\[ E \in \text{GETYPE}, \quad \iota \in \text{LABEL}^2 \]

\[ E \quad ::= \quad \text{Bool}, \quad E \vdash E, \quad \text{Ref}^\iota, \quad E \quad | \quad \text{Unit} \quad (\text{type evidences}) \]

Definition 2.20 (Evidence for types).

\[ \varepsilon \quad ::= \quad \langle E, E \rangle \]

We present the syntax of GSLRef in Figure 10 and the static semantics in Figure 11.

2.6 GSLRef: Dynamic semantics

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

We extend the syntax of GSLRef with frames defined as follows:
Fig. 11. GSL $\mathcal{E}_{Ref}$: Static Semantics

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 GSL$\mathcal{E}_{Ref}$: Translation to GSL$\mathcal{E}_{Ref}$

In this section we present the translation from terms of GSL$\mathcal{E}_{Ref}$ into terms of GSL$\mathcal{E}_{Ref}$ 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:
\[
\begin{align*}
&\text{(r1)} \quad \varepsilon_1(b_1)g_1 \oplus \varepsilon_2(b_2)g_2 | \mu \xrightarrow{\varepsilon g_c} (\varepsilon_1 \lor \varepsilon_2)(b_1 \boxplus b_2)g_1g_2 | \mu \\
&\text{(r2)} \quad \text{prot}\varepsilon_1g_1\varepsilon_2g_2(e_3u) | \mu \xrightarrow{\varepsilon g_c} (\varepsilon_3 \lor \varepsilon_1)(u \lor g_1) | \mu \\
&\text{(r3)} \quad \varepsilon_1(\lambda x : U.t)g @_{\varepsilon_1} e_2u | \mu \xrightarrow{\varepsilon g_c} \begin{cases} \\
\text{prot}_{\text{lbl}(\varepsilon_1)g_1} \varepsilon_2g'((\text{icod}(\varepsilon_1)(\text{ev}(e_2u/x)t)) | \mu & \text{error if } \varepsilon_1' \text{ or } \varepsilon_2' \text{ are not defined} \\
\text{where:} & \\
\varepsilon_1' &= (\varepsilon_1 \lor \text{ilbl}(\varepsilon_1)) \circ \varepsilon_2 \circ \text{idom}(\varepsilon_1) \\
\varepsilon_2' &= \varepsilon_2 \circ \text{idom}(\varepsilon_1) \\
g_1 &= (g_c \lor g) \\
\end{cases} \\
&\text{(r4)} \quad \text{if } \varepsilon_1 b_{g_1} \text{ then } t_2 \text{ else } t_3 | \mu \xrightarrow{\varepsilon g_c} \begin{cases} \\
\text{prot}_{\text{lbl}(\varepsilon_1)g_1} \varepsilon_2g'(e_2t_2) | \mu & \text{if } b = \text{true} \\
\text{prot}_{\text{lbl}(\varepsilon_1)g_1} \varepsilon_2g'(e_2t_3) | \mu & \text{if } b = \text{false} \\
\text{where:} & \\
\varepsilon' &= \varepsilon_1 \lor \text{ilbl}(\varepsilon_1) \\
g' &= g_c \lor g_1 \\
\end{cases} \\
&\text{(r5)} \quad \text{ref}\varepsilon_2\varepsilon_1u | \mu \xrightarrow{\varepsilon g_c} \begin{cases} \\
\text{o} \perp [\mu | o \mapsto \varepsilon'(u \lor g_c)] & \text{error if } \varepsilon \circ \varepsilon_2 \text{ is not defined} \\
\text{where:} & \\
o & \notin \text{dom}(\mu) \\
\varepsilon' &= \varepsilon_1 \lor (\varepsilon \circ \varepsilon_2) \\
\end{cases} \\
&\text{(r6)} \quad \text{!}\varepsilon_1o_{g} | \mu \xrightarrow{\varepsilon g_c} \text{prot}_{\text{lbl}(\varepsilon_1)g} \varepsilon'g'(\text{iref}(\varepsilon_1)\nu) \\
\text{where:} & \\
\mu(\nu) &= \nu \\
\varepsilon' &= \varepsilon_1 \lor \text{ilbl}(\varepsilon_1) \\
g' &= g_c \lor g \\
&\text{(r7)} \quad \varepsilon_1o_{g} :=\varepsilon_3\varepsilon_2u | \mu \xrightarrow{\varepsilon g_c} \begin{cases} \\
\text{unit}_\perp [\mu | o \mapsto \varepsilon'(u \lor (g_c \lor g))] & \text{error if } \varepsilon' \text{ is not defined, or } \varepsilon \leq \text{ilbl}(\varepsilon''') \text{ does not hold} \\
\text{error} & \text{if not defined} \\
\text{where:} & \\
\mu(\nu) &= \varepsilon'''u' \\
\varepsilon' &= (\varepsilon_2 \circ \text{idom}(\varepsilon_1)) \lor ((\varepsilon_1 \text{ilbl}(\varepsilon_1)) \circ \varepsilon_3 \circ \text{idom}(\text{iref}(\varepsilon_1))) \\
\end{cases} \\
\varepsilon_1(\varepsilon_2u) &\to_{<:} (\varepsilon_2 \circ \varepsilon_1)u \\
\text{error} & \text{if not defined} \\
\end{align*}
\]

Fig. 12. GSL\textsuperscript{\text{Ref}}: Dynamic semantics

Definition 2.21 (Operation pattern).

\[
P^T \in \text{GPATTERN}, P^\ell \in \text{LPATTERN} \\
P^T ::=_\mid P^T \text{op}^T P^T \quad \text{(pattern on types)} \\
op^T ::= \text{\lor} \mid \text{\land} \mid \top \quad \text{(operations on types)} \\
P^\ell ::=_\mid P^\ell \text{op}^\ell P^\ell \quad \text{(pattern on labels)} \\
op^\ell ::= \text{\lor} \mid \text{\land} \mid \top \quad \text{(operations on labels)}
\]
The formal definitions of related values and related computations are presented in Figures 20 and 21.

**Definition 2.22 (Related substitutions).** Tuples \( \langle \hat{g}_1, \rho_1, \mu_1 \rangle \) and \( \langle \hat{g}_2, \rho_2, \mu_2 \rangle \) are related on \( k \) steps under \( \Gamma, \Sigma \) and \( g_c \), notation \( \Gamma; \Sigma; g_c \vdash \langle \hat{g}_1, \rho_1, \mu_1 \rangle \approx_{\rho_0}^k \langle \hat{g}_2, \rho_2, \mu_2 \rangle \), if \( \rho_1 \models \Gamma, \Sigma \vdash \mu_1 \approx_{\rho_0}^k \mu_2 \) and

\[
\forall x \in \text{dom}(\Gamma). \Sigma; g_c \vdash \langle \hat{g}_1, \rho_1(x), \mu_1 \rangle \approx_{\rho_0}^k \langle \hat{g}_2, \rho_2(x), \mu_2 \rangle : \Gamma(x)
\]

**Definition 2.23 (Semantic Security Typing).**

\[
\Gamma; \Sigma; \hat{g} \models t : U \iff \forall \ell_0 \in \text{LABEL}, k \geq 0, \rho_1, \rho_2 \in \text{SUBST} \text{ and } \mu_1, \mu_2 \in \text{STORE}, \forall g_c, \hat{g} = \varepsilon g, \varepsilon \vdash \hat{g} \approx g_c, \text{ such that } \Sigma \vdash \mu_1 \text{ and } \Gamma; \Sigma; g_c \vdash \langle \hat{g}_1, \rho_1, \mu_1 \rangle \approx_{\rho_0}^k \langle \hat{g}_2, \rho_2, \mu_2 \rangle, \text{ we have } \Sigma; g_c \vdash \langle \hat{g}, \rho_1(t), \mu_1 \rangle \approx_{\rho_0}^k \langle \hat{g}, \rho_2(t), \mu_2 \rangle : C(U)
\]

**Proposition 2.24 (Security Type Soundness).** \( \Gamma; \Sigma; \hat{g} \vdash t : U \implies \Gamma; \Sigma; \hat{g} \models t : U \)
**Proof.** Proof in Appendix 6.
\[
\begin{align*}
\text{ilbl}(\langle \text{Bool}_{i_1}, \text{Bool}_{i_2} \rangle) &= \langle i_1, i_2 \rangle \\
\text{ilbl}(\langle \text{Unit}_{i_1}, \text{Unit}_{i_2} \rangle) &= \langle i_1, i_2 \rangle \\
\text{ilbl}(\langle \text{Ref}_{i_1} U_1, \text{Ref}_{i_2} U_2 \rangle) &= \langle i_1, i_2 \rangle \\
\text{ilbl}(\langle E_1 \xrightarrow{i_2} E_2, E'_1 \xrightarrow{i'_2} E'_2 \rangle) &= \langle i_1, i'_1 \rangle
\end{align*}
\]

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

\[
\begin{align*}
\text{idom}(\langle E_1 \xrightarrow{i_1} E_2, E'_1 \xrightarrow{i'_2} E'_2 \rangle) &= \langle E'_1, E_1 \rangle \\
\text{idom}(\langle E_1, E_2 \rangle) &= \text{undefined otherwise}
\end{align*}
\]

\[
\begin{align*}
\text{icod}(\langle E_1 \xrightarrow{i_1} E_2, E'_1 \xrightarrow{i'_2} E'_2 \rangle) &= \langle E_2, E'_1 \rangle \\
\text{icod}(\langle E_1, E_2 \rangle) &= \text{undefined otherwise}
\end{align*}
\]

Fig. 16. GSL^{\ell}_{\text{Ref}}: Inversion functions for evidence
\[ \Gamma; g \vdash t \leadsto t' : U \]

\[ \Gamma; g \vdash b \leadsto b : \text{Bool} \]

\[ \Gamma; \Sigma; g + x \leadsto x : U \]

\[ \Gamma; \Sigma; g + \text{unit}_g \leadsto \text{unit}_g : \text{Unit}_g \]

\[ \Gamma; \Sigma; g + (\lambda^\theta x : U_1). t \leadsto (\lambda^\theta x : U_1.t')_g : U_1 \leadsto U_2 \]

\[ \epsilon_1 = g^\Sigma[\text{Bool}_g] \quad \epsilon_2 = g^\Sigma[\text{Bool}_g] \]

\[ \epsilon_1 = g^\Sigma[U_1 \leadsto g U_1] \quad \epsilon_2 = g[U_2 \leq U_1] \quad \epsilon_3 = g[\text{false} \lor g \not\approx g'] \]

\[ \epsilon_1 = g^\Sigma[\text{Label}_g] \quad \epsilon_2 = g[U_2 \leq U_1] \quad \epsilon_3 = g[\text{false} \lor g \not\approx \text{label}(U_1)] \]

\[ \epsilon_1 = g^\Sigma[U_1 \leq U] \quad \epsilon_2 = g[U_1 \leq U] \quad \epsilon_3 = g[U_2 \leq U] \]

\[ \epsilon = g^\Sigma[\text{Ref}_g U] \]

\[ \epsilon = g^\Sigma[\text{Label}_g] \]

where \( g^\Sigma[g] = g[g \not\approx g] \) and \( g^\Sigma[U] = g[U \leq U] \)

Fig. 17. GSL_{Ref} translation to GSL_{Ref}^e terms
\[
\begin{align*}
\text{bounds}(\text{?}) &= [\bot, \top] \\
\text{bounds}(\ell) &= [\ell, \ell] \\
\text{bounds}(x_1 \lor x_2) &= \text{bounds}(x_1) \lor \text{bounds}(x_2) \\
\text{bounds}(x_1 \land x_2) &= \text{bounds}(x_1) \land \text{bounds}(x_2) \\
\text{bounds}(x_1 \setminus x_2) &= \text{bounds}(x_1) \setminus \text{bounds}(x_2) \\
\text{bounds}(F_1(\overline{x})) \lor F_2(\overline{x})) &= \text{bounds}(F_1(\overline{x})) \lor \text{bounds}(F_2(\overline{x})) \\
\text{bounds}(F_1(\overline{x})) \land F_2(\overline{x})) &= \text{bounds}(F_1(\overline{x})) \land \text{bounds}(F_2(\overline{x})) \\
\text{bounds}(F_1(\overline{x}) \setminus F_2(\overline{x})) &= \text{bounds}(F_1(\overline{x}) \setminus \text{bounds}(F_2(\overline{x})) \\
\text{bounds}(F_1(\overline{g})) &= [\ell_1, \ell_2] \\
\text{bounds}(F_2(\overline{g})) &= [\ell_1', \ell_2'] \\
\end{align*}
\]

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

where \( F_1 : \text{GLABEL}^n \to \text{GLABEL} \) and \( F_2 : \text{GLABEL}^m \to \text{GLABEL} \).

\[
\mathcal{G}(F(g_1, \ldots, g_n)) = \mathcal{G}(\overline{F(g_1, \ldots, g_n)})
\]

Fig. 18. GSL\textsuperscript{Ref}: Initial evidence for gradual labels
invert (P₁) = invert (P₁) ∨ invert (P₁)
invert (P₁) = invert (P₁) ∧ invert (P₁)
invert (P₁) = invert (P₁)
tomeet (P₁) = tomeet (P₁) ∨ tomeet (P₁)
tomeet (P₁) = tomeet (P₁) ∨ tomeet (P₁)
tomeet (P₁) = tomeet (P₁) ∨ tomeet (P₁)

\[ G \lliftP(G₁)T₁ : G \lliftP(G₂)T₂ \rrangle = (I₁, I₂) \]
\[ G \lliftP(G₁)T₁ : G \lliftP(G₂)T₂ \rrangle = \langle \text{Bool}_{I₁}, \text{Bool}_{I₂} \rangle \]
\[ G \lliftP(G₂)U₁ : \lliftP(G₁)U₂ \rrangle = \langle E’₁, E’₁ \rangle \]
\[ G \lliftP(G₂)U₁ : \lliftP(G₁)U₂ \rrangle = \langle E₁, E₂ \rangle \]
\[ G \lliftP(G₁)U₁ : \lliftP(G₂)U₂ \rrangle = \langle E’₁, E’₁ \rangle \]
\[ G \lliftP(G₁)U₁ : \lliftP(G₂)U₂ \rrangle = \langle \text{Ref}_{I₁} \cap E₁, \text{Ref}_{I₂} \cap E₂ \cap E’₂ \rangle \]

where \( G₁ : \text{GLABEL}^n \rightarrow \text{GLABEL} \) and \( G₂ : \text{GLABEL}^m \rightarrow \text{GLABEL} \), and \( G₁(x₁, ..., xₙ) = P₁T(x₁, ..., xₙ), \)
\( G₂(x₁, ..., xₙ) = P₂T(x₁, ..., xₙ) \).

\[ G \lliftP(F(U₁, ..., Uₙ)) = G \lliftP(F(U₁, ..., Uₙ) : F(U₁, ..., Uₙ)) \]

Fig. 19. GSLₜ: Initial evidence for gradual types
\[ \begin{align*}
\Sigma; g_c \vdash (\hat{g}_1, \nu_1, \mu_1) & \equiv^k_{\ell_0} (\hat{g}_2, \nu_2, \mu_2) : U \iff g_c \vdash \hat{g}_1 \equiv_{\ell_o} \hat{g}_2 \land \Sigma \vdash \mu_1 \equiv^k_{\ell_0} \mu_2 \land \forall \hat{g}_1 \vdash \nu_1 : U \land \\
(\text{obsVal}_{\ell_0}^U (v_1)) \lor \neg \text{obsVal}_{\ell_0}^U (v_1)) \land ((\text{obsVal}_{\ell_0}^U (v_1) \land \text{obsEv}_{\ell_0}^{\nu_1'} (\nu_1)) \implies \text{obsRef}_{\ell_0}^{\Sigma; g_c, U} (\hat{g}_1, \nu_1, \mu_1, \hat{g}_2, \nu_2, \mu_2))
\end{align*} \]
3 STATIC SECURITY TYPING WITH REFERENCES

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

3.1 SSL_{Ref}: Static type safety

In this section we present the proof of type safety for SSL_{Ref}.

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; ℓ_c ⊢ t : S then ∀ℓ_c′ ≺ ℓ_c, Γ; Σ; ℓ_c′ ⊢ t : S.

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

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

Case (S⊕). Then t = b_1ℓ_1 ⊕ b_2ℓ_2 and

\[
\begin{align*}
\text{(Sb)} & \quad Γ; Σ; ℓ_c ⊢ b_1ℓ_1 : \text{Bool}_{ℓ_1} \\
\text{(Sb)} & \quad Γ; Σ; ℓ_c ⊢ b_2ℓ_2 : \text{Bool}_{ℓ_2} \\
\text{(S⊕)} & \quad Γ; Σ; ℓ_c ⊢ b_1ℓ_1 ⊕ b_2ℓ_2 : \text{Bool}(ℓ_1, ℓ_2)
\end{align*}
\]

Suppose ℓ_c′ such that ℓ_c′ ≺ ℓ_c, then by induction hypotheses on the premises:

\[
\begin{align*}
\text{(Sb)} & \quad Γ; Σ; ℓ_c′ ⊢ b_1ℓ_1 : \text{Bool}_{ℓ_1} \\
\text{(Sb)} & \quad Γ; Σ; ℓ_c′ ⊢ b_2ℓ_2 : \text{Bool}_{ℓ_2} \\
\text{(S⊕)} & \quad Γ; Σ; ℓ_c′ ⊢ b_1ℓ_1 ⊕ b_2ℓ_2 : \text{Bool}(ℓ_1, ℓ_2)
\end{align*}
\]

where ℓ_1′ = ℓ_1 and ℓ_2′ = ℓ_2 and the result holds.

Case (Sprot). Then t = prot_ℓ(t) and

\[
\begin{align*}
\text{(Sprot)} & \quad Γ; Σ; ℓ_c ⊢ ℓ : S \\
& \quad Γ; Σ; ℓ_c ⊢ \text{prot}_ℓ(t) : S \supseteq ℓ
\end{align*}
\]

Suppose ℓ_c′ such that ℓ_c′ ≺ ℓ_c. Considering that ℓ_c′ ⊢ ℓ ≺ ℓ_c ⊢ ℓ, then by induction hypotheses on the premise:

\[
\begin{align*}
\text{(Sprot)} & \quad Γ; Σ; ℓ_c′ ⊢ ℓ : S \\
& \quad Γ; Σ; ℓ_c′ ⊢ \text{prot}_ℓ(t) : S \supseteq ℓ
\end{align*}
\]

and therefore the result holds.

Case (Sapp). Then t = t_1 t_2 and

\[
\begin{align*}
\text{(Sλ)} & \quad D_1 \\
& \quad Γ; Σ; ℓ_c ⊢ t_1 : S_{11} \xrightarrow{ℓ_c′} S_{12} \\
\text{(Sapp)} & \quad Γ; Σ; ℓ_c ⊢ t_2 : S_2 \\
& \quad ℓ_c \supseteq ℓ_c′ \quad S_2 ≺: S_{11}
\end{align*}
\]
Suppose $\ell'_c$ such that $\ell'_c \leq \ell_c$. Then by using induction hypotheses on the premises, considering $S'_{11} \xrightarrow{\ell''_c} S'_{12} \xleftarrow{\ell''_c} S'_{11}$ and $S'_c \xleftarrow{\ell'_c} S'_{11}$. As $S'_c \xleftarrow{\ell''_c} S'_{11}$ and $S_{11} \xleftarrow{\ell''_c} S'_{11}$ then $S'_c \xleftarrow{\ell''_c} S'_{11}$. Also, by definition of the join operator $\ell''_c \vee \ell' \leq \ell_c \vee \ell \leq \ell''_c \leq \ell''_c$, and then:

\[
\begin{array}{c}
\text{(Sd)}
\hline
\Gamma; \Sigma; \ell'_c \vdash t_1 : S'_{11} \xrightarrow{\ell''_c} S'_{12}
\end{array}
\]

\[
\begin{array}{c}
\text{(Sapp)}
\hline
\Gamma; \Sigma; \ell'_c \vdash t_2 : S'_c \\
\Gamma; \Sigma; \ell'_c \vee \ell' \leq \ell''_c \\
\Gamma; \Sigma; \ell'_c \vdash t_1, t_2 : S'_{12} \vee \ell'
\end{array}
\]

Where $S'_{12} \vee \ell' = S_{12} \vee \ell$ and the result holds.

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

\[
\begin{array}{c}
\hline
\text{(Sif)}
\hline
\Gamma; \Sigma; \ell_c \vdash \text{true}_{\ell} : \text{Bool}_{\ell}
\end{array}
\]

\[
\begin{array}{c}
\hline
\text{(Sif)}
\hline
\Gamma; \Sigma; \ell_c \vee \ell \vdash t_1 : S_1
\end{array}
\]

Suppose $\ell'_c$ such that $\ell'_c \leq \ell_c$. As $\ell'_c \vee \ell \leq \ell_c \vee \ell$, by induction hypotheses in the premises:

\[
\begin{array}{c}
\hline
\text{(Sif)}
\hline
\Gamma; \Sigma; \ell'_c \vdash \text{true}_{\ell} : \text{Bool}_{\ell}
\end{array}
\]

\[
\begin{array}{c}
\hline
\text{(Sif)}
\hline
\Gamma; \Sigma; \ell'_c \vee \ell \vdash t_2 : S'_2
\end{array}
\]

\[
\begin{array}{c}
\hline
\text{(Sif)}
\hline
\Gamma; \Sigma; \ell'_c \vdash \text{true}_{\ell} : \text{Bool}_{\ell}
\end{array}
\]

\[
\begin{array}{c}
\hline
\text{(Sif)}
\hline
\Gamma; \Sigma; \ell'_c \vee \ell \vdash t_1 : S'_1
\end{array}
\]

where $S'_1 = S_1, S'_2 = S_2$. Then $(S'_1 \vee S'_2) \vee \ell = (S_1 \vee S_2) \vee \ell$ and therefore the result holds.

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

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

\[
\begin{array}{c}
\hline
\text{(Sref)}
\hline
\Gamma; \Sigma; \ell_c \vdash v : S' \\
S' \xleftarrow{\ell_c} S' \leq \text{label}(S)
\end{array}
\]

\[
\begin{array}{c}
\hline
\text{(Sref)}
\hline
\Gamma; \Sigma; \ell_c \vdash \text{ref}^S v : \text{Ref}_S
\end{array}
\]

Suppose $\ell'_c$ such that $\ell'_c \leq \ell_c$. By using induction hypotheses in the premise, considering $\ell'_c \leq \ell_c \leq \text{label}(S)$:

\[
\begin{array}{c}
\hline
\text{(Sref)}
\hline
\Gamma; \Sigma; \ell'_c \vdash v : S' \\
S' \xleftarrow{\ell'_c} S' \leq \text{label}(S)
\end{array}
\]

\[
\begin{array}{c}
\hline
\text{(Sref)}
\hline
\Gamma; \Sigma; \ell'_c \vdash \text{ref}^S v : \text{Ref}_S
\end{array}
\]

and the result holds.

**Case (Sderef).** Then $t = \text{!}o_{\ell}$ and

\[
\begin{array}{c}
\hline
\text{(Sdref)}
\hline
o : S \in \Sigma
\end{array}
\]

\[
\begin{array}{c}
\hline
\text{(Sdref)}
\hline
\Gamma; \Sigma; \ell_c \vdash o_{\ell} : \text{Ref}_{\ell} S
\end{array}
\]

Suppose $\ell'_c$ such that $\ell'_c \leq \ell_c$, then by using induction hypotheses in the premise:

\[
\begin{array}{c}
\hline
\text{(Sdref)}
\hline
o : S \in \Sigma
\end{array}
\]

\[
\begin{array}{c}
\hline
\text{(Sdref)}
\hline
\Gamma; \Sigma; \ell'_c \vdash o_{\ell} : \text{Ref}_{\ell'} S
\end{array}
\]

\[
\begin{array}{c}
\hline
\text{(Sdref)}
\hline
\Gamma; \Sigma; \ell'_c \vdash \text{!}o_{\ell} : S \vee \ell'
\end{array}
\]

where $\ell' = \ell$ and the result holds.
Case (Sassgn). Then \( t = \alpha_{\ell}:=v \) and

\[
\begin{array}{c}
\alpha : S \in \Sigma \\
\Gamma; \Sigma; \ell_c + \alpha_{\ell} : \text{Ref}_{\ell} S \\
S_2 <: S \\
\hline
\Gamma; \Sigma; \ell_c + \alpha_{\ell} := v : \text{Unit}_1
\end{array}
\]

(Sassgn)

Suppose \( \ell'_c \) such that \( \ell'_c \leq \ell_c \). Considering that \( \ell'_c \leq \ell \leq \ell_c \), then \( S'_2 <: S_2 <: S \), and \( \leq \leq \text{label}(S) \), and \( S'_2 <: S_2 <: S \), then:

\[
\begin{array}{c}
\alpha : S \in \Sigma \\
\Gamma; \Sigma; \ell'_c + \alpha_{\ell} : \text{Ref}_{\ell} S \\
S'_2 <: S \\
\hline
\Gamma; \Sigma; \ell'_c + \alpha_{\ell} := v : \text{Unit}_1
\end{array}
\]

(Sassgn)

but

\[
\text{Unit}_1 <: \text{Unit}_1
\]

and therefore the result holds.

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

\[
\begin{array}{c}
\text{\text{D}} \\
\Gamma; \Sigma; \ell_c + v :: S_1 \\
S_1 <: S \\
\hline
\Gamma; \Sigma; \ell_c + v :: S : S
\end{array}
\]

(S::)

Suppose \( \ell'_c \) such that \( \ell'_c \leq \ell_c \), then by Lemma 3.4

\[
\begin{array}{c}
\text{\text{D}} \\
\Gamma; \Sigma; \ell'_c + v :: S_1 \\
S_1 <: S \\
\hline
\Gamma; \Sigma; \ell'_c + v :: S : S
\end{array}
\]

(S::)

and the result holds.

\[\square\]

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

**Proof.** By induction on the derivation of \( \Gamma; x : S_1; \Sigma; \ell_c + t :: S \). \[\square\]

**Lemma 3.4.** If \( \Gamma; \Sigma; \ell_c + v :: S \) then \( \forall \ell'_c, \Gamma; \Sigma; \ell'_c + v :: S \).

**Proof.** By induction on the derivation of \( \Gamma; \Sigma; \ell_c + v :: S \) observing that for values, there is no premise that depends on \( \ell_c \). \[\square\]

**Proposition 3.5 (\(-\rightarrow\) is well defined).** If \( \vdash \Sigma; \ell_c + t :: S, \vdash \Sigma + \mu \) and \( \forall \ell_r, \) such that \( \ell_r \leq \ell_c \), \( t \mid \mu \xrightarrow{\ell_r} t' \mid \mu' \) then, for some \( \Sigma' \supseteq \Sigma, \vdash \Sigma'; \ell_c + t' :: S', \) where \( S' <: S \) and \( \vdash \Sigma' + \mu' \).

**Proof.**

Case (S\(\oplus\)). Then \( t = b_{1\ell_1} \oplus b_{2\ell_2} \) and

\[
\begin{array}{c}
\vdash \Sigma; \ell_c + b_{1\ell_1} :: \text{Bool}_{\ell_1} \\
\vdash \Sigma; \ell_c + b_{2\ell_2} :: \text{Bool}_{\ell_2} \\
\vdash \Sigma; \ell_c + b_{1\ell_1} \oplus b_{2\ell_2} :: \text{Bool}_{(\ell_1 \lor \ell_2)}
\end{array}
\]

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

$$
\ell_r \quad \frac{b_1 \ell_1 \oplus b_2 \ell_2 \mid \mu}{\ell_r}
$$

Then

$$
(S\oplus) \quad \ell_c \vdash (b_1 \oplus b_2)_{(\ell_1 \lor \ell_2)} : \text{Bool}_{(\ell_1 \lor \ell_2)}
$$

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

$$
\vdash \Sigma; \ell_c \vdash \ell \lor v : S
\quad (S\text{prot})
\vdash \Sigma; \ell_c \vdash \text{prot}_\ell(v) : S \lor \ell
$$

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

$$
\text{prot}_\ell(v) \mid \mu \quad \frac{\ell_r}{v \lor \ell \mid \mu}
$$

But by Lemma 3.2, $\vdash \Sigma; \ell_c \vdash v : S$.

and the result holds.

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

$$
\vdash x : S_{11}; \Sigma; \ell_c' \vdash t : S_{12}
\quad (S\lambda)
\vdash \Sigma; \ell_c \vdash (\lambda^{\ell_c}x : S_{11}.t)\ell : S_{11} \rightarrow S_{12}
\quad \frac{D_1}{\vdash \Sigma; \ell_c \vdash v : S_1}
\quad \frac{\vdash \Sigma; \ell_c \vdash v : S_2}{\vdash \Sigma; \ell_c \vdash (\lambda^{\ell_c}x : S_{11}.t)\ell v : S_1 \lor \ell}
\quad (S\text{app})
\quad \vdash \Sigma; \ell_c \lor \ell \ll \ell_c' \quad \vdash \Sigma; \ell_c \vdash S_2 \ll S_{11}
$$

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

$$
(\lambda^{\ell'}x : S_{11}.t)\ell v \mid \mu \quad \frac{\ell_r}{\text{prot}_\ell([v/x]t) \mid \mu}
$$

But as $\ell_c \lor \ell \ll \ell_c'$ then by Lemma 3.2, $\vdash \Sigma; \ell_c \lor \ell \vdash S'_{12}$, where $S'_{12} \ll S_{12}$.

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

Then

$$
\quad \vdash \Sigma; \ell_c \lor \ell \vdash [v/x]t : S''_{12}
$$

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

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

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

$$
\quad \vdash \Sigma; \ell_c \lor \ell \vdash \text{true}_\ell : \text{Bool}_\ell
\quad \frac{D_0}{\vdash \Sigma; \ell_c \lor \ell \vdash \text{true}_\ell : S_1}
\quad \frac{D_1}{\vdash \Sigma; \ell_c \lor \ell \vdash t_1 : S_1}
\quad \frac{\vdash \Sigma; \ell_c \lor \ell \vdash t_2 : S_2}{\vdash \Sigma; \ell_c \lor \ell \vdash \text{if true}_\ell t_1 \text{ else } t_2 : (S_1 \lor S_2) \lor \ell}
\quad (S\text{if})
\quad \vdash \Sigma; \ell_c \lor \ell \vdash \text{if true}_\ell t_1 \text{ else } t_2 \mid \mu \quad \frac{\ell_r}{\text{prot}_\ell(t_1) \mid \mu}
$$

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

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

\[
\frac{D_1}{\therefore \Sigma; \ell_c \land \ell \vdash S_1}
\]

(Sprot) and therefore

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

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

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

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

\[
\frac{\therefore \Sigma; \ell_c \vdash v : S' \quad S' \vdash : S \quad \ell_c \leq \text{label}(S)}{\therefore \Sigma; \ell_c \vdash \text{ref}^S v : \text{Ref}_\perp S}
\]

(Sref)

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

\[
\text{ref}^S v | \mu \xrightarrow{\ell_r} o_\perp | \mu[o \mapsto v \lor \ell_r]
\]

where \(o \not\in \text{dom}(\mu)\).

Let us take \(\Sigma' = \Sigma, o : S\) and let us call \(\mu' = \mu[o \mapsto v \lor \ell_r]\). 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, \(\vdash \Sigma' ; \bot \vdash v : S' \lor \ell_r\) and \(S' \lor \ell_r \vdash : \Sigma(o) = S\). Therefore \(\vdash \Sigma' \vdash \mu'\).

Then

\[
\frac{o : S \in \Sigma'}{\therefore \Sigma' ; \ell_c \vdash o_\perp : \text{Ref}_\perp S}
\]

(Sl)

and the result holds.

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

\[
\frac{o : S \in \Sigma}{\therefore \Sigma; \ell_c \vdash !o_\ell : S \lor \ell}
\]

(Sderef)

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

\[
!o_\ell | \mu \xrightarrow{\ell_r} v \lor \ell | \mu \text{ where } \mu(o) = v
\]

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

\[
\frac{\therefore \Sigma; \ell_c \vdash v \lor \ell : S' \lor \ell}{\therefore \Sigma; \ell_c \vdash v \lor \ell : S' \lor \ell}
\]

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

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

\[
\frac{o : S \in \Sigma}{\therefore \Sigma; \ell_c \vdash o_\ell : \text{Ref}_\ell S}
\]

(Sasgn)

\[
\frac{\therefore \Sigma; \ell_c \vdash v : S_2 \quad S_2 \vdash : S \quad \ell_c \leq \text{label}(S)}{\therefore \Sigma; \ell_c \vdash o_\ell := v : \text{Unit}_\perp}
\]

(D)

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

\[
\text{o}_\ell := v | \mu \xrightarrow{\ell_r} \text{unit}_\perp | \mu[o \mapsto v \lor \ell_r \lor \ell]
\]

(Su)

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

\[
\frac{\therefore \Sigma; \ell_c \vdash \text{unit}_\perp : \text{Unit}_\perp}{\therefore \Sigma; \ell_c \vdash \text{unit}_\perp : \text{Unit}_\perp}
\]
but

\[
\text{Unit} \perp <: \text{Unit} \perp
\]

and therefore the result holds.

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

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

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

\[
v :: S \mid \mu \xrightarrow{\ell_r} v \vee \text{label}(S) \mid \mu
\]

But \( S_1 <: S \) then \( S_1 \ll S = S \) and therefore \( S_1 \ll \text{label}(S) = S \). Therefore:

\[
\Gamma; \Sigma; \ell_c \vdash v \ll \text{label}(S) : S
\]

and the result holds.

\(\square\)

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

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

**Proof.** By inspection of the type derivation rules. \(\square\)

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

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

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

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

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

\[
\frac{\vdots; \Sigma; \ell_c \ll \ell \vdash t_1 : S_1 \quad \vdots; \Sigma; \ell_c \vdash \text{prot}_\ell(t_1) : S_1 \ll \ell}{\vdots; \Sigma; \ell_c \ll \ell \vdash \text{prot}_\ell(t) : S_1 \ll \ell}
\]

By induction hypotheses, one of the following holds:

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

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

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

Case \((\mathcal{S})\). Then \(t = t_1 \oplus t_2\) and

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

\[
\vdash \Sigma; \ell_c \vdash t_1 \oplus t_2 : \text{Bool}_{(\ell_1 \vee \ell_2)}
\]

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

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:
   a. \(t_2\) is a value. Then by Canonical Forms (Lemma 3.6)

   \[
   (\text{R} \rightarrow) \quad t \mid \mu \xrightarrow{\ell_r} t' \mid \mu
   \]

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

   b. \(t_2 \mid \mu \xrightarrow{\ell_r'} t'_2 \mid \mu'\) for all \(\ell_r'\) such that \(\ell_r' \ll \ell_c\), 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 \((\text{S}f)\), \(t \mid \mu \xrightarrow{\ell_r} t_1 \oplus t'_2 \mid \mu'\) and:

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

   \[
   \vdash \Sigma; \ell_c \vdash t_1 \oplus t'_2 : \text{Bool}_{(\ell_1 \vee \ell_2)}
   \]

   but

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

   and the result holds.

   (2) \(t_1 \mid \mu \xrightarrow{\ell_r} t'_1 \mid \mu'\) for all \(\ell_r'\) such that \(\ell_r' \ll \ell_c\), 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 \((\text{S}f)\), \(t \mid \mu \xrightarrow{\ell_r} t'_1 \oplus t_2 \mid \mu'\) and:

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

   \[
   \vdash \Sigma; \ell_c \vdash t'_1 \oplus t_2 : \text{Bool}_{(\ell_1' \vee \ell_2)}
   \]

   but

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

   and the result holds.

Case \((\mathcal{Sapp})\). Then \(t = t_1 t_2\), \(S = S_{12} \vee \ell\) and

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

\[
S_2 < S_{11} \quad \ell_c \vee \ell < \ell_c'
\]

\[
\vdash \Sigma; \ell_c \vdash t_1 t_2 : S_{12} \vee \ell
\]

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)
\[
\frac{\textit{R=}}{\mu \mid \ell \rightarrow \mu' \mid \mu} \quad \frac{\mu \mid \ell \rightarrow \mu'}{t \mid \mu \mid \ell \rightarrow \mu'}
\]
and by Prop 3.5 ; \Sigma; c + t' : S', where $S' <: S$, therefore the result holds.

(b) $t_2 \mid \mu \mid \ell \rightarrow t_2' \mid \mu'$ for all $\ell_r'$ such that $\ell_r' \leq c$, in particular we pick $\ell_r' = \ell_r$. Then by induction hypothesis, ; $\Sigma_r; c + t_2 : S_2$, where $S_2 <: S$ and ; $\Sigma + \mu'$.

Then by (Sf), $t \mid \mu \mid \ell \rightarrow t_1 \mid \mu'$. But $S_2' <: S_2' <: S_{11}$ and then:

\[
\frac{\text{(Sapp)}}{\therefore \Sigma; c + t_1 : S_{11} \rightarrow_r S_{12} \quad \therefore \Sigma; c + t_2 : S_2 \quad \ell_c \leq \ell \leq \ell_c'}
\]

and the result holds.

(2) $t_1 \mid \mu \mid \ell \rightarrow t_1' \mid \mu'$ for all $\ell_r'$ such that $\ell_r' \leq c$, in particular we pick $\ell_r' = \ell_r$. Then by induction hypotheses, ; $\Sigma_r; c + t_1' : S'_{11} \rightarrow_r S'_{12}$ where $S'_{11} \rightarrow_r S'_{12} <: S_{11} \rightarrow_r S_{12}$, and ; $\Sigma + \mu'$. Then by (Sf), $t \mid \mu \mid \ell \rightarrow t_1 \mid \mu'$. By definition of subtyping, $S_2 <: S_{11} <: S_{11}'$, $\ell_c \leq \ell_c''$ and $\ell' \leq \ell$. Therefore $\ell_c \leq \ell' \leq \ell_c \leq \ell_c'$. Then

\[
\frac{\text{(Sapp)}}{\therefore \Sigma; c + t_1 : S_{11} \rightarrow_r S_{12} \quad \therefore \Sigma; c + t_2 : S_2 \quad \ell_c \leq \ell \leq \ell_c''}
\]

but $S_{12}' \leq \ell' <: S_{12}' \leq \ell$ and the result holds.

Case (Sif). Then $t = \text{if } t_0 \text{ then } t_1 \text{ else } t_2$ and

\[
\frac{\text{\text{(Sif)}}}{\therefore \Sigma; c + t_0 : \text{Bool}_c \quad \therefore \Sigma; c + t_1 : S_1 \quad \therefore \Sigma; c + t_2 : S_2}
\]

By induction hypotheses, one of the following holds:

(1) $t_0$ is a value. Then by Canonical Forms (Lemma 3.6)
\[
\frac{\text{\text{(R=)}}}{\mu \mid \ell \rightarrow \mu' \mid \mu} \quad \frac{\mu \mid \ell \rightarrow \mu'}{t \mid \mu \mid \ell \rightarrow \mu'}
\]

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

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

\[
\frac{\text{\text{(Sif)}}}{\therefore \Sigma; c + t_0' : \text{Bool}_c \quad \therefore \Sigma; c + t_1 : S_1' \quad \therefore \Sigma; c + t_2 : S_2'}
\]

but by definition of join and subtyping $(S_1' \cup S_2') \leq \ell <: (S_1 \cup S_2) \leq \ell$ and the result holds.
Case (S::). Then \( t = t_1 :: S_2 \) and

\[
(S::) - \vdash \Sigma; \ell_c + t_1 :: S_1 \quad S_1 <:: S_2
\]

By induction hypotheses, one of the following holds:

1. \( t_1 \) is a value. Then

\[
(R\rightarrow) - \begin{array}{c}
t | \mu \xrightarrow{\ell_r} t' | \mu \\
t | \mu \xrightarrow{\ell_r} t' | \mu
\end{array}
\]

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

2. \( t_1 | \mu \xrightarrow{\ell_r} t'_1 | \mu' \) for all \( \ell_r' \) such that \( \ell_r' < \ell_c \), 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 \( Sf \), \( t | \mu \xrightarrow{\ell_r} t'_1 :: S_2 | \mu' \).

Also, \( S'_1 <:: S_1 <:: S_2 \) and therefore:

\[
(S::) - \vdash \Sigma; \ell_c + t'_1 :: S'_1 \quad S'_1 <:: S_2
\]

and the result holds.

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

\[
(Sref) - \vdash \Sigma; \ell_c + t_1 :: S'_1 \quad S'_1 <:: S_1 \quad \ell_c \perp \text{label}(S_1)
\]

By induction hypotheses, one of the following holds:

1. \( t_1 \) is a value. Then

\[
(R\rightarrow) - \begin{array}{c}
t | \mu \xrightarrow{\ell_r} t' | \mu \\
t | \mu \xrightarrow{\ell_r} t' | \mu
\end{array}
\]

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

2. \( t_1 | \mu \xrightarrow{\ell_r} t'_1 | \mu' \) for all \( \ell_r' \) such that \( \ell_r' < \ell_c \), 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 \( Sf \), \( t | \mu \xrightarrow{\ell_r} \text{ref}^{S_1} t'_1 | \mu' \) and:

\[
(Sref) - \vdash \Sigma; \ell_c + t'_1 :: S''_1 \quad S''_1 <:: S_1 \quad \ell_c \perp \text{label}(S_1)
\]

and the result holds.

Case (Sderef). Then \( t = !t_1 \) and

\[
(Sderef) - \vdash \Sigma; \ell_c + t_1 :: \text{Ref}_T S_1
\]

By induction hypotheses, one of the following holds:

1. \( t_1 \) is a value. Then by Canonical Forms (Lemma 3.6)

\[
(R\rightarrow) - \begin{array}{c}
t | \mu \xrightarrow{\ell_r} t' | \mu \\
t | \mu \xrightarrow{\ell_r} t' | \mu
\end{array}
\]

and by Prop 3.5, \( \vdash \Sigma; \ell_c + t' :: S' \), therefore the result holds.
(2) \( t_1 \mid \mu \xrightarrow{\ell_r} t'_1 \mid \mu' \) for all \( \ell_r' \) such that \( \ell_r' \leq \ell_c \), in particular we pick \( \ell_r' = \ell_r \). Then by induction hypothesis, \( ; \Sigma; \ell_c \vdash t'_1 : \text{Ref}_\ell S_1 \) where \( \text{Ref}_\ell S_1 <: \text{Ref}_\ell S_1 \) and \( ; \Sigma' \vdash \mu' \). Then by (Sf), \( t \mid \mu \xrightarrow{\ell_r} t' \mid \mu' \) and:

\[
\begin{align*}
(\text{Sderef}) & \quad ; \Sigma; \ell_c \vdash t'_1 : \text{Ref}_\ell S_1 \\
& \quad ; \Sigma; \ell_c \vdash !t'_1 : S_1 \forall \ell' \\
\end{align*}
\]

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

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

\[
\begin{align*}
; \Sigma; \ell_c \vdash t_1 : \text{Ref}_\ell S_1 & \quad ; \Sigma; \ell_c \vdash t_2 : S_2 \\
S_2 <: S_1 & \quad \ell_c \forall \ell \leq \text{label}(S_1) \\
; \Sigma; \ell_c \vdash t_1 := t_2 : \text{Unit}_\perp
\end{align*}
\]

By induction hypotheses, one of the following holds:

1. \( t_2 \) 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)

   \[
   \frac{\begin{align*}
   t_1 & \Rightarrow_{\ell_r} t'_1 \\
   t_2 & \Rightarrow_{\ell_r} \mu'
   \end{align*}}{t \mid \mu \Rightarrow_{\ell_r} t' \mid \mu'}
   \]

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

   b. \( t_2 \mid \mu \Rightarrow_{\ell_r} t'_2 \mid \mu' \) for all \( \ell_r' \) such that \( \ell_r' \leq \ell_c \), in particular we pick \( \ell_r' = \ell_r \). Then by induction hypothesis, \( ; \Sigma'; \ell_c \vdash t_2 : S_2' \) where \( S_2' <: S_2 \) and \( ; \Sigma' \vdash \mu' \).

   Then by (Sf), \( t \mid \mu \Rightarrow_{\ell_r} \mu' \). As \( S_2' <: S_2 <: S_1 \), then:

   \[
   \begin{align*}
   & ; \Sigma; \ell_c \vdash t_1 : \text{Ref}_\ell S_1 \\
   & ; \Sigma; \ell_c \vdash t'_1 : S_2' \\
   S_2' & <: S_1 \\
   & \ell_c \forall \ell \leq \text{label}(S_1) \\
   & ; \Sigma; \ell_c \vdash t_1 := t'_2 : \text{Unit}_\perp
   \end{align*}
   \]

   and the result holds.

2. \( t_1 \mid \mu \Rightarrow_{\ell_r} t'_1 \mid \mu' \) for all \( \ell_r' \) such that \( \ell_r' \leq \ell_c \), in particular we pick \( \ell_r' = \ell_r \). Then by induction hypotheses, \( ; \Sigma'; \ell_c \vdash t'_1 : \text{Ref}_\ell S_1 \), where \( \text{Ref}_\ell S_1 <: \text{Ref}_\ell S_1 \) and \( ; \Sigma' \vdash \mu' \). Then by (Sf), \( t \mid \mu \Rightarrow_{\ell_r} t_1 := t_2 \mid \mu' \). As \( \ell' \leq \ell \) then \( \ell_c \forall \ell' \leq \ell_c \forall \ell \leq \text{label}(S_1) \), and therefore:

   \[
   \begin{align*}
   & ; \Sigma; \ell_c \vdash t'_1 : \text{Ref}_\ell S_1 \\
   & ; \Sigma; \ell_c \vdash t_2 : S_2 \\
   S_2 & <: S_1 \\
   & \ell_c \forall \ell \leq \text{label}(S_1) \\
   & ; \Sigma; \ell_c \vdash t_1 := t_2 : \text{Unit}_\perp
   \end{align*}
   \]

   and the result holds.

\( \square \)

### 3.2 SSL\textsubscript{Ref}: Noninterference

In this section we present the proof of noninterference for SSL\textsubscript{Ref}. 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 \( S' \prec \Gamma(x) \).

**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 \( \Sigma \vdash \langle \ell_1, \rho_1, \mu_1 \rangle \approx_{\ell_o}^k \langle \ell_2, \rho_2, \mu_2 \rangle \), if \( \rho_1 \models \Gamma; \Sigma, \Sigma \vdash \mu_1 \approx_{\ell_o}^k \mu_2 \) and

\[
\forall x \in \Gamma, \Sigma \vdash \langle \ell_1, \rho_1(x), \mu_1 \rangle \approx_{\ell_o}^k \langle \ell_2, \rho_2(x), \mu_2 \rangle : \Gamma(x)
\]

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 \( S' \prec \prec S \).

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

**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 \( \forall j \leq k, \text{if } \Sigma \subseteq \Sigma', \Sigma' + \mu_1' \approx_{\ell_o}^j \mu_2' \text{ then } \Gamma; \Sigma' \vdash \langle \ell_1, \rho_1, \mu_1' \rangle \approx_{\ell_o}^j \langle \ell_2, \rho_2, \mu_2' \rangle \).
Then $t$ such that

ones for reference and assignment, neither of which remove locations.

monotonicity of the join, considering that the label stamping can only make values non observable.

Lemma 3.14. Consider simple values $v_i : S_i$ and

$\Sigma \vdash \langle \ell_1, v_1, \mu_1 \rangle \approx_{\ell_\alpha} \langle \ell_2, v_2, \mu_2 \rangle : S$.

Then $\Sigma \vdash \langle \ell_1, (v_1 \lor \ell), \mu_1 \rangle \approx_{\ell_\alpha} \langle \ell_2, (v_2 \lor \ell), \mu_2 \rangle : S \lor \ell$.

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 $\Sigma; t_1 \vdash t \in T[S], \mu_1 \in Store, \Sigma \vdash \mu_1$, and $\Sigma \vdash \mu_1 \approx_{\ell_\alpha} \mu_2$. Consider $j < k$, posing $t_1 | \mu_1 \xrightarrow{t_1'} \mu_1', \Sigma \subseteq \Sigma', \Sigma' \vdash \mu_1'$ we have

$\Sigma \vdash \langle \ell_1, t_1, \mu_1 \rangle \approx_{\ell_\alpha} \langle \ell_2, t_2, \mu_2 \rangle : C(S)$ if and only if $\Sigma' \vdash \langle \ell_1, t_1', \mu_1' \rangle \approx_{k-j} \langle \ell_2, t_2', \mu_2' \rangle : C(S)$.

Proof. Direct by definition of $S$.

Lemma 3.16. Consider term $\Sigma; t \vdash t : S$, store $\mu$ and $j > 0$, such that $t \mid \mu \xrightarrow{\ell} t' \mid \mu'$. Then $\mu \rightarrow \mu'$.

Proof. Trivial by induction on 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 $\Sigma \vdash \langle \ell_1 \lor \ell'_1, t_1, \mu_1 \rangle \approx_{\ell_\alpha} \langle \ell_2 \lor \ell'_2, t_2, \mu_2 \rangle : C(S)$, and that $\ell_1 \vdash prot_{\ell_1}(t) : S'_i \lor \ell'_i, S'_i \lor \ell'_i < S \lor \ell$ for $i \in \{1, 2\}$. If $\ell_1 \approx_{\ell_\alpha} \ell_2$, and $\ell'_1 \approx_{\ell_\alpha} \ell'_2$, then $\Sigma \vdash \langle \ell_1, \text{prot}_{\ell_1}(t_1), \mu_1 \rangle \approx_{\ell_\alpha} \langle \ell_2, \text{prot}_{\ell_2}(t_2), \mu_2 \rangle : C(S \lor \ell)$.

Proof. Consider $j < k$, we know by definition of related computations that

$t_1 | \mu_1 \xrightarrow{t_1' \mid \mu'_1} t_1' | \mu_1'$

then $\mu_1' \approx_{\ell_\alpha} \mu_2'$, and by Lemma 3.16 $\mu_1 \rightarrow \mu_1'$. If $t'_i$ are reducible after $k - 1$ steps, then the result holds immediately by (Rprot($\ell$)). The interest case if $t'_i$ are irreducible after $j < k$ steps:
Suppose that after \( j \) steps \( t'_i = v_i \), then \( \Sigma + \langle \ell, \ell'_i, v_i, \mu'_i \rangle \approx^{k-j} \langle \ell, \ell'_2, v_2, \mu'_2 \rangle : S \), for some \( \Sigma' \) such that \( \Sigma \subseteq \Sigma' \).

Therefore:

\[
\begin{align*}
\text{prot}_{\ell_i}(t_i) & | \mu'_i \\
\ell_i \mapsto j & \quad \text{prot}_{\ell_i}(v_i) | \mu'_i \\
\ell_i \mapsto 1 & \quad (v_i \equiv \ell'_i) | \mu'_i
\end{align*}
\]

Let us suppose \( \Sigma' ; \ell_i \vdash v_i : S''_i \), where \( S''_i :< S'_i :< S \). Then \( \Sigma' ; \ell_i \vdash v_i \equiv \ell'_i : S'' \equiv \ell'_i, \) and \( S''_i \equiv \ell'_i : S \equiv \ell \).

If \( \neg \text{obs}_{\ell_o}(\ell_i \equiv \ell'_i) \) by monotonicity of the join either \( \neg \text{obs}_{\ell_o}(\ell'_i) \) or \( \neg \text{obs}_{\ell_o}(\ell_i) \). If \( \neg \text{obs}_{\ell_o}(\ell'_i) \) then \( \neg \text{obs}_{\ell_o}(\ell \equiv \ell'_i) \) and the result holds. If \( \neg \text{obs}_{\ell_o}(\ell_i \equiv \ell'_i, S) \) then \( \text{obs}_{\ell_o}(\ell_i \equiv \ell'_i, S) \), then the result follows by Lemma 3.14, and by backward preservation of the relations (Lemma 3.15).

\[ \square \]

**Lemma 3.18.** Consider \( \ell \), such that \( \neg \text{obs}_{\ell_o}(\ell) \), then then \( \forall k > 0 \), such that \( \Sigma ; \ell \vdash t : S, \Sigma \vdash \mu \)

\[
t | \mu \quad \ell \mapsto k \ t' | \mu', \text{then } \forall \ell',
\]

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

**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

\[
o_{ref} := v \quad \ell \mapsto \text{unit}_{\bot} | \mu[o \mapsto (v \equiv (\ell \equiv \ell'))]
\]

Next we have to prove that \( \text{obs}_{\ell_o}(\text{label}(\Sigma(o))) \) is not defined. As \( \Sigma ; \ell \vdash t : S \), then we know that \( \ell \equiv \ell' \equiv \text{label}(\Sigma(o)) \), and as \( \neg (\text{obs}_{\ell_o}(\ell)) \) by monotonicity of the join the result holds.

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

\[
\text{ref}^S v | \mu \quad \ell \mapsto o_{\bot} | \mu[o \mapsto (v \equiv \ell)]
\]

where \( o \notin \text{dom}(\mu) \). We need to prove that \( \text{obs}_{\ell_o}(\text{label}(\ell \equiv \ell)) \) does not hold, which follows directly by monotonicity of the join.

\[ \square \]

**Lemma 3.19.** Consider \( \ell \), such that \( \text{obs}_{\ell_o}(\ell) \) does not hold, then then \( \forall k > 0 \), such that

\( \Sigma ; \ell \vdash t_i : S_i \), and that \( t_i \mid \mu_i \quad \ell \mapsto k t'_i \mid \mu'_i \), then if \( \Sigma \vdash \mu_i \approx^{k} \mu_2 \), then \( \Sigma' \vdash \mu'_i \approx^{k} \mu'_2 \) for some \( \Sigma' \) such that \( \Sigma \subseteq \Sigma' \) and that \( \Sigma' ; \ell \vdash t'_i : S'_i \), where \( S'_i :< S_i \).

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

\( (1) \forall o \in \text{dom}(\mu'_i) \setminus \text{dom}(\mu_i), \text{obs}_{\ell_o}(\ell, \mu'_i(o)) \) does not hold, i.e. new locations are not observable and therefore as \( \Sigma' ; \ell \vdash t'_i : S \), then \( \neg \text{obs}_{\ell_o}(\text{label}(\Sigma(o))) \).

\( (2) \forall o \in \text{dom}(\mu'_i) \cap \text{dom}(\mu_i) \land \mu'_i(o) \neq \mu(o), \neg \text{obs}_{\ell_o}(\text{label}(\Sigma(o))) \)

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 \( \Sigma' \vdash \mu'_i \approx^{k} \mu'_2 \) and the result holds.

\[ \square \]
LEMMA 3.20. Suppose that \( \Sigma; \ell_1 \vdash \text{prot}_{\ell_1'}(t_1) : S' \not\subset \ell_1', S' \not\subset \ell_1' < : S \) for \( i \in \{1, 2\} \), where \( \neg \text{obs}_{\ell_o}(\ell_i \lor \ell_i') \).
Also consider two stores \( \mu_i \) such that \( \Sigma \vdash \mu_1 \approx_{\ell_o}^k \mu_2 \).
Then \( \Sigma + \langle \ell_1, \text{prot}_{\ell_1'}(t_1), \mu_1 \rangle \approx_{\ell_o}^k \langle \ell_2, \text{prot}_{\ell_2'}(t_2), \mu_2 \rangle : C(S) \)

PROOF. Suppose that after at least \( j \) more steps, where \( j < k \), both subterms reduce to a value:
\[
t \mid \mu_i \quad \ell_i \lor \ell_i' \quad j \quad \nu_i \mid \mu_i'
\]
Therefore:
\[
\text{prot}_{\ell_i}(t) \mid \mu_i'
\]
\[
\infer[\ell_i] j \quad \text{prot}_{\ell_i}(\nu_i) \mid \mu_i'
\]
\[
\infer[\ell_i] 1 \quad \nu_i \mid \ell_i'
\]
As the values can be radically different we have to make sure that both values are not observables.
If \( \neg \text{obs}_{\ell_o}(\ell_i) \) then the values are not observables because the security context is not observable.
Let us assume that \( \text{obs}_{\ell_o}(\ell_i) \) holds, but \( \text{obs}_{\ell_o}(\ell_i') \) not. Then by monotonicity of the join, \( \neg \text{obs}_{\ell_o}(\text{label}(\nu_i) \lor \ell_i') \) 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.

\[ \square \]

Next, we present the Noninterference proposition.

PROPOSITION 2.5 (SECURITY TYPE SOUNDNESS). If \( \Gamma; \Sigma; \ell_c \vdash t : S'_t \Rightarrow \forall S, S'_t \not\subset : S, \Gamma; \Sigma; \ell_c \vdash t : S \)

PROOF. We proceed by proving a more general proposition instead:
If \( \Gamma; \Sigma; \ell_i \vdash t : S'_t, S'_t \not\subset : S \), then \( \forall \mu_i \in \text{STORE}, \Sigma \vdash \mu_i \), and \( \forall k \geq 0, \forall \rho_i \in \text{SUBST}, \Gamma; \Sigma \vdash \langle \ell_i, \rho_i, \mu_1 \rangle \approx_{\ell_o}^k \langle \ell_2, \rho_2, \mu_2 \rangle \), we have \( \Sigma \vdash \langle \ell_1, \rho_1(t), \mu_1 \rangle \approx_{\ell_o}^k \langle \ell_2, \rho_2(t), \mu_2 \rangle : C(S) \).

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

Case (x). \( t = x \) and \( \Gamma(x) = S, \Gamma; \Sigma \vdash \langle \ell_1, \rho_1, \mu_1 \rangle \approx_{\ell_o}^k \langle \ell_2, \rho_2, \mu_2 \rangle \) implies by definition that
\( \Sigma \vdash \langle \ell_1, \rho_1(x), \mu_1 \rangle \approx_{\ell_o}^k \langle \ell_2, \rho_2(x), \mu_2 \rangle : S \), 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, \( \Sigma \vdash \langle \ell_1, b_g, \mu_1 \rangle \approx_{\ell_o}^k \langle \ell_2, b_g, \mu_2 \rangle : \text{Bool} \) as required.

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

Case (λ). \( t = (\lambda_{\ell_c} x : S'_t, t_1)_{\ell'} \). Then \( S'_t = S'_t \xrightarrow{\ell_c} S'_t_{12}, \text{and } S = S_1 \xrightarrow{\ell_c} S_2, \text{where } S' \not\subset : S \).
By definition of substitution, assuming \( x \notin \text{dom}(\rho_1) \), and Lemma 3.10:
\[
\Gamma; \Sigma; \ell_i \vdash \rho_i(t) = \Gamma; \Sigma; \ell_i \vdash (\lambda_{\ell_c} x : S_1, \rho_1(t_1))_{\ell'} : S_1 \xrightarrow{\ell_c} S_1_{12} \]

where \( S'_{12} \not\subset : S'_2 \). Consider \( j \leq k, \mu'_1, \mu'_2 \) such that \( \mu_i \rightarrow \mu'_i \) and \( \Sigma \subseteq \Sigma' \Sigma' + \mu'_1 \approx_{\ell_o}^j \mu'_2 \), and assume two values \( \nu_1 \) and \( \nu_2 \) such that \( \Sigma' \vdash \langle \ell_1, \nu_1, \mu'_1 \rangle \approx_{\ell_o}^j \langle \ell_2, \nu_2, \mu'_2 \rangle : S_1 \).
We need to show that:

\[ \Sigma' \vdash \langle \ell_1, (\lambda^{\ell_1} x : S'_1 . \rho_1(t_1)) \rangle_{\mu_1} \]

Then:

\[ \langle \ell_2, (\lambda^{\ell_2} x : S'_2 . \rho_2(t_2)) \rangle_{\mu_2} : C(S_2) \]

We then extend the substitutions to map \( x \) to the arguments:

\[ \rho'_1 = \rho_1[x \mapsto v_1] \]

We know that \( \Sigma' \vdash \langle \ell_1, v_1, \mu'_1 \rangle \approx_{\ell_o} \langle \ell_2, v_2, \mu'_2 \rangle : S_1 \). So as \( \mu_i \rightarrow \mu'_i \) then by Lemma 3.11, \( \Gamma, x : S_1 ; \Sigma' \vdash \langle \ell_1, \rho'_1, \mu'_1 \rangle \approx_{\ell_o} \langle \ell_2, \rho'_2, \mu'_2 \rangle \).

By Lemma 3.10, \( \Gamma, S ; \Sigma' ; \ell''_c \vdash \rho'_1(t_1) : S'_1 \) where \( S'_1 \ll S'_2 \ll S_2 \). We know that \( \ell_1 \ll \ell' \ll \ell''_c \), therefore by Lemma 3.2, \( \Gamma, \Sigma' ; \ell_1 \ll \ell \vdash \rho'_1(t_1) : S'_2 \). Then by induction hypothesis and Lemma 3.13:

\[ \Sigma' \vdash \langle \ell_1 \ll \ell', \rho'_1(t_1), \mu'_1 \rangle \approx_{\ell_o} \langle \ell_2 \ll \ell', \rho'_2(t_1), \mu'_2 \rangle : C(S_2), \]

Finally, by Lemma 3.17:

\[ \Sigma' \vdash \langle \ell_1, \text{prot}_{\ell'}(\rho'_1(t_1)), \mu'_1 \rangle \approx_{\ell_o} \langle \ell_2, \text{prot}_{\ell'}(\rho'_2(t_1)), \mu'_2 \rangle : C(S_2) \]

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

---

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

By definition of substitution:

\[ \rho_1(t) = !\rho_1(t') \]

We have to show that:

\[ \Sigma \vdash \langle \ell_1, !\rho_1(t'), \mu_1 \rangle \approx_{\ell_o} \langle \ell_2, !\rho_1(t'), \mu_2 \rangle : C(S) \]

By Lemma 3.10:

\[ \Sigma ; \ell_1 \vdash !\rho_1(t') : S_1 \ll \ell''_c \ll \ell \]

where \( \ell''_c \ll \ell'' \ll \ell \). By induction hypotheses on the subterm:

\[ \Sigma \vdash \langle \ell_1, \rho_1(t'), \mu_1 \rangle \approx_{\ell_o} \langle \ell_2, \rho_2(t'), \mu_2 \rangle : C(\text{Ref}_{\ell} S_1) \]

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

\[ \rho_1(t') | \mu_i \xrightarrow{\ell_i} j ! t'_i | \mu'_i \implies \Sigma \subseteq \Sigma', \Sigma' \vdash \mu'_i \approx_{\ell_o} \mu'_2 \wedge (\text{irred}(t'_i)) \implies \Sigma' \vdash \langle \ell_1, t'_1, \mu'_1 \rangle \approx_{\ell_o} \langle \ell_2, t'_2, \mu'_2 \rangle : \text{Ref}_{\ell} S_1 \]

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

\[ !\rho_1(t) | \mu_i \xrightarrow{\ell_i} j ! v_i | \mu'_i \] and the result holds.

If after at most \( j \) steps \( t'_i \) is irreducible it means that for some \( j' \leq j \), \( !\rho_1(t) | \mu_i \xrightarrow{\ell_i} j' ! v_i | \mu'_i \). If \( j' = j \) then we use the same same argument for reducible terms and the result holds.
Let us consider now \( j' < j \). Then \( \Sigma' \vdash \langle \ell_1, v_1, \mu'_1 \rangle \approx^{k-j'} \langle \ell_2, v_2, \mu'_2 \rangle : \text{Ref}_\ell S_1 \). By Lemma 3.6, each \( v_i \) is a location \( o_i \ell_i' \), such that \( \Sigma'(o_i \ell_i') = \text{Ref}_\ell S_1 \) and \( \ell_i' \leq \ell' \). Then:

\[
\rho_i(t) \mid \mu_i \xrightarrow{\ell_i} j+1 \mu_i' \\
\rho_i(t) \mid \mu_i \xrightarrow{\ell_i} 1 \text{ prot}_{\ell_i'}(v'_i) \mid \mu_i'
\]

with \( \ell_i' \leq \ell'' \), \( v'_i = \mu'_i(o_i \ell_i') \). As \( \Sigma' \vdash \langle \ell_1, v_1, \mu'_1 \rangle \approx^{k-j} \langle \ell_2, v_2, \mu'_2 \rangle : \text{Ref}_\ell S_1 \), 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 \( \Sigma' \vdash \langle \ell_1, v_1', \mu'_1 \rangle \approx^{k-j'} \langle \ell_2, v_2', \mu'_2 \rangle : S_1 \), by Lemma 6.59,

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

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

---

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

By definition of substitution:

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

and Lemma 3.10:

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

We have to show that

\[
\Sigma \vdash \langle \ell_1, \rho_1(t_1) : \rho_1(t_2), \mu_1 \rangle \\
\approx^k \langle \ell_2, \rho_2(t_1) : \rho_2(t_2), \mu_2 \rangle : C(S)
\]

By induction hypotheses

\[
\Sigma \vdash \langle \ell_1, \rho_1(t_1), \mu_1 \rangle \approx^k \langle \ell_2, \rho_2(t_1), \mu_2 \rangle : C(S_1)
\]

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:

\[
\rho_i(t_1) \mid \mu_i \xrightarrow{\ell_i} j \mu_i \Rightarrow \Sigma \subseteq \Sigma', \Sigma' \vdash \mu_i' \approx^{k-j_1} \mu'_2 \land \Sigma' \vdash \langle \ell_1, v_1, \mu'_1 \rangle \approx^{k-j_1} \langle \ell_2, v_2, \mu'_2 \rangle : \text{Ref}_\ell S_1
\]

By Lemma 3.16 \( \mu_i \rightarrow \mu'_i \), and \( \mu'_i \approx^{k-j_1} \mu'_2 \) then by Lemma 6.41, \( \Sigma' \vdash \langle \ell_1, \rho_1(t_1), \mu'_1 \rangle \approx^{k-j_1} \langle \ell_2, \rho_2(t_1), \mu'_2 \rangle : S_2 \). By induction hypotheses:

\[
\Sigma' \vdash \langle \ell_1, \rho_1(t_2), \mu'_1 \rangle \approx^{k-j_1} \langle \ell_2, \rho_2(t_2), \mu'_2 \rangle : C(S_2)
\]

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 \( \rho_i(t_2) \) reduces to values \( v'_2 \):

\[
\rho_i(t_2) \mid \mu_i' \xrightarrow{\ell_i} j + j_2 \mu_i' \Rightarrow \Sigma' \subseteq \Sigma'', \Sigma'' \vdash \mu''_1 \approx^{k-j_1-j_2} \mu''_2 \land \Sigma'' \vdash \langle \ell_1, v'_2, \mu''_1 \rangle \approx^{k-j_1-j_2} \langle \ell_2, v'_2, \mu''_2 \rangle : S_2
\]

Then

\[
\rho_i(t_2) \mid \mu_i' \xrightarrow{\ell_i} j + j_2 \mu_i' \Rightarrow \mu''_1 \approx^{k-j_1-j_2} \mu''_2
\]

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

\[
\begin{align*}
\ell_i & \mapsto 1 \mu_i'' \mid \mu_i'' \\
\end{align*}
\]

\[
\begin{align*}
\ell_i & \mapsto 1 \mu_i'' \mid \mu_i'' \\
\end{align*}
\]
Where \( \mu''' = \mu''[o \mapsto (v'_i \vee (\ell_i \land \ell'_i))] \). As \( \Sigma'' \vdash \langle \ell_1, v'_1, \mu''_1 \rangle \approx_{\ell_o}^k \langle \ell_2, v'_2, \mu''_2 \rangle : S_2 \), and as \( \ell_i \land \ell'_i \leq \text{label}(S_i) \), where \( \ell'_i \leq \ell_i \), and \( \text{label}(v'_i) \leq \text{label}(S_i) \), then \( \Sigma'' ; \ell_i \vdash v'_i \lor (\ell_i \lor \ell'_i) : S' \) and \( S' <: S_1 \). Then by monotonicity of the join Lemma 3.14,

\[
\Sigma'' \vdash \langle \ell_1, (v'_i \lor (\ell_i \lor \ell'_i)), \mu''_1 \rangle \approx_{\ell_o}^k \langle \ell_2, (v'_2 \lor (\ell_2 \lor \ell'_2)), \mu''_2 \rangle
\]

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

\[
\Sigma'' \vdash \langle \ell''_1, (v'_i \lor (\ell_i \lor \ell'_i)), \mu''_1 \rangle \approx_{\ell_o}^k \langle \ell''_2, (v'_2 \lor (\ell_2 \lor \ell'_2)), \mu''_2 \rangle
\]

As every values are related at type Unit, we only have to prove that \( \Sigma'' + \mu''_1 \approx_{\ell_o}^k \mu''_2 \), but using monotonicity (Lemma 6.46), 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:

\[ \rho_1(t) = \text{ref}^{S_i} \rho_1(t') \]

and Lemma 3.10:

\[ \ell_i \vdash \text{ref}^{S_i} \rho_1(t') : \text{Ref}_\perp S_1 \]

We have to show that

\[ \Sigma \vdash \langle \ell_1, \text{ref}^{S_i} \rho_1(t'), \mu_1 \rangle \approx_{\ell_o}^k \langle \ell_2, \text{ref}^{S_i} \rho_2(t'), \mu_2 \rangle : C(S_1) \]

As \( \Sigma; \ell_i \vdash \rho_1(t') : S'_i \) where \( S'_i <: S_1 \), by induction hypotheses:

\[ \Sigma \vdash \langle \ell_1, \rho_1(t'), \mu \rangle \approx_{\ell_o}^k \langle \ell_2, \rho_2(t'), \mu \rangle : C(S_1) \]

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

\[ \rho_1(t') | \mu_1 \xrightarrow{\ell_i} j t'_i | \mu''_i \implies \Sigma \subseteq \Sigma', \Sigma' + \mu_1 \approx_{\ell_o}^{k-j} \mu'_2 \land (\text{irred}(t'_i) \implies \Sigma' \vdash \langle \ell_1, t'_i, \mu'_1 \rangle \approx_{\ell_o}^{k-j} \langle \ell_2, t'_2, \mu'_2 \rangle : S'_i) \]

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

\[ \text{ref}^{S_i} \rho_1(t') | \mu_1 \xrightarrow{\ell_i} j \text{ref}^{S_i} t'_i | \mu''_i \] 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_1(t') | \mu_1 \xrightarrow{\ell_i} j' \text{ref}^{S_i} v_i | \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:

\[ \rho_1(t) | \mu \xrightarrow{\ell_i} j' + 1 \text{ ref}^{S_i} v_i | \mu''_i \]

with, \( \mu''_i = \mu''[o \mapsto (v'_i \lor (\ell_i \land \ell'_i))] \). Also, as \( \Sigma' \vdash \langle \ell_1, v_1, \mu'_1 \rangle \approx_{\ell_o}^{k-j} \langle \ell_2, v_2, \mu'_2 \rangle : S_1 \), then \( \Sigma'' \vdash \langle \ell_1, v_1, \mu''_1 \rangle \approx_{\ell_o}^{k-j} \langle \ell_2, v_2, \mu'_2 \rangle : S_1 \), with \( \Sigma'' = \Sigma', o : S_1 \). And as \( \text{label}(v_i) \lor \ell_i \leq \text{label}(S_i) \), then by Lemma 3.14, \( \Sigma'' \vdash \langle \ell_1, v_1 \lor \ell_1, \mu''_1 \rangle \approx_{\ell_o}^{k-j} \langle \ell_2, v_2 \lor \ell_2, \mu'_2 \rangle : S_1 \).

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

\[ \Sigma'' \vdash \langle \ell''_1, v_1 \lor \ell_1, \mu''_1 \rangle \approx_{\ell_o}^{k-j} \langle \ell''_2, v_2 \lor \ell_2, \mu'_2 \rangle : S_1 \]. By definition of related stores \( \Sigma'' \vdash \mu''_1 \approx_{\ell_o}^{k-j} \mu''_2 \). Then by Monotonicity of the relation (Lemma 6.46) \( \Sigma'' \vdash \mu''_1 \approx_{\ell_o}^{k-j-2} \mu''_2 \) 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:

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

with $S'' :<: S'_i :<: S$. We use a similar argument to case $=: \mathcal{F}$ 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:

$$\rho_i(t_1) \mid \mu_i \xrightarrow{\ell_i} j_i \nu_{i1} \mid \mu_i' \Rightarrow \Sigma \subseteq \Sigma', \Sigma' + \mu_i' \approx \ell_o \mu_2 \wedge \Sigma' + \langle \ell_1, \nu_{i1}, \mu_i' \rangle \approx \ell_o \langle \ell_2, \nu_{i2}, \mu_i' \rangle : S_1$$

$$\rho_i(t_2) \mid \mu_i' \xrightarrow{\ell_i} j_i \nu_{i2} \mid \mu_i'' \Rightarrow \Sigma' \subseteq \Sigma'', \Sigma'' + \mu_i'' \approx \ell_o \mu_2'' \wedge \Sigma'' + \langle \ell_1, \nu_{i2}, \mu_i'' \rangle \approx \ell_o \langle \ell_2, \nu_{i2}, \mu_i'' \rangle : S_2$$

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

$$\rho_i(t) \mid \mu_i'' = \underbrace{(b_{i1})_{\ell_{i1}} \oplus (b_{i2})_{\ell_{i2}}}_{\ell_{i}'} \mid \mu_i''$$

with $b_i = b_{i1} \oplus b_{i2}$, $\ell_i = \ell_{i1} \wedge \ell_{i2}$, and $\ell_i' \leq \text{label}(S'' \nu_{ij}) \leq \text{label}(S)$.

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 $\text{obs}_{\ell_o}(\ell_i)$, then by monotonicity of the join, $\neg \text{obs}_{\ell_o}(\ell_i')$ and the result holds. If $\text{obs}_{\ell_o}(\ell_{ij})$ 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 $\Sigma; \ell_i \vdash t_1 : S_{i1} \xrightarrow{\ell_{ei}} S_{i2}$, and $\Sigma; \ell_i \vdash t_2 : S''_{i1}$.

Also $S_{i1} \xrightarrow{\ell_{ei}} S_{i2} :<: S_{1}$, and $S = S_2$.

By definition of substitution:

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

and Lemma 3.10:

$$\Sigma; \ell_i \vdash \rho_i(t_1) \rho_i(t_2) : S_{i2}'$$

with $S_{i2}' :<: S_{i2} :<: S_2$. We use a similar argument to case $=: \mathcal{F}$ 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 reduced computations:

$$\rho_i(t_1) \mid \mu_i \xrightarrow{\ell_i} j_i \nu_{i1} \mid \mu_i' \Rightarrow \Sigma \subseteq \Sigma', \Sigma' + \mu_i' \approx \ell_o \mu_2 \wedge \Sigma' + \langle \ell_1, \nu_{i1}, \mu_i' \rangle \approx \ell_o \langle \ell_2, \nu_{i2}, \mu_i' \rangle : S_1$$

$$\rho_i(t_2) \mid \mu_i' \xrightarrow{\ell_i} j_i \nu_{i2} \mid \mu_i'' \Rightarrow \Sigma' \subseteq \Sigma'', \Sigma'' + \mu_i'' \approx \ell_o \mu_2'' \wedge \Sigma'' + \langle \ell_1, \nu_{i2}, \mu_i'' \rangle \approx \ell_o \langle \ell_2, \nu_{i2}, \mu_i'' \rangle : S_2$$

Then

$$\rho_i(t) \mid \mu_i \xrightarrow{\ell_i} j_i \nu_{i1} \nu_{i2} \mid \mu_i''$$
Finally, by backward preservation of the relations (Lemma 3.15) the result holds.

If \( \neg \text{obs}_{\ell_0}(\ell_i, v_{i1}) \), and we assume by canonical forms that \( v_{i1} = (\lambda_{\ell_i}^x.t_i)_{\ell_i'} \) then, either \( \neg \text{obs}_{\ell_0}(\ell_i) \) or \( \neg \text{obs}_{\ell_0}(\ell_i') \) and

\[
\begin{align*}
\text{reduce via different branches of the conditional.}
\end{align*}
\]

If either \( \neg \text{obs}_{\ell_0}(\ell_i) \) or \( \neg \text{obs}_{\ell_0}(\ell_i') \) then by Lemma 3.20,

\[
\begin{align*}
\Sigma' &+ \langle \ell_1, \text{prot}_{\ell_1}(v_{i1}), \mu_1' \rangle \\
\approx_{\ell_0}^{k-j-h} &\langle \ell_2, \text{prot}_{\ell_2}(v_{i2}), \mu_2' \rangle : C(S_2 \land \ell)
\end{align*}
\]

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 \( \Sigma; \ell_i + t_1 : S_1, \Sigma; \ell_i + t_2 : S_2, \Sigma; \ell_i + t_3 : S_3, \ell_i = \ell_i \land \text{label}(S_1) \), and \( S' = S_2 \lor S_3 < S \).

By definition of substitution:

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

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

\[
\rho_i(t_1) \mid \mu_i \xleftarrow{\ell_i} j_{i1} v_{i1} \mid \mu_i' \implies \Sigma \subseteq \Sigma', \Sigma' + \mu_i' \approx_{\ell_0}^{k-j} \mu_2' \land \Sigma' + \langle \ell_1, v_{i1}, \mu_i' \rangle \approx_{\ell_0}^{k-j} \langle \ell_2, v_{i2}, \mu_2' \rangle : S_1
\]

By Lemma 3.6, each \( v_{i1} \) is a boolean \((b_{i1})_{\ell_{i1}}\), such that \( \Sigma'; \ell_i + (b_{i1})_{\ell_{i1}} : \text{Bool}_{\ell_{i1}} \land \text{Bool}_{\ell_{i1}} \approx S_1 \), implies \( S_1 = \text{Bool}_{\ell_{i1}} \). Then:

\[
\rho_i(t) \mid \mu_i \xleftarrow{\ell_i} j_{i1} v_{i1} \mid \mu_i' \implies (b_{i1})_{\ell_{i1}} \text{ then } \rho_i(t_2) \text{ else } \rho_i(t_3) \mid \mu_i'
\]

Let us consider \( \neg \text{obs}_{\ell_0}(\ell_i, (b_{i1})_{\ell_{i1}}) \). Let us assume the worst case scenario and that both execution reduce via different branches of the conditional. Then:

\[
\rho_i(t) \mid \mu_i \xleftarrow{\ell_i} j_{i1} v_{i1} \mid \mu_i' \implies (b_{i1})_{\ell_{i1}} \text{ then } \rho_i(t_2) \text{ else } \rho_i(t_3) \mid \mu_i'
\]

But because \( \neg \text{obs}_{\ell_0}(\ell_i, (b_{i1})_{\ell_{i1}}) \), then either \( \neg \text{obs}_{\ell_0}(\ell_i) \) or \( \neg \text{obs}_{\ell_0}(\ell_i') \) and therefore, \( \neg \text{obs}_{\ell_0}(\ell_i \land \ell_{i1}) \). Then by Lemma 3.20,

\[
\Sigma' + \langle \ell_1, \text{prot}_{\ell_1}(v_{i2}), \mu_i' \rangle \approx_{\ell_0}^k \langle \ell_2, \text{prot}_{\ell_2}(v_{i3}), \mu_2' \rangle
\]

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

Now let us consider if \( \text{obs}_{\ell_0}(\ell_i, (b_{i1})_{\ell_{i1}}) \) holds. Then by definition of \( \approx_{\ell_0} \) on boolean values, \( b_{i1} = b_{i2} \). Because \( b_{i1} = b_{i2} \), both \( \rho_i(t) \) 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 $\Sigma' \vdash \langle \ell_1 \gamma \ell_1, \rho_1(t_2), \mu'_1 \rangle \approx^k \ell_2 \gamma \ell_2, \rho_2(t_2), \mu'_2 : S_2$, and by Lemma 3.17,
\[ \Sigma' \vdash \langle \ell_1, \text{prot}_{\ell_1}(\rho_1(t_2)), \mu'_1 \rangle \approx^k \ell_2, \text{prot}_{\ell_2}(\rho_2(t_2)), \mu'_2 : S \]
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 (α is Sound).** If $\widehat{\ell} \not\in \emptyset$ then $\gamma(\alpha(\widehat{\ell}))$.

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

\[ \square \]

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

**Proof.** By case analysis on the structure of $g$. If $g = \ell$, $\gamma(g) = \{\ell\}$; $\widehat{\ell} \subseteq \emptyset$, $\widehat{\ell} \not\in \emptyset$ implies $\alpha(\widehat{\ell}) = \alpha(\{\ell\}) = \ell \subseteq g$ (if $\widehat{\ell} = \emptyset$, $\alpha(\widehat{\ell})$ is undefined). If $g = \emptyset$, $g' \subseteq g$ for all $g'$.

\[ \square \]

**Proposition 4.3 (α is Sound and Optimal).** If $\widehat{\ell} \not\in \emptyset$ then,

(i) $\widehat{\ell} \subseteq \gamma(\alpha(\widehat{\ell}))$.

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

**Proof.** Trivial using Prop 4.1 and 4.2.

\[ \square \]

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

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

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

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

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

We then consider cases on $\widehat{S}$ according to the definition of $\alpha_S$.

**Case (\{ \text{Bool}_{\ell} \}).**
\[ \gamma_S(\alpha_S(\{ \text{Bool}_{\ell} \})) = \gamma_S(\text{Bool}_{\alpha(\ell)}) \]
\[ = \{ \text{Bool}_{\ell} \mid \ell \in \gamma(\alpha(\{ \ell \})) \} \]
\[ \supseteq \{ \text{Bool}_{\ell} \} \text{ by soundness of } \alpha. \]
Case $\big(\begin{array}{c} \ell \rightarrow \ell, S_{11} \rightarrow \ell, S_{12} \big)$. 

$$
\gamma_S(\alpha_S(\big(\begin{array}{c} \ell \rightarrow \ell, S_{11} \rightarrow \ell, S_{12} \big) )) 
= \gamma_S(\alpha_S(\big(\begin{array}{c} \ell \rightarrow \ell, S_{11} \big) )) 
= \gamma_S(\big(\begin{array}{c} \ell \rightarrow \ell, S_{11} \big) )) 
\subseteq \{ \ell \rightarrow \ell, S_{11} \rightarrow \ell, S_{12} \} 
$$

by induction hypothesis on $\{ S_{11} \}$ and $\{ S_{12} \}$, and soundness of $\alpha$.

Case $\big(\begin{array}{c} \ell \rightarrow \ell, S_{11} \big)$. 

$$
\gamma_S(\alpha_S(\big(\begin{array}{c} \ell \rightarrow \ell, S_{11} \big) )) 
= \gamma_S(\big(\begin{array}{c} \ell \rightarrow \ell, S_{11} \big) )) 
\subseteq \{ \ell \rightarrow \ell, S_{11} \} 
$$

by induction hypothesis on $\{ S_{11} \}$ and soundness of $\alpha$. 

□

**Proposition 4.5 (α_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 (Bool). $\gamma_S(\text{Bool}_g) = \{ \text{Bool}_\ell \mid \ell \in \gamma(g) \}$

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

Case (U). $\gamma_S(U_1 \rightarrow_g U_2) = \gamma_S(U_1) \rightarrow_{(g,c)} \gamma_S(U_2)$.

So $\hat{S} = \{ \ell \rightarrow \ell, S_{11} \}$, with $\{ S_{11} \} \subseteq \gamma_S(U_1)$, $\{ S_{11} \} \subseteq \gamma_S(U_2)$, $\{ \ell \rightarrow \ell \} \subseteq \gamma(g,c)$ and $\{ \ell \rightarrow \ell \} \subseteq \gamma(g)$. By induction hypothesis, $\alpha_S(\{ S_{11} \}) \subseteq U_1$ and $\alpha_S(\{ S_{12} \}) \subseteq U_2$, and by optimality of $\alpha$, $\alpha(\{ \ell \rightarrow \ell \}) \subseteq g,c$ and $\alpha(\{ \ell \rightarrow \ell \}) \subseteq g$. Hence $\alpha_S(\{ \ell \rightarrow \ell, S_{11} \}) = \alpha_S(\{ S_{11} \}) \rightarrow_{(g,c)} \alpha_S(\{ S_{12} \}) \subseteq U_1 \rightarrow_g U_2$.

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

So $\hat{S} = \{ \text{Ref}_\ell S \mid \ell \in \hat{\ell}, S \in \{ S_{11} \} \}$ for some $\{ S_{11} \} \subseteq \gamma(S(U))$ and some $\hat{\ell} \subseteq \gamma(g)$. By induction hypothesis $\alpha_S(\{ S_{11} \}) \subseteq U$ and by optimality of $\alpha$, $\alpha(\hat{\ell}) \subseteq g$, so $\alpha_S(\{ \text{Ref}_\ell S \mid \ell \in \hat{\ell}, S \in \{ S_{11} \} \}) = \text{Ref}_\alpha(\hat{\ell}) \alpha_S(\{ S_{11} \}) \subseteq \text{Ref}_g U$.

□

**Proposition 2.9 (α_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 \( \tau \) denote SSLRef’s type system. Then for any static language term \( t \in \text{TERM} \), \( \cdot; \Sigma; \ell \vdash \; t : S \) if and only if \( \cdot; \Sigma; \ell \vdash \; t : S \).

**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).**

\[
\begin{align*}
(Px) & \quad x \subseteq x \\
(Pb) & \quad g \subseteq g' \quad b_g \subseteq b_{g'} \\
(Pu) & \quad g \subseteq g' \quad \text{unit}_g \subseteq \text{unit}_{g'} \\
(Pa) & \quad \begin{array}{l}
\frac{t \subseteq t'}{\text{prot}_g(t) \subseteq \text{prot}_{g'}(t')} \\
\frac{t_1 \subseteq t'_1 \quad t_2 \subseteq t'_2}{\text{ref}^U(t) \subseteq \text{ref}^{U'}(t')} \\
\frac{t \subseteq t'}{\text{ref}^U(t) \subseteq \text{ref}^{U'}(t')} \\
\frac{t \subseteq t'}{\text{ref}^U(t) \subseteq \text{ref}^{U'}(t')} \\
\end{array}
\end{align*}
\]

**Lemma 4.9.** If \( \Gamma; \cdot; g_e \vdash \; t : U \) and \( \Gamma \subseteq \Gamma' \), then \( \Gamma'; \cdot; g_e \vdash \; t : U' \) for some \( U \subseteq U' \).

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

**Lemma 4.10.** If \( U_1 \subseteq U_2 \) and \( U_1 \subseteq U'_1 \) and \( U_2 \subseteq U'_2 \) then \( U'_1 \subseteq U'_2 \).

**Proof.** By definition of \( \subseteq \), there exists \( \langle S_1, S_2 \rangle \in \gamma^2(U_1, U_2) \) such that \( S_1 \subseteq S_2 \), \( U_1 \subseteq U'_1 \) and \( U_2 \subseteq U'_2 \) mean that \( \gamma(U_1) \subseteq \gamma(U'_1) \) and \( \gamma(U_2) \subseteq \gamma(U'_2) \), therefore \( \langle S_1, S_2 \rangle \in \gamma^2(U'_1, U'_2) \). \( \square \)

**Lemma 4.11.** If \( g_1 \lor g_2 \subseteq g_3, g_1 \in g'_1, g_2 \in g'_2 \) and \( g_3 \in g'_3 \), then \( g'_1 \lor g'_2 \subseteq g'_3 \).

**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 \( \ell_1 \lor \ell_2 \leq \ell_3, g_1 \in g'_1, g_2 \in g'_2 \) and \( g_3 \in g'_3 \) mean that \( \gamma(g_1) \subseteq \gamma(g'_1), \gamma(g_2) \subseteq \gamma(g'_2) \) and \( \gamma(g_3) \subseteq \gamma(g'_3) \) respectively. Therefore \( \langle \ell_1, \ell_2, \ell_3 \rangle \in \gamma^3(g_1, g_2, g_3) \). \( \square \)

**Lemma 4.12.** If \( g_1 \preceq g_2, g_1 \in g'_1 \) and \( g_2 \in g'_2 \), then \( g'_1 \preceq g'_2 \).

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

**Proposition 4.13 (Static Gradual Guarantee).** Suppose \( g_{c1} \subseteq g_{c2} \) and \( t_1 \subseteq t_2 \).
If \( \cdot; g_{c1} \vdash t_1 : U_1 \) then \( \cdot; g_{c2} \vdash t_2 : U_2 \) where \( U_1 \subseteq U_2 \).

**Proof.** We prove the property on opens terms instead of closed terms: If \( \Gamma; \cdot; g_{c1} \vdash t_1 : U_1 \), \( g_{c1} \subseteq g_{c2} \) and \( t_1 \subseteq t_2 \) then \( \Gamma; \cdot; g_{c2} \vdash t_2 : U_2 \) and \( U_1 \subseteq U_2 \).

The proof proceed by induction on the typing derivation.

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

$$
\begin{align*}
\Gamma, x : U_1'; \vdash g' ; t : U_1'' & \\
\Gamma; g c_1 \vdash (\lambda g' x : U_1'.t) g : U_1' \xrightarrow{g} U_1'
\end{align*}
$$

(1)

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 $t_2 = (\lambda g' x : U_1''.t') g'$ and therefore

$$
\begin{align*}
t \sqsubseteq t' & \\
\Gamma ; g c_2 \vdash (\lambda g' x : U_1''.t') g' : U_1'' \xrightarrow{g'} U_2''
\end{align*}
$$

(2)

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

$$
\begin{align*}
\Gamma, x : U_1''; \vdash g c'' ; t' : U_2'' & \\
\Gamma; g c_1 \vdash (\lambda g' x : U_1'.t') g' : U_1'' \xrightarrow{g'} U_2''
\end{align*}
$$

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

$$
U_1' \xrightarrow{g} U_2' \sqsubseteq U_1'' \xrightarrow{g'} U_2''
$$

and the result holds.

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

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

$$
\begin{align*}
\Gamma; g c_1 \vdash t : U & \\
\Gamma; g c_1 \vdash \text{prot}_g (t) : U \sim g
\end{align*}
$$

(3)

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 $t_2 = \text{prot}_g (t')$ and therefore

$$
\begin{align*}
t \sqsubseteq t' & \\
\text{prot}_g (t) \sqsubseteq \text{prot}_g (t')
\end{align*}
$$

(4)

By definition of join on consistent labels, $g c_1 \sim g \sqsubseteq g c_2 \sim g'$. Using induction hypotheses on the premises of 3, we can use rule $(Uprot)$ to derive:

$$
\begin{align*}
\Gamma; g c_2 \vdash t' : U' & \\
\Gamma; g c_2 \vdash \text{prot}_g (t') : U' \sim g'
\end{align*}
$$

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

$$
U \sim g \sqsubseteq U' \sim g'
$$

and the result holds.

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

$$
\begin{align*}
\Gamma; g c_1 \vdash t_1' : \text{Bool}_{g_1} & \\
\Gamma; g c_1 \vdash t_2' : \text{Bool}_{g_2} & \\
\Gamma; g c_1 \vdash t_1' \oplus t_2' : \text{Bool}_{(g_1 \sim g_2)}
\end{align*}
$$

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

$$
(P\oplus) \quad \frac{t_1' \sqsubseteq t_1'' \quad t_2' \sqsubseteq t_2''}{t_1' \oplus t_2' \sqsubseteq t_1'' \oplus t_2''}
$$

(6)

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

$$
(U\oplus) \quad \frac{\Gamma; \vdash g_{c2} \triangleright t_1'' : \text{Bool}_{g_1'} \quad \Gamma; \vdash g_{c2} \triangleright t_2'' : \text{Bool}_{g_2'\lor g_2''}}{
\Gamma; \vdash g_{c2} \triangleright t_1'' \oplus t_2'' : \text{Bool}_{(g_1'\lor g_1'') \lor (g_2'\lor g_2'')}}
$$

Where $g_1' \sqsubseteq g_1''$ and $g_2' \sqsubseteq g_2''$. Using the premise of 6 and the definition of type precision we can infer that

$$
\frac{(g_1' \lor g_1'') \sqsubseteq (g_2' \lor g_2'')}{(g_1' \lor g_1'') \sqsubseteq \text{Bool}_{(g_2' \lor g_2'')}}
$$

and the result holds.

Case (Uapp). Then $t_1 = t_1' t_2'$ and $U_1 = U_{12} \triangleright g$. By (Uapp) we know that:

$$
(U\text{app}) \quad \frac{\Gamma; ; g_{c1} \triangleright t_1'' : U_{11} \xrightarrow{g} g_{12} \quad \Gamma; ; g_{c1} \triangleright t_2'' : U_{22} \quad U_{2}' \leq U_{11}}{
\Gamma; ; g_{c1} \triangleright t_1' t_2' : U_{12} \triangleright g}
$$

(7)

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

$$
(P\text{app}) \quad \frac{t_1' \sqsubseteq t_1'' \quad t_2' \sqsubseteq t_2''}{t_1' t_2' \sqsubseteq t_1'' t_2''}
$$

(8)

Using induction hypotheses on the premises of 7, $\Gamma; \vdash g_{c2} \triangleright t_1'' : U_{11} \xrightarrow{g} g_{12}$ and $\Gamma; \vdash g_{c2} \triangleright t_2'' : U_{22}$, where $U_{2}' \sqsubseteq U_{2}''$, $U_{11} \xrightarrow{g} g_{12} \sqsubseteq U_{11} \xrightarrow{g} g_{12}$. By Lemma 4.10, $U_{2}' \leq U_{11}$. By definition of precision of types, $g_1' \sqsubseteq g_1''$ and $g \sqsubseteq g'$, therefore by Lemma 4.11, $g' \lor g_{c2} \leq g''_{c2}$. Then we can use rule (Uapp) to derive:

$$
(U\text{app}) \quad \frac{\Gamma; ; g_{c2} \triangleright t_1'' : U_{11} \xrightarrow{g} g_{12} \quad \Gamma; ; g_{c2} \triangleright t_2'' : U_{22} \quad U_{2}'' \leq U_{11}}{
\Gamma; ; g_{c2} \triangleright t_1' t_2' : U_{12} \triangleright g'}
$$

Using the definition of type precision we can infer that

$$
U_{12} \triangleright g \sqsubseteq U_{12} \triangleright g'
$$

and the result holds.

Case (Uif). Then $t_1 = \text{if } t \text{ then } t_1' \text{ else } t_2$ and $U_1 = (U_1' \triangleright U_1'') \triangleright g$. By (Uif) we know that:

$$
(U\text{if}) \quad \frac{\Gamma; \vdash g_{c1} \triangleright t : \text{Bool}_{g}}{
\Gamma; ; g_{c1} \triangleright g \triangleright t_1' \triangleright U_1' \quad \Gamma; ; g_{c1} \triangleright g \triangleright t_2 \triangleright U_2'}
$$

(9)

Consider $g_{c2}$ such that $g_{c1} \sqsubseteq g_{c2}$ and $t_2$ such that $t_1 \sqsubseteq t_2$. 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

$$
(P\text{if}) \quad \frac{t \sqsubseteq t' \quad t_1' \sqsubseteq t_1'' \quad t_2' \sqsubseteq t_2''}{\text{if } t \text{ then } t_1' \text{ else } t_2' \sqsubseteq \text{if } t' \text{ then } t_1'' \text{ else } t_2''}
$$

(10)
Consider any $t'$ such that $t \subseteq t'$. As $g_{c_1} \triangleright g \subseteq g_{c_2} \triangleright g'$ then we can use induction hypotheses on the premises of 9 and derive:

\[
\begin{align*}
(\text{Uif}) & \quad \Gamma;::;g_{c_2} + t': \text{Bool}\_g' \\
& \quad \Gamma;::;g_{c_2} \triangleright g' + t'_i : U'_i \quad \Gamma;::;g_{c_2} \triangleright g' + t''_i : U''_i \\
& \quad \Gamma;::;g_{c_2} + \text{if } t' \text{ then } t'_i \text{ else } t''_i : (U'_i \triangleright U''_i) \triangleright g' 
\end{align*}
\]

Where $U'_i \subseteq U''_i$ and $U'_2 \subseteq U''_2$. Using the definition of type precision we can infer that

\[
(U'_i \triangleright U''_i) \triangleright g \supseteq (U'_2 \triangleright U''_2) \triangleright g'
\]

and the result holds.

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

\[
\begin{align*}
(\text{U::}) & \quad \Gamma;::;g_{c_1} + t : U'_1 \quad U'_1 \subseteq U_1 \\
& \quad \Gamma;::;g_{c_1} + t :: U'_1 : U_1 
\end{align*}
\]

(11)

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

\[
(\text{P::}) \quad t \subseteq t' \quad U_1 \subseteq U_2 \\
\quad t :: U_1 \subseteq t' :: U_2 
\]

(12)

Using induction hypotheses on the premises of 11, $\Gamma;::;g_{c} + t' : U'_2$ where $U'_1 \subseteq U'_2$. We can use rule (U::) and Lemma 4.10 to derive:

\[
\begin{align*}
(\text{U::}) & \quad \Gamma;::;g_{c_2} + t' : U'_2 \quad U'_2 \subseteq U_2 \\
& \quad \Gamma;::;g_{c_2} + t' :: U'_2 : U_2 
\end{align*}
\]

Where $U_1 \subseteq U_2$ and the result holds.

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

\[
\begin{align*}
(\text{Uref}) & \quad \Gamma;::;g_{c_1} + t :: U'_1 \quad U'_1 \subseteq U_1 \quad g_{c_1} \triangleright \text{label}(U) \\
& \quad \Gamma;::;g_{c_1} + \text{ref}^{U} t : \text{Ref}_{\bot} U 
\end{align*}
\]

(13)

Consider $g_{c_2}$ such that $g_{c_1} \subseteq g_{c_2}$ and $t_2$ such that $t_1 \subseteq t_2$. By definition of term precision $t_2$ must have the form $t_2 = \text{ref}^{U'} t'$ and therefore

\[
(\text{Pref}) \quad t \subseteq t' \quad U \subseteq U' \\
\quad \text{ref}^{U} t \subseteq \text{ref}^{U'} t' 
\]

(14)

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

\[
\begin{align*}
(\text{Uref}) & \quad \Gamma;::;g_{c_2} + t' : U''_1 \quad U''_1 \subseteq U' \quad g_{c_2} \triangleright \text{label}(U') \\
& \quad \Gamma;::;g_{c_2} + \text{ref}^{U'} t' : \text{Ref}_{\bot} U' 
\end{align*}
\]

Where $U \subseteq U'$ and $U'_1 \subseteq U''_1$. Using the the definition of type precision we can infer that

\[
U \subseteq U' \\
\text{Ref}_{\bot} U \subseteq \text{Ref}_{\bot} U'
\]

and the result holds.

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

\[
\begin{align*}
(\text{Uderefer}) & \quad \Gamma;::;g_{c_1} + t : \text{Ref}_{g} U \\
& \quad \Gamma;::;g_{c_1} + !t : U \triangleright g 
\end{align*}
\]

(15)
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 \( t_2 = !t' \) and therefore

\[
(P_{deref}) \quad \frac{t \sqsubseteq t'}{!t \sqsubseteq !t'}
\]

(16)

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

\[
(U_{deref}) \quad \frac{\Gamma; : g_{c_2} \vdash t' : \text{Ref}_g U'}{\Gamma; : g_{c_2} \vdash !t' : U' \triangleright \triangleright g'}
\]

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

\[
U \triangleright \triangleright g \sqsubseteq U' \triangleright \triangleright g'
\]

and the result holds.

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

\[
(U_{asgn}) \quad \frac{\Gamma; : g_{c_1} \vdash t'_1 : \text{Ref}_g U'_1 \quad \Gamma; : g_{c_1} \vdash t'_2 : U'_2}{U'_2 \sqsubseteq U'_1 \quad g \triangleright \triangleright g_{c_1} \sqsubseteq \text{label}(U'_1)}
\]

(17)

Using induction hypotheses on the premises of 17, \( \Gamma; : g_{c_2} \vdash t'_1 : \text{Ref}_g U'_1'' \) and \( \Gamma; : g_{c_2} \vdash t'_2 : U'_2'' \), where \( \text{Ref}_g U'_1' \sqsubseteq \text{Ref}_g U'_1'' \) and \( U'_2' \sqsubseteq U'_2'' \). By definition of precision on types and Lemma 4.10, \( U'_2'' \sqsubseteq U'_2'' \). Also, as, \( g \sqsubseteq g' \) and \( U'_1' \sqsubseteq U'_1'' \), by Lemma 4.11, \( g' \triangleright \triangleright g_{c_2} \sqsubseteq \text{label}(U'_1) \). Then we can use rule \((U_{asgn})\) to derive:

\[
(U_{asgn}) \quad \frac{U'_2'' \sqsubseteq U'_1'' \quad g' \triangleright \triangleright g_{c_2} \sqsubseteq \text{label}(U'_1'')}{\Gamma; : g_{c_2} \vdash t''_1 := t''_2 : \text{Unit}_\bot}
\]

(18)

Using the definition of type precision we can infer that

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

and the result holds.

\[\square\]
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.

\[ i \in \text{Label}^2 \]

\[ i ::= [\ell, \ell] \quad \text{(interval)} \]

Definition 5.2 (Interval Concretization). Let \(\gamma_i : \text{Label}^2 \to \mathcal{P}(\text{Label})\) be defined as follows:

\[ \gamma_i([\ell_1, \ell_2]) = \{ \ell \mid \ell \in \text{Label}, \ell_1 \preceq \ell \preceq \ell_2 \} \]

We can only concretize valid intervals:

Definition 5.3 (Valid Gradual Label). \(\ell_1 \preceq \ell_2\) valid \([\ell_1, \ell_2]\)

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

\[ \gamma_\varepsilon(\langle \ell_1, \ell_2 \rangle) = \{ \langle \ell_1, \ell_2 \rangle \mid \ell_1 \in \gamma_i(\ell_1), \ell_2 \in \gamma_i(\ell_2) \} \]

Definition 5.5 (Interval Abstraction). Let \(\alpha_i : \mathcal{P}(\text{Label}) \to \text{Label}^2\) be defined as follows:

\[ \alpha_i(\emptyset) \text{ is undefined} \]

\[ \alpha_i([\ell_i]) = [\bigwedge_i, \bigvee_i] \text{ otherwise} \]

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

\[ \alpha_\varepsilon(\emptyset) \text{ is undefined} \]

\[ \alpha_\varepsilon(\{ \langle \ell_1, \ell_2 \rangle \}) = \langle \alpha_i(\{ \ell_1 \}), \alpha_i(\{ \ell_2 \}) \rangle \text{ otherwise} \]

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

\[ E \in \text{GEType}, \quad i \in \text{Label}^2 \]

\[ E ::= \text{Bool}, \mid E \to_i E \mid \text{Ref}, E \mid \text{Unit}, \quad \text{(evidence types)} \]

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

\[ \gamma_E(\text{Bool}) = \{ \text{Bool}_i \mid \ell \in \gamma_i(\ell) \} \]

\[ \gamma_E(E_1 \to_i E_2) = \gamma_E(E_1) \to_i \gamma_E(E_2) \]

\[ \gamma_E(\text{Ref}_i E) = \{ \text{Ref}_i S \mid \ell \in \gamma_i(\ell), S \in \gamma_E(E) \} \]

where \(\to\) is the set of all possible combinations of function types, using each member of the sets obtained by the \(\gamma_E\) and \(\gamma_i\) functions.
Definition 5.9 (Evidence Concretization). Let 
\( \gamma_{\ell_1} : \text{GETYPE}^2 \rightarrow \wp(\text{TYPE}^2) \) be defined as follows:
\[
\gamma_{\ell_1}(\langle E_1, E_2 \rangle) = \{ \langle S_1, S_2 \rangle \mid S_1 \in \gamma_E(E_1), S_2 \in \gamma_E(E_2) \}
\]

Definition 5.10 (Type Evidence Abstraction). Let the abstraction function \( \alpha_E : \wp(\text{TYPE}) \rightarrow \text{GETYPE} \) be defined as:
\[
\alpha_E(\emptyset) = \text{Bool}_{\omega((T_{\omega}))}
\]
\[
\alpha_E(\langle S_{\ell_1} \rightarrow \ell_1, S_{\ell_2} \rangle) = \alpha_E(\langle S_{\ell_1} \rangle) \rightarrow_{\alpha_E((T_{\omega}))} \alpha_E(\langle S_{\ell_2} \rangle)
\]
\[
\alpha_E(\langle \text{Ref}_{\ell_1}, S_{\ell_1} \rangle) = \text{Ref}_{\alpha_E((T_{\omega}))} \alpha_E(\langle S_{\ell_1} \rangle)
\]
\[
\alpha_E(\hat{S}) \text{ is undefined otherwise}
\]

Definition 5.11 (Evidence Abstraction). Let \( \alpha_\ell : \wp(\text{TYPE}^2) \rightarrow \text{GETYPE}^2 \) be defined as follows:
\[
\alpha_\ell(\emptyset) = \langle \alpha_E(\langle S_{\ell_1} \rangle), \alpha_E(\langle S_{\ell_2} \rangle) \rangle
\]
\[
\alpha_\ell(\langle S_{\ell_1}, S_{\ell_2} \rangle) \text{ otherwise}
\]

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).
\[
\begin{align*}
\text{valid}(\{ \text{Bool}_{\ell_1} \}) & \quad \text{valid}(\langle S_{\ell_1} \rangle) \quad \text{valid}(\langle S_{\ell_2} \rangle) \quad \text{valid}(\{ \text{Unit}_{\ell_1} \}) \\
\text{valid}(\langle S_{\ell_1} \rangle) & \quad \text{valid}(\langle S_{\ell_2} \rangle) \\
\text{valid}(\langle S_{\ell_1} \rangle) & \quad \text{valid}(\langle S_{\ell_2} \rangle) \\
\end{align*}
\]

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

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

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 \( \gamma_{\ell_1}(t_1) \subseteq \gamma_{\ell_1}(t_2) \); inductively:
\[
\ell_3 \ll \ell_1 \quad \ell_2 \ll \ell_4 \quad [\ell_1, \ell_2] \ll [\ell_3, \ell_4]
\]

(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:
\[
\begin{align*}
\text{bool}_{t_1} \subseteq \text{bool}_{t_2} \quad E_{t_1} \subseteq E_{t_2} \quad E_{t_1} \triangleleft E_{t_2} \quad E_{t_1} \leftrightarrow E_{t_2} \quad \text{ref}_{t_1} \subseteq \text{ref}_{t_2} \quad E_{t_1} \triangleleft E_{t_2}
\end{align*}
\]
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 $F_1(\bar{g}_i) \triangleq F_2(\bar{g}_j)$, notation $\mathcal{J}[F_1(\bar{g}_i) \triangleq F_2(\bar{g}_j)]$, is defined as follows:

$\mathcal{J}[F_1(g_1, \ldots, g_n) \triangleq F_2(g_{n+1}, \ldots, g_{n+m})] = \alpha_{\ell}((\{F_1(\bar{\ell}_i), F_2(\bar{\ell}_j)\} | \langle \bar{\ell}_i \rangle \in \gamma^n(\bar{g}_i[1/n]), \langle \bar{\ell}_j \rangle \in \gamma^m(\bar{g}_j[n+1/m]) | F_1(\bar{\ell}_i) \triangleq F_2(\bar{\ell}_j)))$

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 $F_1(\bar{U}_i) \triangleq F_2(\bar{U}_j)$, notation $\mathcal{J}[F_1(\bar{U}_i) \triangleq F_2(\bar{U}_j)]$, is defined as follows:

$\mathcal{J}[F_1(U_1, \ldots, U_n) \triangleq F_2(U_{n+1}, \ldots, U_{n+m})] = \alpha_{\ell}((\{F_1(\bar{S}_i), F_2(\bar{S}_j)\} | \langle \bar{S}_i \rangle \in \gamma^n(\bar{U}_i[1/n]), \langle \bar{S}_j \rangle \in \gamma^m(\bar{U}_j[n+1/m]) | F_1(\bar{S}_i) \triangleq F_2(\bar{S}_j)))$

**Proposition 5.20.** [Elaboration preserves typing] Consider $\Gamma; \Sigma; g_c \vdash t : U$ then if $\Gamma; \Sigma; g_c \vdash t \sim t' : U$, and $\varepsilon = \delta^U(\ell_c)$, then $\Gamma; \Sigma; \varepsilon g_c \vdash t' : U$

**Proof.** Straightforward induction on type $U$. □

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 $\circ^{<}$ 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:

$\langle t_{11}, t_{12} \rangle \circ^{\leq} \langle t_{21}, t_{22} \rangle = \alpha_{\ell}((\langle \ell_{11}, \ell_{22} \rangle \in \gamma_{\ell}(\langle t_{11}, t_{12} \rangle) | \exists \ell \in \gamma_{t}(t_{12}) \land \gamma_{t}(t_{21}) \land \ell_{11} \leq \ell \land \ell \leq \ell_{22}))$

**Proposition 5.22.** Suppose $\varepsilon_1 \vdash F_1(\bar{g}_i) \triangleq F_2(\bar{g}_j)$ and $\varepsilon_2 \vdash F_2(\bar{g}_j) \triangleq F_3(\bar{g}_k)$

If $\varepsilon_1 \circ^{\leq} \varepsilon_2$ is defined, then $\varepsilon_1 \circ^{\leq} \varepsilon_2 \vdash F_1(\bar{g}_i) \triangleq F_3(\bar{g}_k)$

**Proposition 5.23.** $\gamma_{t}(t_{1} \cap t_{2}) = \gamma_{t}(t_{1}) \cap \gamma_{t}(t_{2})$

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

**Proposition 5.24.** $\langle t_{1}, t_{2} \rangle \circ^{<} \langle t_{22}, t_{3} \rangle = \Delta^{<}(t_{1}, t_{22} \cap t_{22}, t_{3})$

where $\Delta^{<}(t_{1}, t_{2}, t_{3}) = \alpha_{\ell}((\langle \ell_{1}, \ell_{3} \rangle \in \gamma_{\ell}(\langle t_{1}, t_{3} \rangle) | \exists \ell_{2} \in \gamma_{t}(t_{3}) \land \ell_{1} \leq \ell_{2} \land \ell_{2} \leq \ell_{3}))$
Definition 5.25 (Consistent transitivity for subtyping). Suppose

\[ [E_{11}, E_{12}] \vdash F_1(U_1) <: F_2(U_2) \quad \text{and} \quad [E_{21}, E_{22}] \vdash F_2(U_2) <: F_3(U_3) \]

We deduce evidence for consistent transitivity for subtyping:

\[ [E_{11}, E_{12}] \circ <: [E_{21}, E_{22}] \vdash F_1(U_1) <: F_3(U_3) \]

where \( \circ <: : \text{ETYPE}^2 \times \text{ETYPE}^2 \rightarrow \text{ETYPE}^2 \) is defined as:

\[ [E_{11}, E_{12}] \circ <: [E_{21}, E_{22}] = \alpha_y([\{S_{11}, S_{22}\} \in \gamma_y([E_{11}, E_{22}]) | \exists S_{2} \in \gamma_y(E_{21}).S_{1} :: S_{2} \wedge S_{2} < : S_{22}]) \]

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.

\[ [E_{11}, E_{12}] \circ <: [E_{22}, E_{3}] = \Delta <: (E_{11}, E_{21} \cap E_{22}, E_{3}) \]

where

\[ \Delta <: (E_{11}, E_{22}, E_{3}) = \alpha_y([\{S_{1}, S_{3}\} \in \gamma_y([E_{1}, E_{3}]) | \exists S_{2} \in \gamma_y(E_{2}).S_{1} :: S_{2} \wedge S_{2} < : S_{3}]) \]

Definition 5.28 (Intervals join).

\[ [\ell_1, \ell_2] \sqcup [\ell_3, \ell_4] = [\ell_1 \vee \ell_3, \ell_2 \vee \ell_4] \]

Definition 5.29 (Evidence label join).

\[ [t_1, \ell_2] \sqcup [t_3, t_4] = [\ell_1 \vee t_3, \ell_2 \vee t_4] \]

Definition 5.30.

\[ \text{Bool}_{t_1 \vee t_2} = \text{Bool}_{\ell_1 \vee \ell_2} \]

\[ E_{1} \overset{t_2}{\rightarrow}_{t_1} E_{2} \overset{\ell_1 \wedge \ell_2}{\rightarrow} E_{1} \overset{t_2}{\rightarrow}_{t_1 \vee t_3} E_{2} \]

\[ \text{Ref}_{t_1} E \overset{\ell_1 \vee t_2}{\rightarrow} E \]

Definition 5.31.

\[ [E_{1}, E_{2}] \overset{\ell_1 \vee t_2}{\rightarrow} [t_3, t_4] = [E_{1} \overset{\ell_1 \vee t_1, E_{2} \overset{\ell_1 \vee t_2}{\rightarrow}}{\rightarrow} t_3, t_4] \]

Proposition 5.32. If \( \epsilon_S \vdash U_1 \leq U_2 \) and \( \epsilon_1 \vdash g_1 \leq g_2 \) then \( \epsilon_S \overset{\ell_1 \vee t_1}{\rightarrow} U_1 \overset{\ell_1 \vee t_2}{\rightarrow} g_1 \vdash U_2 \overset{\ell_1 \vee t_2}{\rightarrow} g_2 \)

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).

\[ [\ell_1, \ell_2] \sqcup [\ell_3, \ell_4] = [\ell_1 \vee \ell_3, \ell_2 \vee \ell_4] \]

Definition 5.34 (Intervals meet).

\[ [\ell_1, \ell_2] \sqcap [\ell_3, \ell_4] = [\ell_1 \wedge \ell_3, \ell_2 \wedge \ell_4] \]
Definition 5.35. Let $F_1 : \text{GLABEL}^n \rightarrow \text{GLABEL}$ and $F_2 : \text{GLABEL}^m \rightarrow \text{GLABEL}$. The initial evidence for consistent judgment $F_1(\overline{g}) \preceq F_2(\overline{g})$ is defined as follows:

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

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

where $F_1 : \text{GLABEL}^n \rightarrow \text{GLABEL}$ and $F_2 : \text{GLABEL}^m \rightarrow \text{GLABEL}$.

\[
\mathcal{G}(\cup)(F(g_1, \ldots, g_n)) = \mathcal{G}(F(g_1, \ldots, g_n) \preceq F(g_1, \ldots, g_n))
\]

The algorithmic definition of meet:

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

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

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

\[
\Delta^\preceq([\ell_1, \ell_2], [\ell_3, \ell_4], [\ell_5, \ell_6]) = \langle [\ell_1, \ell_2 \land \ell_4 \land \ell_6], [\ell_1 \lor \ell_3 \lor \ell_5, \ell_6] \rangle
\]

5.4.2 Type Evidences. We define a function $\text{lift}()$ 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).

\[
P'^T \in \text{GPATTERN}, P'^{\ell} \in \text{LPATTERN}
\]

\[
P'^T ::= _ | P'^T \text{ op'}^T P'^T \quad \text{(pattern on types)}
\]

\[
op'^T ::= \lor | \land | \rightarrow \quad \text{(operations on types)}
\]

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

\[
op'^{\ell} ::= \lor | \land | \rightarrow \quad \text{(operations on labels)}
\]
We use case-based analysis to calculate the algorithmic rules for the initial evidence of consistent subtyping on gradual security types:

\[
\begin{align*}
\mathcal{G}[[\text{liftP}(G_1)(\overline{t_1})] &\triangleleft \text{liftP}(G_2)(\overline{t_2})] = \langle t_1, t_2 \rangle \\
\mathcal{G}[[\text{GLabel}_i \triangleleft \text{GLabel}_j] &\triangleleft \langle \text{Bool}_i, \text{Bool}_j \rangle = \langle \text{Bool}_i, \text{Bool}_j \rangle \\
\mathcal{G}[[\text{invert}(G_2)(\overline{U_j})] &\triangleleft \text{invert}(G_1)(\overline{U_i})] = \langle E_{11}, E_{12} \rangle \\
\mathcal{G}[[\text{liftP}(G_1)(\overline{t_1})] &\triangleleft \text{liftP}(G_2)(\overline{t_2})] = \langle t_1, t_2 \rangle \\
\mathcal{G}[[\text{tomeet}(G_1)(\overline{U_i})] &\triangleleft \text{tomeet}(G_2)(\overline{U_j})] = \langle E_i, E_j \rangle \\
\mathcal{G}[[\text{GLabel}_i \triangleleft \text{GLabel}_j] &\triangleleft \langle \text{Ref}_i, \text{Ref}_j \rangle = \langle \text{Ref}_i, \text{Ref}_j \rangle,
\end{align*}
\]

where \(G_1 : \text{GLabel}^n \rightarrow \text{GLabel} \) and \(G_2 : \text{GLabel}^m \rightarrow \text{GLabel} \), and \(G_1(x_1, ..., x_n) = P^T_1(x_1, ..., x_n), \) \(G_2(x_1, ..., x_n) = P^T_2(x_1, ..., x_m). \)

We calculate a recursive meet operator for gradual types:

\[
\text{Bool}_i \triangleleft \text{Bool}_j = \text{Bool}_{i \cap j}
\]

\[
(E_{11} \rightarrow_{E_{12}} \rightarrow_{E_{21}} \rightarrow_{E_{22}}) \triangleleft \langle E_{11} \cap E_{21} \rightarrow_{E_{12} \cap E_{22}} \rangle
\]

\[
\text{Ref}_i \triangleleft \text{Ref}_j = \text{Ref}_{i \cap j}
\]

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

We calculate a recursive definition for \(\triangleleft^<=\) by case analysis on the structure of the second argument,
5.4.3 Evidence inversion functions. The evidence inversion functions are defined as follows

\[
\text{idom}(\langle E_1 \xrightarrow{t_1} E_2, E'_1 \xrightarrow{t'_1} E'_2 \rangle) = \langle E_1, E'_1 \rangle
\]
\[
\text{idom}(\langle E_1, E_2 \rangle) = \text{undefined otherwise}
\]
\[
\text{icod}(\langle E_1 \xrightarrow{t_1} E_2, E'_1 \xrightarrow{t'_1} E'_2 \rangle) = \langle E_2, E'_2 \rangle
\]
\[
\text{icod}(\langle E_1, E_2 \rangle) = \text{undefined otherwise}
\]

5.5 Proofs

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

PROOF. Suppose \(\hat{\ell} = \{ \overline{\ell} \}\). By definition of \(\alpha_\ell\), \(\alpha_\ell(\{ \overline{\ell} \}) = [\overline{\ell}, \overline{\ell}]\). Therefore

\[
\gamma_\ell(\alpha_\ell(\{ \overline{\ell} \})) = \{ \ell \mid \ell \in \text{LABEL}, \overline{\ell} \leq \ell < \overline{\ell} \}
\]

And it is easy to see that if \(\ell \in \{ \overline{\ell} \}\), then \(\ell \in \gamma_\ell(\alpha_\ell(\{ \overline{\ell} \}))\), and therefore the result holds. \(\square\)

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

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

PROPOSITION 5.15 (\(\alpha_E\) is Sound). If \(\text{valid}(\hat{S})\) then \(\hat{S} \subseteq \gamma_E(\alpha_E(\hat{S}))\).
Therefore, by definition of intersection on intervals, \( \ell \) is equivalent to \( \{ \ell \mid \ell \in \text{LABEL}, \ell_1 \preceq \ell \preceq \ell_2 \} \cap \{ \ell \mid \ell \in \text{LABEL}, \ell_3 \preceq \ell \preceq \ell_4 \} \). Which is equivalent to \( \{ \ell \mid \ell \in \text{LABEL}, \ell_1 \preceq \ell \preceq \ell_2 \wedge \ell_3 \preceq \ell \preceq \ell_4 \} \), which is defined. Suppose \( t_1 = [\ell_1, \ell_2] \) and \( t_2 = [\ell_3, \ell_4] \). Then \( t_1 \cap t_2 = [\ell_1 \lor \ell_3, \ell_2 \land \ell_4] \).

By induction on evidence types \( \varepsilon_1 \) and \( \varepsilon_2 \) and Prop 5.23.

**Proposition 5.24.** \( \langle t_1, t_2 \rangle \circ \triangleleft \langle t_{22}, t_3 \rangle = \Delta \triangleleft \langle t_{122}, t_3 \rangle \)

**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 the evidence types \( \varepsilon_1 \) and \( \varepsilon_2 \) and Prop 5.23.

**Proposition 5.27.**

\[
(E_1, E_2) \circ \triangleleft (E_{22}, E_3) = \Delta \triangleleft (E_{122}, E_3)
\]

where

\[
\Delta \triangleleft (E_1, E_2, E_3) = \alpha_{\gamma}(\{ \langle S_1, S_3 \rangle \in \gamma((E_1, E_3)) \mid \exists S_2 \in \gamma((E_2), S_1 <: S_2 \wedge S_2 <: S_3) \})
\]

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

**Proposition 5.32.** If \( \varepsilon_S \vdash U_1 \leq U_2 \) and \( \varepsilon_1 \vdash g_1 \preceq g_2 \) then \( \varepsilon_S \triangledown \varepsilon_1 \triangledown U_1 \triangledown g_1 <: U_2 \triangledown g_2 \)

**Proof.** By induction on types \( U_1 \) and \( U_2 \), using the definition of \( S \preceq \); and Proposition 5.43.

**Proposition 5.37.** \( (\ell_1, \ell_2) \triangledown (\ell_3, \ell_4) = (\ell_1 \lor \ell_3, \ell_2 \lor \ell_4) \)

**Proof.** Follows directly by definition of \( \gamma \) and \( \triangledown \).

**Proposition 5.38.**

\( \langle t_1, t_2 \rangle \triangledown \langle t_1', t_2' \rangle = \langle t_1 \lor t_1', t_2 \lor t_2' \rangle \)

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

**Proposition 5.39.**

\[
[t_1, t_2] \cap [t_3, t_4] = [t_1 \lor t_3, t_2 \land t_4] \quad \text{if } t_1 \lor t_3 \preceq t_2 \land t_4
\]

\( t \cap t' \) undefined otherwise

**Proof.** By definition of meet:

\[
[t_1, t_2] \cap [t_3, t_4] = \alpha_{\gamma}([t' \mid t' \in \gamma([t_1, t_2]) \cap \gamma([t_3, t_4])])
\]

But by definition of intersection on intervals, \( \gamma([t_1, t_2]) \cap \gamma([t_3, t_4]) = \gamma([t_1 \lor t_3, t_2 \land t_4]) \) if \( t_1 \lor t_3 \preceq t_2 \land t_4 \) (otherwise the intersection is empty), and the result follows by definition of \( \alpha \).
Proposition 5.40.  
\[ \ell_1 \leq \ell_4 \quad \ell_3 \leq \ell_6 \quad \ell_1 \leq \ell_6 \]  
\[ \Delta^K([\ell_1, \ell_2], [\ell_3, \ell_4], [\ell_5, \ell_6]) = \langle \ell_1, \ell_2 \land \ell_4 \land \ell_6, [\ell_1 \lor \ell_3 \lor \ell_5, \ell_6] \rangle \]

Proof. By definition:
\[ \Delta^K([\ell_1, \ell_2], [\ell_3, \ell_4], [\ell_5, \ell_6]) = \alpha_i \{ ((\ell'_1, \ell'_2) \in \gamma_i \langle [\ell_1, \ell_2], [\ell_3, \ell_6] \rangle) \mid \exists \ell'_2 \in \gamma_i \langle [\ell_3, \ell_4], \ell'_1 \leq \ell'_2 \leq \ell'_3 \rangle \}
\]
It is easy to see that \( \alpha_i \{ ((\ell'_1, \ell'_2) = [\ell_1, \ell_2], for some \ell'_2 \leq \ell_4 \text{ and } \ell'_2 \leq \ell_6, \text{ i.e. } \ell'_2 \leq \ell_2 \land \ell_4 \land \ell_6 \text{. But } \ell_2 \land \ell_4 \land \ell_6 \leq \ell_4 \text{ and } \ell_6 \text{ therefore}
\]
\[ \langle \ell_2 \land \ell_4 \land \ell_6, \ell_6 \rangle \in \{ \langle \ell_1', \ell_2' \rangle \in \gamma_i \langle [\ell_1, \ell_2], [\ell_3, \ell_6] \rangle \mid
\]
and by definition of \( \alpha_i \), \( \ell_2 \land \ell_4 \land \ell_6 \leq \ell'_2 \), then \( \alpha_i \{ ((\ell'_1, \ell'_2) = [\ell_1, \ell_2 \land \ell_4 \land \ell_6] \text{. Similar argument is used to prove that } \alpha_i \{ ((\ell'_1, \ell'_2) = [\ell_1 \land \ell_3 \land \ell_5, \ell_6] \).

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

Proof.
\[ (\ell_1 \land \ell_2) \lor (\ell_3 \land \ell_4) \leq (\ell_1 \lor (\ell_3 \land \ell_4)) \land (\ell_2 \lor (\ell_3 \land \ell_4)) \]
\[ \leq ((\ell_1 \lor \ell_3) \land (\ell_1 \lor \ell_4)) \land ((\ell_2 \lor \ell_3) \land (\ell_2 \lor \ell_4)) \]
\[ \leq (\ell_1 \lor \ell_3) \land (\ell_2 \lor \ell_4) \]

Proposition 5.42. Suppose \( \varepsilon_1 \vdash F_1(\overline{g_1}) \leq F_2(\overline{g_2}) \) and \( \varepsilon_2 \vdash F_3(\overline{g_3}) \). If \( \varepsilon_1 \preceq \varepsilon_2 \) is defined, then \( \varepsilon_1 \preceq \varepsilon_2 \vdash F_1(\overline{g_1}) \leq F_3(\overline{g_3}) \).

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:
\[ \langle t_{11}, t_{12} \rangle = \langle \ell_1, \ell_2 \rangle, [\ell_3, \ell_4] \rangle \subseteq \gamma \langle [F_1(\overline{g_1}) \leq F_2(\overline{g_2})] \rangle = \langle t'_{11}, t'_{12} \rangle \]
and
\[ \langle t_{21}, t_{22} \rangle = \langle [\ell_5, \ell_6], [\ell_7, \ell_8] \rangle \subseteq \gamma \langle [F_3(\overline{g_3}) \leq f_3(\overline{g_3})] \rangle = \langle t'_{21}, t'_{22} \rangle \]
Suppose that \( \gamma \langle [F_1(\overline{g_1}) \leq F_3(\overline{g_3})] \rangle = \langle t'_1, t'_2 \rangle \). We have to prove that \( \langle t_{11}, t_{12} \rangle \circ \varepsilon \langle t_{21}, t_{22} \rangle \subseteq \langle t'_1, t'_2 \rangle \).

If \( \text{bounds}(F_1(\overline{g_1})) = [\ell'_1, \ell'_2] \), \( \text{bounds}(F_2(\overline{g_2})) = [\ell'_3, \ell'_4] \), and \( \text{bounds}(F_3(\overline{g_3})) = [\ell'_5, \ell'_6] \) We know that \( \gamma \langle [F_1(\overline{g_1}) \leq F_2(\overline{g_2})] \rangle = \langle [\ell'_1, \ell'_2 \land \ell'_4], [\ell'_3 \lor \ell'_6, \ell'_6] \rangle \). Therefore \( \ell'_1 \leq \ell_1, \ell_2 \leq \ell'_2 \land \ell'_4, \ell'_1 \lor \ell'_2 \leq \ell_3 \) and \( \ell_4 \leq \ell'_4 \).

Using the same argument,
\( \gamma \langle [F_2(\overline{g_2}) \leq F_3(\overline{g_3})] \rangle = \langle [\ell'_1, \ell'_2 \land \ell'_4], [\ell'_3 \lor \ell'_6, \ell'_6] \rangle \). Therefore \( \ell'_1 \leq \ell_5, \ell_6 \leq \ell'_4 \land \ell'_6, \ell'_3 \lor \ell'_2 \leq \ell_7 \) and \( \ell_8 \leq \ell'_6 \).

But \( \gamma \langle [F_1(\overline{g_1}) \leq F_3(\overline{g_3})] \rangle = \langle [\ell'_1, \ell'_2 \land \ell'_4], [\ell'_3 \lor \ell'_6, \ell'_6] \rangle \) and
\[ \langle t_{11}, t_{12} \rangle \circ \varepsilon \langle t_{21}, t_{22} \rangle = \Delta^K(t_{11}, t_{12} \cap t_{21}, t_{22}) = \]
\[ \Delta^K([\ell_1, \ell_2], [\ell_3 \lor \ell_5, \ell_4 \land \ell_6], [\ell_7, \ell_8]) = \langle [\ell_1, \ell_2 \land \ell_4 \land \ell_6 \land \ell_8], [\ell_1 \lor \ell_3 \lor \ell_5 \lor \ell_7, \ell_8] \rangle \]
we need to prove that
\[ \langle [\ell_1, \ell_2 \land \ell_4 \land \ell_6 \land \ell_8], [\ell_3 \lor \ell_5 \lor \ell_7, \ell_8] \rangle \subseteq \langle [\ell'_1, \ell'_2 \land \ell'_4 \land \ell'_6], [\ell'_3 \lor \ell'_5, \ell'_8] \rangle \]
. 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. \( [\ell_1, \ell_2 \land \ell_4 \land \ell_6 \land \ell_8] \subseteq [\ell'_1, \ell'_2 \land \ell'_6] \). The argument is applied for the second components and the result holds.

\[ \square \]

**Proposition 5.43.** Suppose \( \varepsilon_1 \vdash F_{11} (\overline{g}) \ll F_{12} (\overline{g}) \) and \( \varepsilon_2 \vdash F_{21} (\overline{g}) \ll F_{22} (\overline{g}) \)
Then \( \varepsilon_1 \lor \varepsilon_2 \vdash F_{11} (\overline{g}) \lor F_{21} (\overline{g}) \ll F_{12} (\overline{g}) \lor F_{22} (\overline{g}) \)

**Proof.** By definition of initial evidence noticing that \( \varepsilon_1 \lor \varepsilon_2 \) can be more precise than the initial evidence of judgment
Suppose \( \varepsilon_1 = \langle [\ell_1, \ell_2], [\ell_3, \ell_4] \rangle \) and \( \varepsilon_2 = \langle [\ell_5, \ell_6], [\ell_7, \ell_8] \rangle \) then \( \varepsilon_1 \lor \varepsilon_2 = \langle [\ell_1 \lor \ell_5, \ell_2 \lor \ell_6], [\ell_3 \lor \ell_6, \ell_4 \lor \ell_8] \rangle \).
If \( \text{bounds}(F_{11} (\overline{g})) = [\ell'_{111}, \ell'_{112}], \text{bounds}(F_{12} (\overline{g})) = [\ell'_{121}, \ell'_{122}] \), \( \text{bounds}(F_{21} (\overline{g})) = [\ell'_{211}, \ell'_{212}] \) and \( \text{bounds}(F_{22} (\overline{g})) = [\ell'_{221}, \ell'_{222}] \).
We know that \( \mathcal{G} [F_{11} (\overline{g}) \ll F_{12} (\overline{g})] = \langle [\ell'_{111}, \ell'_{112} \land \ell'_{122}], [\ell'_{111} \lor \ell'_{121}, \ell'_{112}] \rangle \). Therefore \( \ell'_{111} \ll \ell_1 \), \( \ell_2 \ll \ell'_{121} \land \ell'_{122} \), \( \ell'_{111} \lor \ell'_{121} \ll \ell_3 \) and \( \ell_4 \ll \ell'_{222} \). Using the same argument, \( \mathcal{G} [F_{21} (\overline{g}) \ll F_{22} (\overline{g})] = \langle [\ell'_{211}, \ell'_{212} \land \ell'_{222}], [\ell'_{211} \lor \ell'_{212}, \ell'_{221}] \rangle \). Therefore \( \ell'_{211} \ll \ell_5 \), \( \ell_6 \ll \ell'_{221} \land \ell'_{222} \), \( \ell'_{211} \lor \ell'_{221} \ll \ell_7 \) and \( \ell_8 \ll \ell'_{222} \).

But the \( \mathcal{G} [F'_1 (\overline{g}) \ll F'_2 (\overline{g})] = \langle [\ell'_1, \ell'_2 \land \ell'_4], [\ell'_1 \lor \ell'_3, \ell'_4] \rangle \) where
\( \text{bounds}(F'_1 (\overline{g})) = \text{bounds}(F_{11} (\overline{g})) \lor \text{bounds}(F_{21} (\overline{g})) = [\ell'_{111}, \ell'_{112}] \lor [\ell'_{211}, \ell'_{212}] = [\ell'_{111} \lor \ell'_{211}, \ell'_{112} \lor \ell'_{212}] \).
and
\( \text{bounds}(F'_2 (\overline{g})) = \text{bounds}(F_{12} (\overline{g})) \lor \text{bounds}(F_{22} (\overline{g})) = [\ell'_{121}, \ell'_{122}] \lor [\ell'_{221}, \ell'_{222}] = [\ell'_{121} \lor \ell'_{221}, \ell'_{122} \lor \ell'_{222}] \).
We need to prove that \( [\ell_1 \lor \ell_5, \ell_2 \lor \ell_6] \subseteq [\ell'_{111}, \ell'_{121}, \ell'_{112} \lor \ell'_{122}], \) i.e. \( \ell'_{111} \lor \ell'_{211} \ll \ell_1 \lor \ell_5 \) and \( \ell_2 \lor \ell_6 \ll \ell'_{121} \lor \ell'_{122} \). But \( \ell'_{111} \ll \ell_1 \) and \( \ell'_{211} \ll \ell_5 \), therefore \( \ell'_{111} \lor \ell'_{211} \ll \ell_1 \lor \ell_5 \). Similarly, as \( \ell_2 \ll \ell'_{121} \land \ell'_{122} \) and \( \ell_6 \ll \ell'_{221} \land \ell'_{222} \), then \( \ell_2 \lor \ell_6 \ll \ell'_{121} \lor \ell'_{122} \). Therefore \( [\ell_1 \lor \ell_5, \ell_2 \lor \ell_6] \subseteq [\ell'_{111} \lor \ell'_{211}, \ell'_{112} \lor \ell'_{221}] \).
Using analogous argument, we also know that \( [\ell_3 \lor \ell_6, \ell_4 \lor \ell_8] \subseteq [\ell'_{121} \lor \ell'_{221}, \ell'_{122} \lor \ell'_{222}] \). Therefore \( \varepsilon_1 \lor \varepsilon_2 = \mathcal{G} [F'_1 (\overline{g}) \ll F'_2 (\overline{g})] \), and the result holds.

\[ \square \]

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

(1) If \( (S_1 \lor S_2) \) is defined then \( S_1 \ll (S_1 \lor S_2) \).

(2) If \( (S_1 \land S_2) \) is defined then \( (S_1 \land S_2) \ll S_1 \).

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

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

Case \( (S \rightarrow \ell) \). If \( S_1 = S_{11} \rightarrow_{\ell_1} S_{12} \) then as \( (S_1 \lor S_2) \) is defined then \( S_2 \) must have the form \( S_{21} \rightarrow_{\ell_2} S_{22} \) for some \( S_{21}, S_{22} \) and \( \ell_2 \).
We also know that \((S_1 \triangledown S_2) = (S_1 \land S_{21}) \rightarrow (\ell_1 \lor \ell_2) \rightarrow (S_{12} \land S_{22})\). By definition of \(\preceq\), \(\ell_1 \preceq (\ell_1 \lor \ell_2)\).

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

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

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

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

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

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

### 6 GSL<sup>e</sup>Ref: 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<sup>e</sup>Ref 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 \(T[U]\) of type-indexed sets, such that ill-typed terms do not exist. They are built up from disjoint families \(U[V] \in V[U]\) and \(\sigma[U] \in \Sigma[U]\) of intrinsically typed variables and locations respectively. Unless required, we omit the type exponent on intrinsic terms, writing \(t \in T[U]\).

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 \(\phi \triangleq (\varepsilon g_c, g_c)\) contains the static program counter label \(g_c\) used...
to type-check the source term, as well as the runtime program counter label \( g_c \), along with the evidence \( \varepsilon \vdash g_c \lessapprox g_c \). For simplicity we define accessors \( \phi(g_c) \approx g_c \), \( \phi(g) \approx g \), and \( \phi, \varepsilon \approx \varepsilon \).

Figure 23 presents the syntax of intrinsic terms. Fig. 24 presents the intrinsic terms formation rules for GSL\textsubscript{Ref}. In rule (iprot), labels \( g \) and \( g' \) represent the static and dynamic information of the label used to increase the program counter label in the subterm, respectively. Evidence \( \varepsilon_1 \) justifies that the type of the subterm is a consistent subtype of \( U \), the static type of the subterm. \( \phi' \) represents the context information associated to the subterm \( \ell : \phi'(g_c) \) (resp. \( \phi'(g) \)) is the program counter label used to typecheck (resp. evaluate) \( \ell \).

In the intrinsic term formation rule for applications (iapp), \( U_1 \) is the runtime type of the function term. We annotate the initial static type information with @. The evidence \( \varepsilon_2 \) for the label ordering premise is also annotated, since it is needed to reconstruct the derivation. The intrinsic term of a conditional, described in Rule (icif)\(^2\), carries the static information of the label of the conditional term \( g \). The context information \( \phi' \) used for both branches is obtained by joining the term context \( \phi \) point-wise with the evidence and labels associated with the consistent subtyping judgment of the conditional. Evidence \( \varepsilon_2 \) and \( \varepsilon_3 \) justify that the type of each branch is a consistent subtype of the join of both types. Finally, rule (iassgn) is built similarly to the application rule (iapp).

### 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 \( \tilde{t}_1 \mid \mu_1 \mapsto \tilde{t}_2 \mid \mu_2 \), to denote the reduction of the intrinsic term \( \phi \gg \tilde{t}_1 \in T[U] \) in store \( \mu_1 \) to the intrinsic term \( \phi \gg \tilde{t}_2 \in T[U] \) in store \( \mu_2 \). We note \( C[U] \) the combination of a term \( \tilde{t} \in T[U] \) (without context) and a store \( \mu \). Function applications reduce to to an error if consistent transitivity fails to justify \( U_2 \ll U_1 \). 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

\(^1\)We 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.

\(^2\)Evidence inversion functions (\textit{idom}, \textit{icod}, \textit{iref}, \textit{ilbl} and \textit{ilat}) manifest the evidence for the inversion principles on consistent subtyping judgments; \textit{e.g.} starting from the evidence that \( U_1 \lessapprox U_2 \), \textit{ilbl} produces the evidence of the judgment \( \text{label}(U_1) \lessapprox \text{label}(U_2) \).
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 $\phi.gc \searrow \ell \leq U$.

6.3 Relating Intrinsic and Evidence-augmented Terms

In this section we present the translation rules from GSL$\text{Ref}$ terms to intrinsic terms in Figure 27. Also this section presents the erasure function in in Figure 28—highlighting the syntactic 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 erasure of the translation to intrinsic term is equal to the translation of the source term to an evidence-augmented term:

**Proposition 6.1.** If $\Gamma; \Sigma; g_c \vdash t \rightsquigarrow \tilde{t} : U$ and $\Gamma; \Sigma; gc \vdash t \rightsquigarrow t' : U$, then $|\tilde{t}| = t'$.

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

Second, given a reducible intrinsic term $\tilde{t}$, if it reduces to an error, then it erasure also reduces to an error; or, if reduces to an intrinsic term $t'$, then the erasure of $t'$ also reduces to the erasure of $\tilde{t}$:

**Proposition 6.2.** Consider $\phi = e\, gc_c, \phi \vdash \tilde{t} : T[U]$, and $; \Sigma; gc_c \vdash t : U$, such that $\Sigma \models \mu_2$. Then if $\tilde{t} = t$ and $\mu_1 = \mu'_1$ then either

\[
\begin{align*}
(Ix) & \quad \phi \vdash xU \in T[U] \\
(Ib) & \quad \phi \vdash g \in T[\text{Bool}_g] \\
(lu) & \quad \phi \vdash \text{unit}_g \in T[\text{Unit}_g]
\end{align*}
\]

\[
\begin{align*}
(Ii) & \quad \phi \vdash gU \in T[\text{Ref}_g U] \\
(lu') & \quad \phi' = (\epsilon, g', g'') \quad \phi \vdash t \in T[U_2] \\
(lu) & \quad \phi \vdash (\lambda \delta xU_1, i)_g \in T[U_1 \xrightarrow{g'} g' U_2]
\end{align*}
\]

\[
\begin{align*}
(Iprot) & \quad \phi \vdash \text{ref}^{\delta U}(\epsilon, i) \in T[U \xrightarrow{g} g'] \\
(lapp) & \quad \phi \vdash I_1 \in T[U_1] \quad \epsilon_1 + U_1 \leq U_2 \quad \epsilon_2 + g' \leq g \\
(lif) & \quad \phi \vdash i_1 \in T[U_1] \quad \epsilon_1 + U_1 \leq \text{Bool}_g \quad \phi' = \phi \xrightarrow{i} \langle \text{ibl}(\epsilon_1), \text{label}(U_1), g \rangle \\
(lif) & \quad \phi \vdash i_2 \in T[U_2] \quad \epsilon_2 + U_2 \leq U_3 \quad \phi' \vdash i_3 \in T[U_3] \quad \epsilon_3 + U_3 \leq U_3 \quad \phi \vdash \text{ref}^{\delta U}(\epsilon, i_1) \text{ then } \epsilon_2 + U_2 \leq U_3 \quad \epsilon_3 + \phi.g_c \searrow g \leq g' \\
(lif) & \quad \phi \vdash i \in T[U'] \quad \epsilon + U' \leq \text{Ref}_g U \\
(lif) & \quad \phi \vdash !\text{Ref}_g U \in T[U \xrightarrow{g} g'] \\
(lapp) & \quad \phi \vdash i_1 \in T[\text{Ref}_g U_1] \quad \epsilon_1 + \text{Ref}_g U_1 \leq \text{Ref}_g U_1 \\
(lapp) & \quad \phi \vdash i_2 \in T[U_2] \quad \epsilon_2 + U_2 \leq U_1 \quad \epsilon_3 + \phi.g_c \searrow g \leq \text{label}(U_1) \\
(lapp) & \quad \phi \vdash i_1 ; U_1 = \epsilon_1, \epsilon_2 + U_2 \in T[\text{Unit}_g]
\end{align*}
\]
\[
\phi \mapsto C[U] \times (C[U] \cup \{\text{error}\})
\]

\[
\begin{align*}
(R\rightarrow) & \quad t^U | \mu \xrightarrow{\phi} r & r \in C[U] \cup \{\text{error}\} \\
(Rf) & \quad \tilde{i}_1 | \mu \xrightarrow{\phi} \tilde{i}_2 | \mu' \\
(R\text{prot}) & \quad \tilde{i}_1 | \mu \xrightarrow{\phi'} \tilde{i}_2 | \mu' \\
(R\text{herr}) & \quad et \xrightarrow{c} \text{error} \\
(R\text{protherr}) & \quad \tilde{i} | \mu \xrightarrow{\phi'} \text{error} \\
(R\text{pred}) & \quad \text{prot}_{\epsilon \phi'} (et) | \mu \xrightarrow{\phi} \text{prot}_{\epsilon \phi'} (et') | \mu'
\end{align*}
\]

Fig. 25. GSLRef: Intrinsic Reduction

- \(\tilde{i} | \mu_1 \xrightarrow{\phi} \tilde{i}' | \mu_2 \Rightarrow |\tilde{i}| |\mu_2| \xrightarrow{\epsilon g} |\tilde{i}'| |\mu'_2|\), or
- \(\tilde{i} | \mu_1 \xrightarrow{\phi} \text{error} \Rightarrow |\tilde{i}| |\mu_2| \xrightarrow{\epsilon} \text{error}\)

**Proof.** By induction on the type derivation of \(\tilde{i}\).

**Case (I::).** Then \(\tilde{i} = \epsilon_1 \tilde{i}' :: U\) and by (E::), \(t = \epsilon_t \tilde{t}'\) for some \(t'\) such that \(\tilde{t}' = \tilde{t}\). Suppose that \(\tilde{t}' \ll U\). By inspection on the type derivations, \(\phi \triangleright \tilde{i}' \in T[U']\) and \(;\Sigma; \epsilon_1 \epsilon_t \epsilon_t' : U\).

Let us suppose that \(\tilde{i}' | \mu_1 \xrightarrow{\phi} \tilde{i}'' | \mu_2\), then by induction hypothesis \(t' | \mu_2 \xrightarrow{\epsilon g} t'' | \mu'_2\) and \(\tilde{t}' = \tilde{t}''\) and \(\mu_1' = \mu_2'\). Then \(\epsilon_1 \tilde{i}' :: U | \mu_1 \xrightarrow{\phi} \epsilon_1 \tilde{i}'' :: U | \mu_1'\) and \(\epsilon_1 t' | \mu_2 \xrightarrow{\epsilon g} \epsilon_1 \epsilon_t \epsilon_t' | \mu_2'.\) But as \(\mu_1' = \mu_2',\) and by (E::) \(\epsilon_1 \epsilon_t \epsilon_t' : U = \epsilon_1 \tilde{t}'\), the result holds.

Let us suppose now that \(\tilde{i}' = \epsilon_2 u :: U\). Then as \(\tilde{i}' = \tilde{t}'\), \(t' = \epsilon_2 u\), for some \(u\) such that \(u = u'\). If \(\epsilon_2 \circ \triangleleft : \epsilon_1 = \epsilon'\), then \(\epsilon_1 (\epsilon_2 u :: U') :: U | \mu_1 \xrightarrow{\phi} \epsilon' u :: U | \mu_1\) and \(\epsilon_1 (\epsilon_2 u') | \mu_2 \xrightarrow{\epsilon g} \epsilon' u' | \mu_2.\) But as \(\mu_1 = \mu_2\), and by (E::) \(\epsilon' u :: U = \epsilon' u\), the result holds.

If \(\tilde{i}' = u\), then as \(\tilde{i}' = t'\), \(t' = \epsilon_2 u\), for some \(u\) such that \(u = u'\), and the result holds immediately.

The other cases proceed analogous. \(\square\)

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

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

**Proof.** By induction on the type derivation of \(\tilde{i}\). \(\square\)
Notions of Reduction

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

\[ \epsilon_1(b_1)_{g_1} \oplus^\phi \epsilon_2(b_2)_{g_2} | \mu \xrightarrow{\phi} (\epsilon_1 \parallel \epsilon_2)(b_1 \parallel b_2)_{(g_1 \parallel g_2)} : \text{Bool}_g | \mu \]

\[ \text{prot}^g_{\epsilon_2} \phi' \epsilon_1 \ u | \mu \xrightarrow{\phi} (\epsilon_1 \parallel \epsilon_2)(u \parallel g') : U \parallel g | \mu \]

\[ \epsilon_1(\lambda \epsilon_2 x.U_{11} \cdot t)^*_{g_2} \parallel \epsilon_2 u | \mu \xrightarrow{\phi} \begin{cases} \text{prot}^{g_1}\epsilon_2 U^2_{\text{ilbl}(\epsilon_1)} \phi' \epsilon_1 (\text{idom}(\epsilon_1)) | \mu & \text{error} \\
\text{if } \epsilon \text{ or } \epsilon' \text{ are not defined} \\
\text{where } \epsilon = \epsilon_2 \circ <: \text{idom}(\epsilon_1), \epsilon' = (\phi.g \parallel \text{ilbl}(\epsilon_1)) \circ < \epsilon_3 \circ < \text{ilat}(\epsilon_1)
\end{cases} \]

\[ \text{if } \epsilon_1 \text{ true}_g, \text{ then } \epsilon_2 U^2 \text{ else } \epsilon_1 U_1 | \mu \xrightarrow{\phi} \begin{cases} \text{prot}^{g_1}\epsilon_2 U^2_{\text{ilbl}(\epsilon_1)} \phi' \epsilon_1 U^1 | \mu & \text{error} \\
\text{where } \epsilon' = (\phi.g \parallel \text{ilbl}(\epsilon_1), \phi.g \parallel g_1, \phi.g \parallel g) \text{ and } U = (U_2 \parallel U_3) \end{cases} \]

\[ \text{if } \epsilon_1 \text{ false}_g, \text{ then } \epsilon_2 U_2 \text{ else } \epsilon_1 U_1 | \mu \xrightarrow{\phi} \begin{cases} \text{prot}^{g_1}\epsilon_2 U_2_{\text{ilbl}(\epsilon_1)} \phi' \epsilon_1 U_1 | \mu & \text{error} \\
\text{where } \epsilon' = (\phi.g \parallel \text{ilbl}(\epsilon_1), \phi.g \parallel g_1, \phi.g \parallel g) \text{ and } U = (U_2 \parallel U_3) \end{cases} \]

\[ \text{ref}^U_{\epsilon \ u} | \mu \xrightarrow{\phi} \begin{cases} U | \mu[u^U \mapsto \epsilon' (u \parallel g \cdot c) :: U] & \text{error} \\
\text{if } \mu(o^U) \notin \text{dom}(|\mu|) \end{cases} \]

\[ !\text{ref}^U_{\epsilon \ u} | \mu \xrightarrow{\phi} \text{prot}^{g_1}\epsilon_2 U^2_{\text{ilbl}(\epsilon)} g' (\text{iref}(\epsilon) v) \]

\[ \epsilon_1 o^U_{\mu} := \epsilon_1 \epsilon_2 u | \mu \xrightarrow{\phi} \begin{cases} \text{unit}_U | \mu[o^U \mapsto \epsilon' (u \parallel g \cdot c) :: U] & \text{error} \\
\text{if } \epsilon' \text{ is not defined, or} \\
\phi.g \parallel \text{ilbl}(|\epsilon|) \text{ does not hold} \end{cases} \]

\[ \text{where } \epsilon' = (\epsilon_2 \circ <: \text{iref}(|\epsilon|)) \parallel ((\phi.g \parallel \text{ilbl}(\epsilon_1)) \circ < \epsilon_3 \circ < \text{ilbl}(|\epsilon|)) \text{ and } \mu(o^U) = \epsilon u' :: U \]

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

\[ \epsilon_1(\epsilon_2 v :: U) \longrightarrow_{c} \begin{cases} (\epsilon_2 \circ <: \epsilon_1) v & \text{error} \\
\text{if not defined} \end{cases} \]

\[ \langle \ell_1, \ell_2, [\ell_3, \ell_4] \rangle \parallel \langle \ell_1', \ell_2' [\ell_3', \ell_4'] \rangle \iff \ell_3 \leq \ell_4' \]

Fig. 26. GSLRef: Intrinsic Notions of Reduction

Finally, if an evidence-augmented term type checks, then there must exist 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 \( \exists i \exists \phi \text{ such that } |i| = t \text{ and } |\phi| = \epsilon g c \text{ and } \phi \triangleright t \in T[U] \)

**Proof.** By induction on the type derivation of \( t \).
Case ($\epsilon', t'$). Then $t = \epsilon' t'$, for some $\epsilon', t'$. But we know that $\Gamma; g_c \vdash (\epsilon', t') : U$ and suppose $\epsilon' + U \leq U$ and $\epsilon + g_c \leq g_c'$. Then by choosing $\phi = (\epsilon, g_c)g_c'$ 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 $|\bar{t}_1| \subseteq |\bar{t}_2|$.

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

**LEMMA 6.6.** Consider $\phi \triangleright \bar{t}_1 \in T[U]$. If $|\bar{t}_1| \subseteq |\bar{t}_2|$, 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.

\[ \varepsilon_\text{ref} \text{ depends on } \varepsilon_\text{ref} \text{ of evidence, we can build the term } \varepsilon_\text{ref} \text{.} \]

\[ \text{Lemma 6.9.} \quad \text{Suppose } \phi \vdash t^U \in T[U], \text{ then } \forall g', \forall \varepsilon_3', \text{ such that } g'_3 \leq \phi, g_c \text{ and } \varepsilon'_3 \vdash g'_3 \leq \phi, g_c, \phi' = (\varepsilon'_3, g'_3, \phi, g_c) \text{ then } \phi' \vdash t^U \in T[U]. \]

Proof. By induction on the derivation of \( \phi \vdash t^U \in T[U]. \) Noticing that no typing derivation depends on \( \varepsilon'_3, g'_3, \) save for the judgement \( \varepsilon'_3 \vdash g'_3 \leq g_c \) which is premise of this lemma.

\[ \text{Lemma 6.9.} \quad \text{Suppose } \phi \vdash v \in T[U], \text{ then } \forall \phi', \text{ then } \phi' \vdash v \in T[U]. \]
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 \( v = \varepsilon u :: U \) with \( u \in T[U'] \) and \( \varepsilon \vdash U' \leq U \). Furthermore:

1. If \( U = \text{Bool}_g \) then either \( v = b_g \) or \( v = \varepsilon b' \vdash \text{Bool}_{g'} \) with \( b' \vdash \text{Bool}_{g'} \leq \text{Bool}_g \).
2. If \( U = U_1 \Gamma U_2 \) then either \( v = (\lambda x : U_1 . t : U_2) \) with \( t : U_2 \in T[U_2] \) or \( v = \varepsilon (\lambda x : U_1' . t : U_2') \vdash U_1 \Gamma U_2 \) with \( t : U_2' \in T[U_2] \) and \( \varepsilon \vdash U_1' \gamma g \leq U_1 \Gamma g \).
3. If \( U = \text{Ref}_g U_1 \) then either \( v = o_{U_1} \) or \( v = \varepsilon o_{U_1'} \vdash \text{Ref}_{g'} U_1 \) with \( o_{U_1'} \leq \text{Ref}_{g'} \) and \( \varepsilon \vdash \text{Ref}_{g'} U_1' \leq \text{Ref}_{g'} U_1 \).

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

Lemma 6.11 (Substitution). If \( \phi \triangleright t : U \) and \( \phi' \triangleright v \in T[U] \), then \( \phi \triangleright [v/x : U_1] t : U \).

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

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

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

Case (\( \oplus \)). Then \( t : U = b_1 : U_1 \oplus b_2 : U_2 \). By construction we can suppose that \( g = g_1 \triangleright \neg g_2 \), then

\[
\phi \vdash b_1 : U_1 \quad \varepsilon_1 \vdash \text{Bool}_{g_1} \leq \text{Bool}_{g_2}
\]

\[
\phi \vdash b_2 : U_2 \quad \varepsilon_2 \vdash \text{Bool}_{g_2} \leq \text{Bool}_{g_2}
\]

\[(\oplus)\]

Therefore

\[
\phi \vdash \varepsilon_1(b_1 : U_1) \oplus \varepsilon_2(b_2 : U_2) \mid \mu
\]

\[
(\oplus) \quad \phi \vdash (\varepsilon_1 \neg \varepsilon_2)(b_1 \oplus b_2)_{(g_1 \neg g_2)} : \text{Bool}_g \mid \mu
\]

Then

\[
(\oplus) \quad \phi \vdash \varepsilon_1 \neg \varepsilon_2 \mid \mu
\]

and the result holds.

Case (\( \text{prot} \)). Then \( t : U = \phi \triangleright \text{prot}_{g_1 : U}^g \phi' (\varepsilon u) \) and

\[
\phi \vdash \varepsilon \leq g_1 \quad \varepsilon \vdash g \leq g_1
\]

\[(\text{prot})\]

Therefore

\[
\text{prot}_{g_1 \neg g}^g \phi' (\varepsilon u) \mid \mu \quad \phi \vdash \varepsilon \neg \varepsilon \mid \mu
\]
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

\[
\phi \triangleright (u \triangleright g') \in T[U' \triangleright g']
\]

and the result holds.

**Case (lapp).** Then \( t^U = e_1(\lambda \delta'^g x^{U_1}, t^{U_2})g_1 \odot_{\ell'} u \epsilon_2 u \) and \( U = U_2 \triangleright \triangleright g \). Then

\[
\frac{\phi \triangleright t^{U_2} \in T[U_2]}{\phi \triangleright (\lambda \delta'^g x^{U_1}, t^{U_2})g_1 \in T[U_1]}
\]

\[
\frac{\phi \triangleright u \in T[U'_2]}{\epsilon_2 \triangleright u \leq U_1}
\]

\[
\frac{\epsilon_1 + U_1 \delta'^g \triangleright g_1 u \leq U_1 \delta'_e U_2}{\phi \triangleright \epsilon_\ell \triangleright g_e \leq g'_e}
\]

If \( \epsilon' = (\epsilon_2 \circ \triangleright idom(\epsilon_1)) \) or \( \epsilon'_l = (\phi, \epsilon_\ell \triangleright ilbl(\epsilon_1)) \circ \triangleleft \epsilon_\ell \circ \triangleleft ilat(\epsilon_1) \) are not defined, then \( t^U \mid \mu \rightarrow \text{error} \), and then the result hold immediately. Suppose that consistent transitivity does hold, then if \( \phi' = (\phi, \epsilon_\ell \triangleright g_e \triangleright g_1, g'_e) \)

\[
e_1(\lambda \delta'^g x^{U_1}, t^{U_2})g_1 \odot_{\ell'} u \epsilon_2 u \mid \mu \rightarrow \text{prot}^{12}_{\text{prot}, U_2} (\text{cod}(\epsilon_1) \mid \text{prot}^{12}_{(\epsilon' \triangleright U_1) / x^{U_1}, t^{U_2}}) \mid \mu
\]

As \( \epsilon_2 \triangleright u \leq U_1 \) and by inversion lemma \( idom(\epsilon_1) \triangleleft U_1 \leq U_11 \), then \( \epsilon' \triangleright U'_2 \leq U_1 \). Therefore

\[
\phi \triangleright \epsilon'_l \triangleright U_1 \leq U_11 \text{ and by Lemma 6.11,}
\]

\[
\phi \triangleright (\epsilon' \triangleright t^{U_2}) \in T[U_1] \text{.}
\]

We know that \( \epsilon_\ell \triangleright g_e \leq g'_e \). By inversion on the label of types, \( ilbl(\epsilon_1) \triangleleft g_1 \leq g \). Also by monotonicity of the join, \( \phi, \epsilon_\ell \triangleright ilbl(\epsilon_1) \triangleright \phi, g_e \triangleright g_1 \leq g_e \triangleright g \). Then, by inversion on the latent effect of function types, \( ilat(\epsilon_1) \triangleright g_e' \leq g_e'' \). Therefore combining evidences, as \( \phi', \epsilon_\ell = (\phi, \epsilon_\ell \triangleright ilbl(\epsilon_1)) \circ \triangleleft \epsilon_\ell \circ \triangleleft ilat(\epsilon_1) \), we may justify the runtime judgment \( \phi', \epsilon_\ell \triangleright \phi, g_e \triangleright g_1 \leq g_e'' \).

Let us call \( t^{U_2} = (\epsilon' \triangleright U_1) / x^{U_1}, t^{U_2} \). By Lemma 6.8, \( \phi' \triangleright t^{U_2} \in T[U_2] \). Then

\[
\frac{\phi, \epsilon_\ell \triangleright g_e \leq \phi, g_e}{\phi' \triangleright t^{U_2} \in T[U_2]}
\]

\[
\frac{icod(\epsilon_1) \triangleleft U_1 \leq U_2 \triangleleft ilbl(\epsilon_1) \triangleleft g_1 \leq g}{\phi' \triangleright \text{prot}^{12}_{ilbl(\epsilon_1) \epsilon_1} (\text{cod}(\epsilon_1) \mid t^{U_2}) \in T[U_2 \triangleright g]}
\]

and the result holds.
Case (if-true). Then \( t^U = \text{if}_{\varphi b} \varphi_1 b_{\varphi_1} \) then \( \varepsilon_2 t^{U_2} \) else \( \varepsilon_3 t^{U_3} \), \( U = (U_2 \triangleright U_3) \triangleright g \) and

\[
\begin{align*}
\phi \triangleright b_{\varphi_1} &\in T[\text{Bool}_{\varphi_1}] \\
\varphi' = (\phi, \varepsilon \triangleright \text{ilbl}(\varphi_1)(\phi, \varepsilon, \triangleright \triangleright \varphi_1). \phi, \varepsilon, \triangleright \triangleright \varphi_1 &\leq \phi, \varepsilon. \\
\phi' &\triangleright t^{U_2} \in T[U_2] \\
\phi' &\triangleright t^{U_3} \in T[U_3]
\end{align*}
\]

Therefore

\[
\text{if}_{\varphi b} \varphi_1 b_{\varphi_1} \text{ then } \varepsilon_2 t^{U_2} \text{ else } \varepsilon_3 t^{U_3} \in T[(U_2 \triangleright U_3) \triangleright g]
\]

But

\[
\phi, \varepsilon, \triangleright \triangleright \varphi_1 \leq \phi, \varepsilon. \\
\phi' \triangleright t^{U_2} \in T[U_2] \\
\phi' \triangleright t^{U_3} \in T[U_3]
\]

and the result holds.

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

Case (iref). Then \( t^U = \text{iref}_{\varphi b} \varphi u \) and

\[
\begin{align*}
\phi \triangleright \varphi u &\in T[U'''] \\
\varepsilon &\triangleright U''' \leq U' \\
\varepsilon &\triangleright u \leq \text{label}(U')
\end{align*}
\]

If \( \varepsilon' = \varepsilon \triangleright (\phi, \varepsilon) \) is not defined, then \( t^{U''} \mid \mu \xrightarrow{\phi} \text{error} \), and then the result hold immediately. Suppose that consistent transitivity does hold, then

\[
\text{ref}_{\varphi b} \varphi u \mid \mu \xrightarrow{\phi} (\phi, \varepsilon, \triangleright) \triangleright (u \triangleright \varphi. \phi \varepsilon_1) : U'
\]

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

We know that \( \varepsilon \triangleright g \leq \text{label}(U') \), therefore \( \phi, \varepsilon \triangleright o^{U''} \leq (o^{U''}) \leq \text{label}(U') \). We also know that \( \varepsilon \triangleright U''' \leq U' \). Therefore combining both evidences we can justify that \( \varepsilon \triangleright (\phi, \varepsilon, \triangleright) \triangleright U_2' \triangleright \varphi. \phi \varepsilon_1 \leq U'' \). But

\[
\begin{align*}
\phi \triangleright \varphi u \mid \mu &\xrightarrow{\phi} (\phi, \varepsilon, \triangleright) \triangleright (u \triangleright \varphi. \phi \varepsilon_1) : U'
\end{align*}
\]

Let us call \( \mu' = \mu[o^{U''} \mapsto \varepsilon'(u \triangleright \varphi. \phi \varepsilon_1) : U'] \). It is easy to see that \( \text{freeLocs}(o^{U''}) = o^{U''} \) and \( \text{dom}(\mu') \subseteq \text{dom}(\mu) \). Given that \( t^{U''} \mid \mu \) then \( \text{freeLocs}(\mu) \subseteq \text{dom}(\mu) \), and therefore \( \forall u \in \text{cod}(\mu') = \text{cod}(\mu) \cup (u \triangleright \varphi. \phi \varepsilon_1 : U') \), \( \text{freeLocs}(v') \subseteq \text{dom}(\mu') \). Finally as \( t^{U''} \mid \mu \) and \( \mu'(o^{U''}) = \varepsilon'(u \triangleright \varphi. \phi \varepsilon_1) : U' \in T[U'] \) then we can conclude that \( t^{U''} + \mu' \) and \( \text{dom}(\mu) \subseteq \text{dom}(\mu') \), and the result holds.

Case (ideref). Then \( t^U = \text{ideref}_{\varphi b} \varphi o^{U''} b_\varphi \), \( U = U \triangleright g \) and

\[
\begin{align*}
\phi &\triangleright o^{U''} \in T[\text{Ref}_{\varphi} U'''] \\
\varepsilon &\triangleright \text{Ref}_{\varphi} U'' \leq \text{Ref}_{\varphi} U'
\end{align*}
\]

Therefore

\[
\phi \triangleright o^{U''} \in T[\text{Ref}_{\varphi} U''']
\]

But

\[
\phi \triangleright o^{U''} \in T[\text{Ref}_{\varphi} U''']
\]

Therefore

\[
\phi \triangleright o^{U''} \in T[\text{Ref}_{\varphi} U''']
\]
Then for $\phi' = ((\phi.ε \triangleright ilbl(\epsilon))(φ.g_c \triangleright g'), φ.g_c \triangleright g)$

\[ t^\text{Ref}_g U'_\phi \overset{\mu}{\longrightarrow} \text{prot}_{\text{ilbl}(\epsilon)g'}^g(\text{iref}(\epsilon)v) | \mu \]

where $\mu(o^{U''}) = v$. As the store is well typed, therefore $\phi \triangleright v \in T[U'']$. By Lemma 6.9, $\phi' \triangleright v \in T[U'']$. By inversion lemma on references, $\text{ilbl}(\epsilon) + g' \triangleleft g$ and $\text{iref}(\epsilon) + U'' \triangleleft U'$

\[ \phi.ε + φ.g_c \triangleleft \phi.g_c \quad \text{iref}(\epsilon) + U'' \triangleleft U' \]

\[ (\text{prot}) \phi \overset{\mu}{\longrightarrow} \text{prot}_{\text{ilbl}(\epsilon)g'}^g(\text{iref}(\epsilon)v) \in T[U' \triangleright g] \]

and the result holds.

**Case (assgn).** Then $t^U = ε_1 o^{g_1}_{g_2} := ε_2 u$ and

\[ ε_1 + \text{Ref}_g U'_1 \triangleleft \text{Ref}_g U \quad φ \overset{o^{g_1}_{g_2}}{\longrightarrow} \text{Ref}_g U_1' \]

\[ (\text{assgn}) \quad φ.ε + φ.g_c \triangleleft \phi.g_c \quad ε_2 + φ.g_c \triangleright g \leq \text{label}(U_1) \]

\[ (\text{assgn}) \quad \phi \overset{ε_1 o^{g_1}_{g_2}}{\longrightarrow} \quad ε_2 u \in T[\text{Unit}_\perp] \]

If $ε' = (ε_2 o^{<}_g \text{iref}(\epsilon_1))^{\text{ref}(\epsilon_1)}(φ.ε \triangleright ilbl(\epsilon))^{\text{ref}(\epsilon_1)} φ.ε \triangleright ilbl(\epsilon)$ is not defined, then $t^{U'} | \mu \overset{\phi}{\longrightarrow} \text{error}$, and then the result hold immediately. Suppose that consistent transitivity does hold, then

\[ ε_1 o^{g_1}_{g_2} := ε_2 u \in T[\text{Unit}_\perp] \]

We know that $ε_2 \triangleright φ.g_c \triangleright g \leq \text{label}(U_1)$. Then by inversion on reference evidence types and inversion in the label of types, $\text{ilbl}(\text{iref}(\epsilon_1)) + \text{label}(U_1) \leq \text{label}(U_1')$. But $\text{ilbl}(\epsilon_1) + g' \triangleleft g$, using monotonicity of the join, $φ.ε \triangleright ilbl(\epsilon_1) + φ.g_c \triangleright g' \triangleleft φ.g_c \triangleright g$. Therefore

\[ ((φ.ε \triangleright ilbl(\epsilon_1))^{\text{ref}(\epsilon_1)} φ.ε \triangleright ilbl(\epsilon) + φ.g_c \triangleright g' \leq \text{label}(U_1)) \]

We also know that if $u \in T[U_2]$, then $ε_2 o^{<}_g \text{iref}(\epsilon_1) + \text{Ref}_g U_2 \triangleleft U_2'$ Combining both evidences, $ε' = (ε_2 o^{<}_g \text{iref}(\epsilon_1))^{\text{ref}(\epsilon_1)}((φ.ε \triangleright ilbl(\epsilon_1))^{\text{ref}(\epsilon_1)} φ.ε) \triangleright ilbl(\text{iref}(\epsilon_1))$, and by Proposition 5.43 we can then justify that $ε' + η \leq \text{label}(U_1)$ and therefore justify the ascription in the heap.

Let us call $μ' = μ[O^{U'} \mapsto ε'(u \triangleright (φ.g_c \triangleright g)) :: U_1']$. As freeLocs(unit⊥) = ∅ then freeLocs(unit⊥) ⊆ μ'.

As $t^U \triangleright μ$ then freeLocs(u) ∈ dom(μ), and as dom(μ) = dom(μ') then it is trivial to see that $\forall v' \in \text{cod}(μ'), \text{freeLocs}(v') \subseteq \text{dom}(μ')$, and the result holds.

\[ \square \]

**Proposition 6.13 (→ is well defined).** If $t^U | μ \overset{\phi}{\longrightarrow} r$ and $t^U \triangleright μ$, then $r \in \text{Config}_U \cup \{\text{error}\}$ and if $r = t^{U'} | μ' \in \text{Config}_U$ then also $t^{U'} \triangleright μ'$ and dom(μ) ⊆ dom(μ').

**Proof.** By induction on the structure of a derivation of $t^U | μ \overset{\phi}{\longrightarrow} r$.

**Case (R→).** $t^U | μ \overset{φ}{\longrightarrow} r$. By well-definedness of $→$ (Prop 6.12), $r \in \text{Config}_U \cup \{\text{error}\}$ and if $r = t^{U'} | μ' \in \text{Config}_U$ then also $t^{U'} \triangleright μ'$ and dom(μ) ⊆ dom(μ').
Case (Rprot). \( t^U = \text{prot}^{g,U'}_{\phi'} (\epsilon_{t_1^U}) \) and 
\[
\phi : c + \phi \rightarrow_\epsilon \phi \rightarrow_\epsilon \phi \rightarrow_\epsilon g \rightarrow_\epsilon g' \leq g_c \\
\phi \rightarrow_\epsilon t_1^U \in T[U']
\]

\((\text{prot})\)
\[
\phi \rightarrow_\epsilon \text{prot}^{g,U'}_{\phi'} (\epsilon t_1^U) \in T[U' \lor g]
\]

Using induction hypothesis on the premise of (Rprot()), then
\[
t_1^U | \mu \rightarrow_\epsilon t_2^U | \mu'
\]

\((\text{Rprot}())\)
\[
\text{prot}^{g,U'}_{\phi'} (\epsilon t_1^U) | \mu \rightarrow_\epsilon \text{prot}^{g,U'}_{\phi'} (\epsilon t_2^U) | \mu'
\]

where \( \phi' \rightarrow t_2^U \in T[U'], t_2^U + \mu' \) and \( \text{dom}(\mu) \subseteq \text{dom}(\mu') \). Therefore
\[
\phi : c + \phi \rightarrow_\epsilon \phi \rightarrow_\epsilon \phi \rightarrow_\epsilon g \rightarrow_\epsilon g' \leq g_c \\
\phi \rightarrow_\epsilon t_2^U \in T[U']
\]

\((\text{prot})\)
\[
\phi \rightarrow_\epsilon \text{prot}^{g,U'}_{\phi'} (\epsilon t_2^U) \in T[U' \lor g]
\]

and the result holds.

Case (Rf). \( t^U = f[t_1^U], \phi \rightarrow f[t_1^U] \in T[U], t_1^U | \mu \rightarrow_\phi t_2^U | \mu' \), and consider \( F : T[U'] \rightarrow T[U] \), where \( F(\phi \rightarrow t') = \phi \rightarrow f[t'] \). By induction hypothesis, \( \phi \rightarrow t_2' \in T[U'] \), so \( F(\phi \rightarrow t_2') = \phi \rightarrow f[t_2'] \in T[U] \).

By induction hypothesis we also know that \( t_2'^U \vdash \mu' \).

If \( \text{freeLocs}(t_2^U) \subseteq \mu', \text{freeLocs}(f[t_1^U]) \subseteq \mu, \) and \( \text{dom}(\mu) \subseteq \text{dom}(\mu') \), then it is easy to see that \( \text{freeLocs}(f[t_1^U]) \subseteq \mu' \), and therefore conclude that \( f[t_1^U] \vdash \mu' \).

Case (Rerr, Rherr). \( r = \text{error} \).

Case (Rh). \( t^U = h[et], \phi \rightarrow h[et] \in T[U], \) and consider \( G : \text{EvTerm} \rightarrow \text{EvTerm} \rightarrow \text{EvTerm} \rightarrow \text{EvTerm} \), such that \( G(\phi, et) = \phi \rightarrow h[et] \) and \( et \rightarrow_\epsilon e' \). Then there exists \( U_e, U_x \) such that \( e = e_{e U_e} \rightarrow_\epsilon e' \). Also, \( U_e \subseteq U_e \) and \( U_x \subseteq U_x \). We know that \( e_{e U_e} \subseteq e \) is defined, and \( et = e_{e U_e} \rightarrow_\epsilon e_{e U_e} = e' \). By definition of \( \circ \leq \), we have \( e_{e U_e} \subseteq e \), so \( G(\phi, et') = \phi \rightarrow h[et'] \in T[U] \).

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 \( P : \text{EvTerm} \rightarrow T[U], P(et) = \phi \rightarrow \text{prot}^{g,U'}_{\phi'} (et) \).

\( \square \)

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 \( \phi \rightarrow t^U \in T[U] \) then either \( t^U \) is a value \( v \); \( t^U | \mu \rightarrow_\phi \text{error} \); or if \( t^U \vdash \mu \) then \( t^U | \mu \rightarrow_\phi t'^U | \mu' \) for some term \( \phi \rightarrow t'^U \in T[U] \) and some \( \mu' \) such that \( t'^U \vdash \mu' \) and \( \text{dom}(\mu) \subseteq \text{dom}(\mu') \).

**Proof.** By induction on the structure of \( \phi \rightarrow t^U \).

Case (Iu, Iil, Ix, Ix). \( t^U \) is a value.
Case (Iprot). \( t^U = \text{prot}^\phi_g f'^r c t^U' \), and
\[
\phi \vdash \phi.g_c \trianglelefteq \phi.g_c. \quad \epsilon \uparrow g_r \land g' \trianglelefteq g_c
\]
\[
\phi' \vdash t^U' \in T[U']
\]
\[
\phi \vdash \text{prot}^\phi_g f'^r c (\epsilon t^U') \in T[U \triangleright g]
\]

By induction hypothesis on \( t^U' \), one of the following holds:

1. \( t^U' \) is a simple value, then by (R\(
\rightarrow \)), \( t^U | \mu \xrightarrow{\phi} \nu | \mu \), and by Prop 6.13, \( \phi \vdash \nu \in T[U] \) and the result holds.
2. \( t^U' \) is an ascribed value \( \nu \), then, \( \epsilon t^U' \rightarrow_c et' \) for some \( et' \in \text{EvTerm} \cup \{ \text{error} \} \). Hence \( t^U | \mu \xrightarrow{\phi} r \) for some \( r \in \text{Config}_U \cup \{ \text{error} \} \) by Prop 6.13 and either (Rg), or (Rgerr).
3. \( t^U' | \mu \xrightarrow{\phi} r_1 \) for some \( r_1 \in T[U_1] \cup \{ \text{error} \} \). Hence \( t^U | \mu \xrightarrow{\phi} r \) for some \( r \in \text{Config}_U \cup \{ \text{error} \} \) by Prop 6.13 and either (Rprot()), or (Rprot()err).

Case (I:). \( t^U = \epsilon_1 t^{U_1} :: U_2 \), and
\[
\phi \vdash t^{U_1} \in T[U_1]
\]
\[
\epsilon_1 \vdash U_1 \leq U_2 \quad \phi \vdash \phi.g_c \trianglelefteq \phi.g_c
\]
\[
\phi \vdash \epsilon_1 t^{U_1} :: U_2 \in T[U_2]
\]

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. \( t^{U_1} | \mu \xrightarrow{\phi} r_1 \) for some \( r_1 \in T[U_1] \cup \{ \text{error} \} \). Hence \( t^U | \mu \xrightarrow{\phi} r \) for some \( r \in \text{Config}_U \cup \{ \text{error} \} \) by Prop 6.13 and either (Rf), or (Rferr).

Case (Iif). \( t^U = \text{if}^U \epsilon_1 t^{U_1} \) then \( \epsilon_2 t^{U_2} \) else \( \epsilon_3 t^{U_3} \) and
\[
\phi \vdash t^{U_1} \in T[U_1]
\]
\[
\epsilon_1 \vdash U_1 \leq \text{Bool}_g \quad \phi \vdash \phi.g_c \trianglelefteq \phi.g_c
\]
\[
\phi' = ((\phi.\epsilon \triangleright \text{label}(U_1))(\phi.g_c \triangleright g))
\]
\[
\phi' \vdash t^{U_2} \in T[U_2]
\]
\[
\epsilon_2 \vdash U_2 \leq (U_2 \triangleright U_3)
\]
\[
\phi' \vdash t^{U_3} \in T[U_3]
\]
\[
\epsilon_3 \vdash U_3 \leq (U_2 \triangleright U_3)
\]
\[
\phi \vdash \text{if}^U \epsilon_1 t^{U_1} \text{ then } \epsilon_2 t^{U_2} \text{ else } \epsilon_3 t^{U_3} \in T[(U_2 \triangleright U_3) \triangleright g]
\]

By induction hypothesis on \( t^{U_1} \), one of the following holds:

1. \( t^{U_1} \) is a value \( u \), then by (R\(
\rightarrow \)), \( t^U | \mu \xrightarrow{\phi} r \) and \( r \in \text{Config}_U \cup \{ \text{error} \} \) by Prop 6.13.
2. \( t^{U_1} \) is an ascribed value \( \nu \), then, \( \epsilon_1 t^{U_1} \rightarrow_c et' \) for some \( et' \in \text{EvTerm} \cup \{ \text{error} \} \). Hence \( t^U | \mu \xrightarrow{\phi} r \) for some \( r \in \text{Config}_U \cup \{ \text{error} \} \) by Prop 6.13 and either (Rg), or (Rgerr).
3. \( t^{U_1} | \mu \xrightarrow{\phi} r_1 \) for some \( r_1 \in T[U_1] \cup \{ \text{error} \} \). Hence \( t^U | \mu \xrightarrow{\phi} r \) for some \( r \in \text{Config}_U \cup \{ \text{error} \} \) by Prop 6.13 and either (Rf), or (Rferr).
Case (lapp). \( t^U = \varphi_1 t^{U_1} @_{\varphi_2} t^{U_2} \)

\[
\begin{align*}
\phi &\triangleright t^{U_1} \in T[U_1] & \varphi_1 \triangleright U_1 \leq U_1^\varphi \rightarrow_{g_c} U_{12} \\
\phi &\triangleright t^{U_2} \in T[U_2] & \varphi_2 \triangleright U_2 \leq U_1
\end{align*}
\]

(lapp) \( \varphi \triangleright \varphi_1 t^{U_1} @_{\varphi_2} t^{U_2} \in T[U_1 \rightarrow_{g_c} g_c'] \)

\( \varphi \triangleright \varphi_1 t^{U_1} @_{\varphi_2} t^{U_2} \in T[U_1 \rightarrow_{g_c} g_c'] \)

By induction hypothesis on \( t^{U_1} \), one of the following holds:

1. \( t^{U_1} \) is a value \((\lambda x^{U_1} . t^{U_1}_c)\eta \) (by canonical forms Lemma 6.10), posing \( U_1 = U_1^\varphi \rightarrow_{g_c} U_{12} \).

Then by induction hypothesis on \( t^{U_2} \), one of the following holds:

(a) \( t^{U_2} \) is a value \( v \), then by (R →), \( t^U \mid \mu \triangleright r \) and \( r \in Config_U \cup \{ error \} \) by Prop 6.13.

(b) \( t^{U_2} \) is an ascribed value \( v \), then, \( \varepsilon_2 t^{U_2} \rightarrow_c \varepsilon \) for some \( \varepsilon \in EvTerm \cup \{ error \} \). Hence

\( t^U \mid \mu \triangleright \varepsilon \rightarrow r \) for some \( r \in Config_U \cup \{ error \} \) by Prop 6.13 and either (Rg), or (Rgerr).

(c) \( t^{U_2} \mid \mu \triangleright r_2 \) for some \( r_2 \in Config_U \cup \{ error \} \). Hence \( t^U \mid \mu \triangleright r \) for some \( r \in Config_U \cup \{ error \} \) 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 \( dom(\mu) \subseteq dom(\mu') \).

(2) \( t^{U_1} \mid \mu \triangleright \varepsilon \rightarrow r \) for some \( r \in Config_U \cup \{ error \} \) by Prop 6.13 and either (Rg), or (Rgerr).

(3) \( t^{U_1} \mid \mu \triangleright r_1 \) for some \( r_1 \in Config_U \cup \{ error \} \). Hence \( t^U \mid \mu \triangleright r \) for some \( r \in Config_U \cup \{ error \} \) 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 \( dom(\mu) \subseteq dom(\mu') \).

Case (Id). Similar case to (lapp)

Case (Iref). \( t^U = ref^U \varphi t^{U'} \) and

\[
\begin{align*}
\phi \triangleright \phi \cdot g_c &\sim\phi \cdot g_c \\
\varepsilon \triangleright U' &\leq U' \\
\varphi_2 \triangleright g_c &\leq label(U')
\end{align*}
\]

By induction hypothesis on \( t^{U'} \), one of the following holds:

1. \( t^{U'} \) is a value \( v \), then by (R →), \( t^{U'} \mid \mu \triangleright r \) and \( r \in Config_{U'} \) by Prop 6.13. Also by Prop 6.13, if \( r = t^{U'} \mid \mu' \in T[U] \) then \( dom(\mu) \subseteq dom(\mu') \).

2. \( t^{U'} \mid \mu \triangleright \varepsilon \rightarrow r_1 \) for some \( r_1 \in Config_{U'} \cup \{ error \} \). Hence \( t^{U'} \mid \mu \triangleright r \) for some \( r \in Config_{U'} \cup \{ error \} \) by Prop 6.13 and either (Rf), or (Rferr). Also by Prop 6.13, if \( r = t^{U'} \mid \mu' \in T[U] \) then \( dom(\mu) \subseteq dom(\mu') \).

Case (Ideref). \( t^U = [Ref^U \varepsilon] t^{U'} \)

\[
\begin{align*}
\phi \triangleright t^{U''} \in T[U''] &\quad \varepsilon \triangleright U'' \leq Ref^U U' \\
\phi \triangleright t^{U''} \in T[U''] &\quad \varepsilon \triangleright U'' \leq Ref^U U'
\end{align*}
\]
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\(\rightarrow\)), \(t'' \phi \rightarrow r\) and \(r \in \text{Config}_U\) by Prop 6.13.

2. \(t''\) is an ascribed value \(v\), then, \(\varepsilon t'' \rightarrow c\) et' for some \(et' \in \text{EvTerm} \cup \{\text{error}\}\). Hence \(t'' \phi \rightarrow r\) for some \(r \in \text{Config}_U \cup \{\text{error}\}\) by Prop 6.13 and either (Rg), or (Rgerr).

3. \(t''\) | \(\mu\) \(\phi\) \(r_1\) for some \(r_1 \in \text{Config}_{U''} \cup \{\text{error}\}\). Hence \(t'' \mu \phi \rightarrow r\) 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 (Iassign). \(t'' = \varepsilon_t t''\ v =: \varepsilon_{\ell_t} \varepsilon_{\ell_t} t''\) and

\[
\begin{align*}
\varepsilon_1 + \text{Ref}_{g_r} U_1' & \leq \text{Ref}_{g_1} U_1 \\
\varepsilon_2 + U_2 & \leq U_1 \\
\phi & \in T[\text{Ref}_{g_r} U_1'] \\
\phi & \in T[U_2] \Rightarrow \\
\phi \varepsilon_1 + \phi \varepsilon_2 & \leq \phi \varepsilon_1 + \phi \varepsilon_2 \text{ for some } \varepsilon_1, \varepsilon_2 \in \text{EvTerm} \cup \{\text{error}\} \\
\phi & \varepsilon_1 t'' & = \varepsilon_{\ell_t} \varepsilon_{\ell_t} t'' & \in T[\text{Unit}_1] \\
\end{align*}
\]

By induction hypothesis on \(t''\), one of the following holds:

1. \(t''\) is a value \(U''\) (by canonical forms Lemma 6.10), where \(U_1'' = \text{Ref}_g U_1''\). Then by induction hypothesis on \(t''\), one of the following holds:

   a. \(t''\) is a value \(u\), then by (R\(\rightarrow\)), \(t'' \phi \rightarrow r\) and \(r \in \text{Config}_U \cup \{\text{error}\}\) by Prop 6.13. Also by Prop 6.13, if \(r = t''U\) \(\phi \rightarrow r\) \(\text{dom}(\mu) \subseteq \text{dom}(\mu')\).

   b. \(t''\) is an ascribed value \(v\), then, \(\varepsilon_t t'' \rightarrow c\) et' for some \(et' \in \text{EvTerm} \cup \{\text{error}\}\). Hence \(t'' \phi \rightarrow r\) for some \(r \in \text{Config}_U \cup \{\text{error}\}\) by Prop 6.13 and either (Rg), or (Rgerr).

   c. \(t''\) \(\mu\) \(\phi\) \(r_2\) for some \(r_2 \in \text{Config}_{U''} \cup \{\text{error}\}\). Hence \(t'' U\) \(\mu \phi \rightarrow r\) 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 \(\text{dom}(\mu) \subseteq \text{dom}(\mu')\).

2. \(t''\) is an ascribed value \(v\), then, \(\varepsilon_t t'' \rightarrow c\) et' for some \(et' \in \text{EvTerm} \cup \{\text{error}\}\). Hence \(t'' \phi \rightarrow r\) for some \(r \in \text{Config}_U \cup \{\text{error}\}\) by Prop 6.13 and either (Rg), or (Rgerr).

3. \(t''\) \(\mu\) \(\phi\) \(r_1\) for some \(r_1 \in \text{Config}_{U''} \cup \{\text{error}\}\). Hence \(t'' U\) \(\mu' \in T[U]\) 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 \(t\in \text{StaticTerm}, \phi = \langle e', \ell_c \rangle\), and \(\mu_s\), such that \(e \in \phi'[\ell_c' \leq \ell_c], \phi \triangleright t\in T[S]\), and that \(\forall v\in \text{cod}(\mu_s), v\in \text{StaticTerm}\). Then \(t\) is either a value, or

\[t_s | \mu_s \phi \rightarrow t_s' | \mu_s'.\]

**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 \(\phi.\varepsilon \leq \ell_l\). 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:

\[
\begin{align*}
\epsilon_1 \vdash \text{Ref}_{\ell'} S'_1 &\leq \text{Ref}_\ell S_1 & \phi \triangleright o^\ell_{S_1} \in T[\text{Ref}_{\ell'} S'_1] \\
\epsilon_2 \vdash S_2 &\leq S_1 & \phi \triangleright u \in T[S_2] \\
\phi \triangleright \ell_1' &\preceq \ell_2 & \phi \triangleright \ell_2 \leq \text{label}(S_1) \\
\phi \triangleright \ell_1' \otimes S'_1 :\ell_2 S_1 &\leq \phi \triangleright \ell_2 u \in T[\text{Unit}]_1
\end{align*}
\]

By inspection of the reduction rules we have to prove that $\phi \triangleright [\leq] \text{ilbl}(\epsilon)$. $\phi \triangleright [\leq] \text{ilbl}(\epsilon)$. We know by definition of interior between two static labels that $\epsilon = g[\ell_1' \preceq \ell_2] = ([\ell'_1, \ell'_2], [\ell_2, \ell_2])$. Also, if $\mu_S(o^{S_1}) = \varepsilon u' : S'_1$, as everything is static, $\text{ilbl}(\epsilon)$ have to have the form $([\ell_u, \ell_u], \text{label}(S'_1), \text{label}(S'_2))$, for some $\ell_u$. Then we have to prove that $\ell_2 \leq \text{label}(S'_1)$, but notice that as everything is static, $\ell_2 \leq \text{label}(S'_1)$ is equivalent to $\ell_2 \leq \text{label}(S_1)$, therefore we know that $\ell_2 \leq \text{label}(S_1)$ and the result holds.

\[\Box\]

### 6.5 Dynamic Gradual Guarantee

In this section we present the proof the Dynamic Gradual Guarantee for GSLRef without the specific check in rule (r7).

**Definition 6.16 (Intrinsic term precision).** Let $\Omega \in \mathcal{P}(\mathcal{V}[\ast] \times \mathcal{V}[\ast]) \cup \mathcal{P}(\mathcal{Loc}_c \times \mathcal{Loc}_c)$ be defined as $\Omega := \{ x^{U_1} \subseteq x^{U_2}, o^{U_1} \subseteq o^{U_2} \}$ We define an ordering relation $(\cdot \vdash \cdot \subseteq \cdot) \in (\mathcal{P}(\mathcal{V}[\ast] \times \mathcal{V}[\ast]) \cup \mathcal{P}(\mathcal{Loc}_c \times \mathcal{Loc}_c)) \times T[\ast] \times T[\ast]$ shown in Figure 29.

**Definition 6.17 (Well Formedness of $\Omega$).** We say that $\Omega$ is well formed iff $\forall \{ t^{U_1} \subseteq t^{U_2} \} \in \Omega, U_1 \subseteq U_2$

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 $\Omega \vdash t^{U_1} \subseteq t^{U_2}$ for some $\Omega \in \mathcal{P}(\mathcal{V}[\ast] \times \mathcal{V}[\ast]) \cup \mathcal{P}(\mathcal{Loc}_c \times \mathcal{Loc}_c)$, then $U_1 \subseteq U_2$.

**Proof.** Straightforward induction on $\Omega \vdash t^{U_1} \subseteq t^{U_2}$, since the corresponding precision on types is systematically a premise (either directly or transitively). $\Box$

**Proposition 6.19.** Let $g_1, g_2 \in EvFrame$ such that $\phi_1 \triangleright g_1[\epsilon_1 t^{U_1}_1] \in T[U'_1], \phi_2 \triangleright g_2[\epsilon_2 t^{U_1}_2] \in T[U'_2]$, with $U'_1 \subseteq U'_2$. Then if $g_1[\epsilon_1 t^{U_1}_1] \subseteq g_2[\epsilon_2 t^{U_1}_2], \epsilon_1 \subseteq \epsilon_2$ and $t^{U_1}_1 \subseteq t^{U_1}_2$, then $g_1[\epsilon_1 t^{U_1}_1] \subseteq g_2[\epsilon_2 t^{U_1}_2]$

**Proof.** We proceed by case analysis on $g_i$.

**Case** $(\square @^{U}_\ell et)$. Then for $i \in \{1, 2\} g_i$ must have the form $(\square @^{U'}_\ell \epsilon_i t^{U'}_i)$ for some $U'_i, \epsilon_i$ and $t^{U'}_i$. As $g_1[\epsilon_1 t^{U_1}_1] \subseteq g_2[\epsilon_2 t^{U_1}_2]$ then by $\subseteq_{\text{APP}} \epsilon_1 \subseteq \epsilon_2, \epsilon_1 \otimes \epsilon_2 \subseteq \epsilon_2, U'_1 \subseteq U'_2$ and $t^{U'_1} \subseteq t^{U'_2}$.

As $\epsilon_1 \subseteq \epsilon_2$ and $t^{U_1}_1 \subseteq t^{U_1}_2$, then by $\subseteq_{\text{APP}} \epsilon_1 t^{U_1}_1 \otimes \epsilon_2 t^{U_1}_2 \subseteq \epsilon_2 t^{U_1}_2$, and the result holds.

**Case** $(\square \triangleright^{g} et, ev \triangleright^{g} \square, ev \otimes^{g} \square : \square) : U, t^{U}, \square \triangleright^{g} U_1 et, ev \triangleright^{g} \ell, if^{g} \square$ then $et$ else $et)$. Straightforward using similar argument to the previous case. $\Box$
We proceed by case analysis on $\Omega \cup \{ x^U_1 \subseteq x^U_2 \} \vdash x^U_1 \subseteq x^U_2$

$g_1 \subseteq g_2$

$\Omega + b_{g_1} \subseteq b_{g_2}$

$g_1 \subseteq g_2$

$\Omega + \text{unit}_{g_1} \subseteq \text{unit}_{g_2}$

$U_{11} \subseteq U_{12}$

$g_{c_1} \subseteq g_{c_2}$

$g_1 \subseteq g_2$

$\Omega + (\lambda g_{c_1} x^U_{11}, t_{U_{12}})_{g_1} \subseteq (\lambda g_{c_2} x^U_{21}, t_{U_{22}})_{g_2}$

$g_{c_1} \subseteq g_{c_2}$

$\Omega + t_{U_{11}} \subseteq t_{U_{21}}$

$\Omega + t_{U_{12}} \subseteq t_{U_{22}}$

$\Omega + t_{U_{13}} \subseteq t_{U_{23}}$

$\Omega + \text{ref}^g_{\epsilon_1} \epsilon_{1} \cdot t_{U_{11}} \subseteq \text{ref}^g_{\epsilon_2} \epsilon_{2} \cdot t_{U_{12}}$

$\Omega + t_{U_{11}} \subseteq t_{U_{21}}$

$\Omega + t_{U_{12}} \subseteq t_{U_{22}}$

$\Omega + t_{U_{13}} \subseteq t_{U_{23}}$

$\forall \alpha_{U_1} \in \text{dom}(\mu_1), \exists \alpha_{U_2} \in \text{dom}(\mu_2)$ s.t.

$\Omega + \mu_1(\alpha_{U_1}) \subseteq \mu_2(\alpha_{U_2})$

where $\phi_1 \subseteq \phi_2 \iff \phi_1 \cdot \epsilon \subseteq \phi_2 \cdot \epsilon \land \phi_1 \cdot g_\epsilon \subseteq \phi_2 \cdot g_\epsilon \land \phi_1 \cdot g_c \subseteq \phi_2 \cdot g_c$

Fig. 29. Intrinsic term precision

**Proposition 6.20.** Let $g_1, g_2 \in \text{EvFrame}$ such that $\phi_1 \circ g_1[\epsilon_{1} \cdot t_{U_{11}}] \in T[U']_1$, $\phi_2 \circ g_2[\epsilon_{2} \cdot t_{U_{12}}] \in T[U']_2$, with $U'_{1} \subseteq U'_{2}$. Then if $g_1[\epsilon_{1} \cdot t_{U_{11}}] \subseteq g_2[\epsilon_{2} \cdot t_{U_{12}}]$ then $t_{U_{1}} \subseteq t_{U_{2}}$ and $\epsilon_{1} \subseteq \epsilon_{2}$.

**Proof.** We proceed by case analysis on $g_1$. 
Case (□ ⊢^U et). Then there must exist some ε_1, U_1, ε'_1 and t^U_1 such that g[ε_1 t^U_1] = ε_1 t^U_1 ⊑ ε'_1 t^U'_1 and g[ε_2 t^U_2] = ε_2 t^U_2 ⊑ ε'_2 t^U'_2. Then by the hypothesis and the premises of (⊆_APP), t^U_1 ⊑ t^U_2 and ε_1 ⊑ ε_2, and the result holds immediately.

Case (□ ⊖^Φ et, ev ⊖^Φ □, ev ⊖^U □, □ :: U_1, t^U_1, □ ⇒^U_1 et, ev' := ε_1 □, if^Φ □ then et else et). Straightforward using similar argument to the previous case.

□

**Proposition 6.21.** Let f_1, f_2 ∈ EvFrame such that φ_1 ⊑ f_1[t^U_1] ∈ T[U'_1], φ_2 ⊑ f_2[t^U_2] ∈ T[U'_2], with U'_1 ⊑ U'_2. Then if f_1[t^U_1] ⊑ f_2[t^U_2] and t^U_1 ⊑ t^U_2, then f_1[t^U_1] ⊑ f_2[t^U_2].

**Proof.** Suppose f_1[t^U_1] = g_1[ε_1 t^U_1]. We know that φ_1 ⊑ g_1[ε_1 t^U_1] ∈ T[U'_1], φ_2 ⊑ g_2[ε_2 t^U_2] ∈ T[U'_2] and U'_1 ⊑ U'_2. Therefore if g_1[ε_1 t^U_1] ⊑ g_2[ε_2 t^U_2], by Prop 6.20, ε_1 ⊑ ε_2. Finally by Prop 6.19 we conclude that g_1[ε_1 t^U_1] ⊑ g_2[ε_2 t^U_2].

□

**Proposition 6.22.** Let f_1, f_2 ∈ EvFrame such that φ_1 ⊑ f_1[t^U_1] ∈ T[U'_1], φ_2 ⊑ f_2[t^U_2] ∈ T[U'_2], with U'_1 ⊑ U'_2. Then if f_1[t^U_1] ⊑ f_2[t^U_2] then t^U_1 ⊑ t^U_2.

**Proof.** Suppose f_1[t^U_1] = g_1[ε_1 t^U_1]. We know that φ_1 ⊑ g_1[ε_1 t^U_1] ∈ T[U'_1], φ_2 ⊑ g_2[ε_2 t^U_2] ∈ T[U'_2] and U'_1 ⊑ U'_2. Therefore if g_1[ε_1 t^U_1] ⊑ g_2[ε_2 t^U_2], then using Prop 6.20 we conclude that t^U_1 ⊑ t^U_2.

□

**Proposition 6.23 (Substitution preserves precision).** If Ω ∪ {x^U_1 ⊑ x^U_2} ⊢ t^U_1 ⊑ t^U_2 and Ω ⊢ t^U_3 ⊑ t^U_4, then Ω ⊢ [t^U_1 / x^U_1]t^U_3 ⊑ [t^U_1 / x^U_2]t^U_4.

**Proof.** By induction on the derivation of t^U_1 ⊑ t^U_2 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 ⊑_^<).** If ε_1 ⊑ ε_2 and ε_3 ⊑ ε_4 then ε_1 ⊑^< ε_3 ⊑^< ε_2 ⊑^< ε_4.

**Proof.** By definition of consistent transitivity for ⊑_^< and the definition of precision.

□

**Proposition 6.25 (Monotone precision for ⊑^≤).** If ε_1 ⊑ ε_2 and ε_3 ⊑ ε_4 then ε_1 ⊑^≤ ε_3 ⊑^≤ ε_2 ⊑^≤ ε_4.

**Proof.** By definition of consistent transitivity for ⊑^≤ and the definition of precision.

□

**Proposition 6.26 (Monotone precision for join).** If ε_1 ⊑ ε_2 and ε_3 ⊑ ε_4 then ε_1 ▷ ε_3 ⊑ ε_2 ▷ ε_4.

**Proof.** By definition of join and the definition of precision.

□

**Proposition 6.27.** If Ref U_1 ⊑ Ref U_2 then U_1 ⊑ U_2.

**Proof.** By definition of precision we know that \{ Ref T | T ∈ γ(U_1) \} ⊑ \{ Ref T | T ∈ γ(U_2) \}. This relation is true only if γ(U_1) ⊑ γ(U_2) which is equivalent to U_1 ⊑ U_2.

□

**Proposition 6.28.** If U_{11} ⊑ U_{12} and U_{21} ⊑ U_{22} then U_{11} ▷ U_{21} ⊑ U_{12} ▷ U_{22}.

**Proof.** By induction on the type derivation of the types and consistent join.

□
LEMMA 6.29. If \( \xi_1 \vdash \text{Ref}_{\gamma_1} U_{11} \preceq \text{Ref}_{\gamma_1} U_{12} \) and \( \xi_2 \vdash \text{Ref}_{\gamma_2} U_{21} \preceq \text{Ref}_{\gamma_2} U_{22} \), and \( \xi_1 \subseteq \xi_2 \), then \( \text{iref}(\xi_1) \subseteq \text{iref}(\xi_2) \).

PROOF. By definition of precision and \( \text{iref} \).

\( \square \)

PROPOSITION 6.30 (Dynamic guarantee for \( \rightarrow \)). Suppose \( \Omega \vdash t^{U_1}_{1} \subseteq t^{U_2}_{1}, \phi_1 \subseteq \phi_2, \) and \( \Omega \vdash \mu_1 \subseteq \mu_2. \) If \( \xi^{U_1}_1 \mid \xi^{U_2}_1 \mid \mu^{\phi_1}_1 \) then \( \xi^{U_2}_1 \mid \mu^{\phi_2}_2 \) where \( \Omega' \vdash t^{U_1}_2 \subseteq t^{U_2}_2 \) and \( \Omega' \vdash \mu'_1 \subseteq \mu'_2 \), for some \( \Omega' \supseteq \Omega. \)

PROOF. By induction on the structure of \( t^{U_1}_1 \subseteq t^{U_2}_1 \). For simplicity we omit the \( \Omega \vdash \) notation on precision relations when it is not relevant for the argument.

Case \( (\rightarrow \oplus) \). We know that \( t^{U_1}_1 = (\xi^{U_1}_1(b_{1,1}) \oplus \xi^{U_1}_1(b_{2,1}) \oplus \xi^{U_1}_1(b_{2,2})) \) then by \( (\Xi) \) \( t^{U_2}_1 = (\xi^{U_2}_1(b_{1,1}) \oplus \xi^{U_1}_2(b_{2,1}) \oplus \xi^{U_1}_2(b_{2,2})) \) for some \( \xi^{U_1}_1, \xi^{U_2}_1, g_{21}, g_{22} \) such that \( \xi^{U_1}_1 \subseteq \xi^{U_1}_2, \xi^{U_1}_2 \subseteq \xi^{U_2}, g_{11} \subseteq g_{21} \) and \( g_{12} \subseteq g_{22} \).

If \( t^{U_1}_1 \mid \mu^{\phi_1}_1 \) \( b_3 \mid \mu_1 \) where \( b_3 = (\xi^{U_1}_1 \setminus \xi^{U_1}_2)(b_1 \oplus b_2)_{(g_{11},g_{21})} : \text{Bool}_{g_{11}} \), then

\[
\begin{align*}
\Omega \vdash (b_1 \oplus b_2)_{(g_{11},g_{21})} \subseteq (b_1 \oplus b_2)_{(g_{11},g_{21})} \\
\text{Bool}_{g_{11}} \subseteq \text{Bool}_{g_{21}} \subseteq (\xi^{U_1}_1 \setminus \xi^{U_1}_2) \subseteq (\xi^{U_1}_1 \setminus \xi^{U_1}_2) \\
(\xi^{U_1}_1 \setminus \xi^{U_1}_2)(b_1 \oplus b_2)_{(g_{11},g_{21})} : \text{Bool}_{g_{11}} \\
(\xi^{U_1}_1 \setminus \xi^{U_1}_2)(b_1 \oplus b_2)_{(g_{11},g_{21})} : \text{Bool}_{g_{21}}.
\end{align*}
\]

Therefore \( t^{U_2}_1 \subseteq t^{U_2}_2 \). As \( \Omega' = \Omega, \mu'_1 = \mu_1 \) and \( \mu_2' = \mu_2 \) then \( \Omega' \vdash \mu'_1 \subseteq \mu'_2 \).

Case \( (\rightarrow \text{prot}) \). We know that \( t^{U_1}_1 = \text{prot}_{\xi^{U_1}_1} \phi'_1(\xi^{U_1}_1) \), then by \( (\Xi) \) \( t^{U_2}_1 = \text{prot}_{\xi^{U_2}_1} \phi'_2(\xi^{U_2}_1) \), and therefore

\[
\begin{align*}
g'_1 \subseteq g'_2 \\
\phi'_1 \subseteq \phi'_2 \\
\xi^{U_1}_1 \subseteq U_2 \\
\Omega + \mu_1 \subseteq \mu_2 \subseteq \xi^{U_1}_2 \subseteq \xi^{U_2}_1 \subseteq \text{prot}_{\xi^{U_1}_1} \phi'_1(\xi^{U_1}_1) \subseteq \text{prot}_{\xi^{U_2}_1} \phi'_2(\xi^{U_2}_1)
\end{align*}
\]

for some \( \xi^{U_1}_2, \xi^{U_2}_2, g_2, g_2' \), where \( u_1 \in T[U^{U_1}_1] \) and \( u_2 \in T[U^{U_2}_2] \). If

\[
\begin{align*}
t^{U_1}_1 \mid \mu_1 \mid \phi^{\xi^{U_1}_1}(u_1 \triangledown g'_1) :: U_1 \triangledown g_1 \mid \mu_1. \text{Therefore, } t^{U_2}_1 \mid \mu_2 \mid \phi^{\xi^{U_2}_1}(u_2 \triangledown g'_2) :: U_2 \triangledown g_2 \mid \mu_2.
\end{align*}
\]

By Lemma 6.26, \( (\xi^{U_1}_1 \setminus \xi^{U_1}_2) \subseteq (\xi^{U_2}_1 \setminus \xi^{U_2}_2) \), and as join is monotone \( U_1 \triangledown g_1 \subseteq U_2 \triangledown g_2 \) and \( (u_1 \triangledown g'_1) \subseteq (u_2 \triangledown g'_2) \).

Therefore by \( \Xi \), \( (\xi^{U_1}_1 \setminus \xi^{U_1}_2)(u_1 \triangledown g'_1) :: U_1 \triangledown g_1 \subseteq (\xi^{U_2}_1 \setminus \xi^{U_2}_2)(u_2 \triangledown g'_2) :: U_2 \triangledown g_2 \). As \( \Omega' = \Omega, \mu'_1 = \mu_1 \) and \( \mu_2 = \mu'_2 \) then \( \Omega' \vdash \mu'_1 \subseteq \mu'_2 \).

Case \( (\rightarrow \text{app}) \). We know that

\[
\begin{align*}
t^{U_1}_1 = (\lambda x^{U_1}_1. t^{U_1}_1) \phi^{\xi^{U_1}_1}(\xi^{U_1}_1) \mid \phi^{\xi^{U_1}_1}(\xi^{U_1}_1) u_1 \text{ then by } (\Xi) \text{ } t^{U_1}_1 \text{ must have the form }
\end{align*}
\]

\[
\begin{align*}
t^{U_1}_1 = (\lambda x^{U_1}_1. t^{U_1}_1) \phi^{\xi^{U_1}_1}(\xi^{U_1}_1) \phi^{\xi^{U_1}_1}(\xi^{U_1}_1) u_1 \text{ for some } \xi^{U_1}_1, x^{U_1}_1, t^{U_1}_1, U_3, \xi^{U_1}_1, g_2, g_2' \text{ and } u_2.
\end{align*}
\]

Let us pose \( \xi_1 = \xi^{U_1}_1 \oslash \text{idm}(\xi^{U_1}_1) \) and \( \xi'_1 = (\phi^{\xi^{U_1}_1} \triangledown \text{ilbl}(\xi^{U_1}_1)) \oslash \xi^{U_1}_1 \oslash \text{ilat}(\xi^{U_1}_1), \phi'_1 = (\phi^{\xi^{U_1}_1} \triangledown \phi^{\xi^{U_1}_1} \triangledown \phi^{\xi^{U_1}_1} \text{ and } g_1 \triangledown \phi^{\xi^{U_1}_1} \text{ and } g_1 \triangledown \phi^{\xi^{U_1}_1}). \)

Then

\[
\begin{align*}
t^{U_1}_1 \mid \mu_1 \mid \phi^{\xi^{U_1}_1}(\xi^{U_1}_1) \phi^{\xi^{U_1}_1}(\xi^{U_1}_1) \text{ with } \mu_1 \text{ and } t_1 = [(\xi^{U_1}_1 :: U_1) \oslash x^{U_1}_1] \text{ app}. \text{ Also, let us pose } \xi_2 = \xi^{U_2}_1 \oslash \text{idm}(\xi^{U_2}_1) \text{ and } \xi'_2 = (\phi^{\xi^{U_2}_1} \triangledown \text{ilbl}(\xi^{U_2}_1)) \oslash \xi^{U_2}_1 \oslash \text{ilat}(\xi^{U_1}_1) \phi'_2 = \text{ and } g_2 \text{ and } u_2.
\end{align*}
\]
\[ \langle e'_2(g'_2 \triangleright \phi_2, g_c), g_2 \triangleright \phi_2, g_c \rangle. \]

Then

\[ t_1^U \mid \mu_2 \xrightarrow{\phi_2} \text{prot}_{\text{ibbl}(\varepsilon_1)} g'_2 \text{idoc}(\varepsilon_2) t'_2 \mid \mu_2 \text{ with } t'_2 = [[\varepsilon_2 u_2 : U_2] \cup x'_{U_2}] t'_{U_2}. \]

As \( \Omega \vdash t_{U_1}^1 \subseteq t_{U_1}^2 \), then \( u_1 \subseteq u_2, \varepsilon_{12} \subseteq \varepsilon_{22} \) and \( \text{idom}(\varepsilon_{11}) \subseteq \text{idom}(\varepsilon_{21}) \) as well, then by Prop. 6.24 \( \varepsilon_1 \subseteq \varepsilon_2 \). Then \( \varepsilon_1 u_1 : U_1 \subseteq \varepsilon_2 u_2 : U_2 \) by \( \varepsilon_2 \).

We also know by \( (\varepsilon_2 \cup \{ u_{U_2} \}) \vdash t_{U_2} \subseteq t_{U_2} \cup \{ u_{U_2} \} \) by Substitution preserves precision (Prop 6.23) \( \varepsilon'_2 \subseteq \varepsilon'_2 \) therefore \text{idoc}(\varepsilon_{11}) t'_1 \subseteq U_1 \subseteq \text{idoc}(\varepsilon_{21}) t'_2 \subseteq U_2 \) by \( \varepsilon_2 \). Also \( g_1 \subseteq g_2, \)
\[ \text{ibbl}(\varepsilon_{11}) \subseteq \text{ibbl}(\varepsilon_{21}) \text{ and } g'_1 \subseteq g'_2 \text{ and by Lemma 6.24 and 6.26, } \varepsilon'_1 \subseteq \varepsilon'_2. \]

Also, as \( \phi_1, g_c \subseteq \phi_2, g_c \) by monotonicity of the join \( g_1 \triangleright \phi_1, g_c \subseteq g_2 \triangleright \phi_2, g_c \) and as \( \phi_1, g_c \subseteq \phi_2, g_c \) also by monotonicity of the join \( g'_1 \triangleright \phi_1, g_c \subseteq g'_2 \triangleright \phi_2, g_c \). Then by \( (\varepsilon_1 \text{prot}) \) \( t_{U_1}^1 \subseteq t_{U_2}^2 \). As \( \Omega' = \Omega, \mu'_1 = \mu_1 \) and \( \mu_2 = \mu'_2 \) then \( \Omega' + \mu'_1 \subseteq \mu'_2 \).

\text{Case (\textit{\texttt{if-true}}). } t_{U_1}^1 \text{ is } \text{if}^\varepsilon_{11} \varepsilon_{11} \text{true}_{\phi_1} \text{ then else } \varepsilon_{12} \text{true}_{\phi_2} \text{ then by } (\varepsilon_1 \text{if}) \text{ } t_{U_1}^1 \text{ has the form}

\[ t_{U_1}^1 = \text{if}^\varepsilon_{12} \text{true}_{\phi_1} \text{ then else } \varepsilon_{12} \text{true}_{\phi_2} \text{ then by } (\varepsilon_1 \text{if}) \text{ } t_{U_1}^1 \text{ has the form}

\text{Case (\textit{\texttt{if-false}}). Same as case \textit{\texttt{if-true}}, using the fact that } \varepsilon_{13} \subseteq \varepsilon_{23} \text{ and } t_{U_{13}} \subseteq t_{U_{23}}.

\text{Case (\textit{\texttt{ref}}). We know that } t_{U_1}^1 = \text{ref}_{\varepsilon_1}^U \varepsilon_{11} u_1, \text{ then by } (\varepsilon_2 \text{ref}) \text{ } t_{U_2}^1 = \text{ref}_{\varepsilon_2}^U \varepsilon_{22} u_2 \text{ and therefore}

\[ U_{U_1}^1 \subseteq U_{U_2}^1 \]

\[ \Omega \vdash U_{U_1}^1 \]

\[ \Omega \vdash \text{ref}_{\varepsilon_1}^U \varepsilon_{11} u_1 \subseteq \text{ref}_{\varepsilon_2}^U \varepsilon_{22} u_2 \text{ for some}

\text{for some } \varepsilon_{23} u_2, U_{U_1}^1 \text{ and } \varepsilon_{22} u_2, \text{ where } u_1 \subseteq \{ U_{U_1}^1 \} \text{ and } u_2 \subseteq \{ U_{U_2}^1 \}.

\text{If}

\[ t_{U_1}^1 \mid \mu_1 \xrightarrow{\phi_1} \text{ref}_{\varepsilon_1}^U \varepsilon_{11} \mid \mu_1 \] for some \( \varepsilon_{11} U_{U_1}^1 \subseteq U_{U_1}^1 \), where \( \varepsilon'_{11} = \varepsilon_{11} U_{U_1} \rightarrow U_{U_1}^1 \rightarrow U_{U_1}^1 \), and hence

\[ U_{U_1}^1 \subseteq U_{U_2}^1 \]

\[ \Omega \vdash U_{U_1}^1 \]

\[ \Omega \vdash \text{ref}_{\varepsilon_1}^U \varepsilon_{11} u_1 \subseteq \text{ref}_{\varepsilon_2}^U \varepsilon_{22} u_2 \]

\text{Case (\textit{\texttt{deref}}). We know that } t_{U_1}^1 = \text{deref}_{\varepsilon_1}^U \varepsilon_{11} U_{U_1}^1 \text{, } t_{U_1}^2 = \text{deref}_{\varepsilon_2}^U \varepsilon_{22} U_{U_2}^2 \text{ and so}

\text{As } \Omega \vdash \mu_1 \subseteq \mu_2, \text{ using } (\varepsilon_1 \mu) \text{ then } \Omega + \mu_1 (U_{U_1}^1) \subseteq \mu_2 (U_{U_2}^2). \text{ Then}

\text{As } \Omega \vdash \mu_1 \subseteq \mu_2, \text{ using } (\mu) \text{ then } \Omega + \mu_1 (U_{U_1}^1) \subseteq \mu_2 (U_{U_2}^2). \text{ Then}

\text{As } \Omega \vdash \mu_1 \subseteq \mu_2, \text{ using } (\mu) \text{ then } \Omega + \mu_1 (U_{U_1}^1) \subseteq \mu_2 (U_{U_2}^2). \text{ Then}

\text{As } \Omega \vdash \mu_1 \subseteq \mu_2, \text{ using } (\mu) \text{ then } \Omega + \mu_1 (U_{U_1}^1) \subseteq \mu_2 (U_{U_2}^2). \text{ Then}
Where $\phi'_1 = ((\phi_1 \cdot \nu \nu \epsilon_1'))(\phi_1 \cdot g_1 \nu \nu \phi'_1), (\phi_1 \cdot g_1 \nu \nu \phi'_1)$. By monotonicity of the join $\phi_1 \cdot g_1 \nu \nu g_1 \subseteq \phi_2 \cdot g_1 \nu \nu g_2$, $\phi_1 \cdot g_1 \nu \nu g_1' \subseteq \phi_2 \cdot g_1 \nu \nu g_2'$ and $((\phi_1 \cdot \nu \nu \epsilon_1))(\phi_1 \cdot \nu \nu \epsilon_1') \subseteq (\phi_2 \cdot \nu \nu \epsilon_2').$ As $\epsilon_1 \subseteq \epsilon_2$, then by Lemma 6.29, $\text{iref}(\epsilon_1) \subseteq \text{iref}(\epsilon_2)$. Thus $\text{Use}((\epsilon_{\text{prot}}))$ we can conclude that $\Omega + t_{U_1}^{U_1} \subseteq t_{U_2}^{U_2}$. As $\Omega' = \Omega, \mu_1 = \mu'_1$ and $\mu_2 = \mu'_2$ then also $\Omega' + \mu'_1 \subseteq \mu'_2$.

**Case (→-assign).** We know that $t_{U_1}^{U_1} = \epsilon_{U_1}^{U_1} = \epsilon_{U_2}^{U_2} = \epsilon_{U_1}^{U_2} = \epsilon_{U_2}^{U_1} = \epsilon_{U_2}^{U_2} = \epsilon_{U_1}^{U_2}$ and so $\Omega + \epsilon_{U_1}^{U_1} = \epsilon_{U_2}^{U_2} \subseteq \epsilon_{U_1}^{U_2} \subseteq \epsilon_{U_1}^{U_2} \subseteq \epsilon_{U_2}^{U_2} \subseteq \epsilon_{U_1}^{U_2} \subseteq \epsilon_{U_2}^{U_2}$. Then

$$t_{U_1}^{U_1} \mid \mu_1 \rightarrow \phi_1 \rightarrow t_{U_2}^{U_2} \mid \mu'_1 \rightarrow t_{U_1}^{U_2} \mid \mu'_1 \rightarrow t_{U_2}^{U_2} \mid \mu'_2$$

where $t_{U_1}^{U_1} \subseteq t_{U_2}^{U_2}$ and $\mu_1 \subseteq \mu_2$.

**Proposition 6.31 (Dynamic Guarantee).** Suppose $t_{U_1}^{U_1} \subseteq t_{U_2}^{U_2}, \phi_1 \subseteq \phi_2, \text{and } \mu_1 \subseteq \mu_2$. If $t_{U_1}^{U_1} \mid \mu_1 \rightarrow \phi_1 \rightarrow t_{U_2}^{U_2} \mid \mu'_1$ then $t_{U_1}^{U_2} \mid \mu'_1 \rightarrow t_{U_2}^{U_2} \mid \mu'_2$ where $t_{U_1}^{U_2} \subseteq t_{U_2}^{U_2}$ and $\mu_1 \subseteq \mu_2$.

**Proof.** We prove the following property instead: Suppose $\Omega + t_{U_1}^{U_1} \subseteq t_{U_2}^{U_2}, \phi_1 \subseteq \phi_2,$ and $\Omega + \mu_1 \subseteq \mu_2$. If $t_{U_1}^{U_1} \mid \mu_1 \rightarrow \phi_1 \rightarrow t_{U_2}^{U_2} \mid \mu'_1$ then $t_{U_1}^{U_2} \mid \mu_2 \rightarrow \phi_2 \rightarrow t_{U_2}^{U_2} \mid \mu'_2$ where $\Omega' + t_{U_2}^{U_2} \subseteq t_{U_2}^{U_2}$ and $\Omega' + \mu'_1 \subseteq \mu'_2$, for some $\Omega' \supseteq \Omega$.

By induction on the structure of a derivation of $t_{U_1}^{U_1} \subseteq t_{U_2}^{U_2}$. For simplicity we omit the $\Omega$ notation on precision relations when it is not relevant for the argument.

**Case (→→).** $\Omega + t_{U_1}^{U_1} \subseteq t_{U_2}^{U_2}, \Omega + \mu_1 \subseteq \mu_2$ and

$$t_{U_1}^{U_1} \mid \mu_1 \rightarrow \phi_1 \rightarrow t_{U_2}^{U_2} \mid \mu'_1.$$ By dynamic guarantee of $\rightarrow$ (Prop 6.30), $t_{U_1}^{U_2} \mid \mu_2 \rightarrow \phi_2 \rightarrow t_{U_2}^{U_2} \mid \mu'_2$ where $\Omega' + t_{U_2}^{U_2} \subseteq t_{U_2}^{U_2}$ and $\Omega' + \mu'_1 \subseteq \mu'_2$, for some $\Omega' \supseteq \Omega$. And the result holds immediately.

**Case (Rf).** $t_{U_1}^{U_1} = f_1(t_{U_1}^{U_1})$, $t_{U_2}^{U_2} = f_2(t_{U_2}^{U_2})$. We know that $\Omega + f_1(t_{U_1}^{U_1}) \subseteq f_2(t_{U_2}^{U_2})$. By using Prop 6.18, $U_1' \subseteq U_2'$. By Prop 6.22, we also know that $\Omega + t_{U_1}^{U_1} \subseteq t_{U_2}^{U_2}$. By induction hypothesis, $t_{U_1}^{U_1} \mid \mu_1 \rightarrow t_{U_2}^{U_2} \mid \mu'_1 \rightarrow t_{U_1}^{U_2} \mid \mu'_1 \rightarrow t_{U_2}^{U_2} \mid \mu'_2$ where $\Omega' + t_{U_2}^{U_2} \subseteq t_{U_2}^{U_2}$ and $\Omega' + \mu'_1 \subseteq \mu'_2$, for some $\Omega' \supseteq \Omega$. Then by Prop 6.21, $\Omega' + f_2(t_{U_2}^{U_2}) \subseteq f_2(t_{U_2}^{U_2})$, and the result holds.

**Case (Rprot).** Then $t_{U_1}^{U_1} = \text{prot}_{\epsilon_1}^{U_1} \phi_1(\epsilon_{t_1}^{U_1})$ and $t_{U_1}^{U_2} = \text{prot}_{\epsilon_2}^{U_1} \phi_2(\epsilon_{t_2}^{U_2})$

As $t_{U_1}^{U_1} \subseteq t_{U_2}^{U_2}$ then by $\text{Use}(\epsilon_{\text{prot}}))$, $t_{U_1}^{U_1} \subseteq t_{U_2}^{U_2}, \phi_1 \subseteq \phi_2, \epsilon_{t_1} \subseteq \epsilon_{t_2}, g_1 \subseteq g_2, g_1' \subseteq g_2'$, and $\epsilon_1 \subseteq \epsilon_2$. By (Rprot), $t_{U_1}^{U_1} \mid \mu_1 \rightarrow \phi_1 \rightarrow \mu'$ and by induction hypothesis, $t_{U_2}^{U_2} \subseteq t_{U_2}^{U_2}$ and $\Omega' + \mu'_1 \subseteq \mu'_2$ for some $\Omega' \supseteq \Omega$.

But then by $\text{Use}(\epsilon_{\text{prot}}))$, $\Omega' + \text{prot}_{\epsilon_{t_1}^{U_1}} \phi_1(\epsilon_{t_1}^{U_1}) \subseteq \text{prot}_{\epsilon_{t_2}^{U_2}} \phi_2(\epsilon_{t_2}^{U_2})$, and the result holds.

**Case (Rg).** $t_{U_1}^{U_1} = g_1[et_1], t_{U_2}^{U_2} = g_2[et_2]$, where $\Omega + g_1[et_1] \subseteq g_2[et_2]$. Also $et_1 \rightarrow c \ e't_1$ and $et_2 \rightarrow c \ e't_2$. 


Then there exists $U_1$, $\varepsilon_{11}$, $\varepsilon_{12}$ and $v_1$ such that $e_1 = \varepsilon_{11}(\varepsilon_{12}v_1 :: U_1)$. Also there exists $U_2$, $\varepsilon_{21}$, $\varepsilon_{22}$ and $v_2$ such that $e_2 = \varepsilon_{21}(\varepsilon_{22}v_2 :: U_2)$. 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 $e_1 \to_c (\varepsilon_{12} \circ < \varepsilon_{11})v_1$ and $e_2 \to_c (\varepsilon_{22} \circ < \varepsilon_{21})v_2$ then, by Prop 6.24 we know that $\varepsilon_{12} \circ < \varepsilon_{11} \sqsubseteq \varepsilon_{22} \circ < \varepsilon_{21}$. Then using this information, and the fact that $v_1 \sqsubseteq v_2$, by Prop 6.19, it follows that $\Omega + g_1[e_1'] \sqsubseteq g_1[e_2']$. As $\Omega' = \Omega$, $\mu_1' = \mu_1$ and $\mu_2 = \mu_2'$ then $\Omega' + \mu_1' \sqsubseteq \mu_2'$.

Case (Rprotg). Analogous to (Rprot) case but using $\to_c$ instead.

□
6.6 Noninterference

In this section we present the proof of noninterference for GSLRef. We use a logical relation that is more general than the one presented in the paper. The main difference (beyond 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 6.64), 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 \( rval(v) \) operator, which strips away any checking-related information like labels or evidence-carrying ascriptions:

\[
rval : \text{Value} \to \text{RawValue}
\]

\[
rval(b) = b
\]

\[
rval(\ell b :: U) = b
\]

\[
rval(\text{unit}) = \text{unit}
\]

\[
rval(\ell \text{unit}) :: U = \text{unit}
\]

\[
rval(\ell o :: U) = o
\]

\[
rval(\ell o U :: U) = o
\]

\[
rval(\ell (\lambda^g x^{U_1}.t^{U_2})_g :: U) = (\lambda^g x^{U_1}.t^{U_2})
\]

\[
rval(\ell (\lambda^g x^{U_1}.t^{U_2})_g :: U) = (\lambda^g x^{U_1}.t^{U_2})
\]

Definition 6.32 (Gradual security logical relations). For an arbitrary element \( \ell_\alpha \) of the security lattice, the \( \ell_\alpha \)-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 \( \langle \phi_1, v_1, \mu_1 \rangle \approx_{\ell_\alpha}^k \langle \phi_2, v_2, \mu_2 \rangle : U \) indicates that the tuple of security effect \( \phi_1 \), value \( v_1 \) and store \( \mu_1 \) is related to the tuple of security effect \( \phi_2 \), value \( v_2 \) and store \( \mu_2 \) at type \( U \) for \( k \) steps when observed at the security level \( \ell_\alpha \). Similarly, the notation \( \langle \phi_{r\ell o}, t_{r\ell o}, \mu_{r\ell o} \rangle \approx_{\ell_\alpha}^k \langle \phi_2, t_2, \mu_2 \rangle C(U) \) 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 \( \ell_\alpha \). Notation \( \mu_1 \approx_{\ell_\alpha}^{k} \mu_2 \) relates stores \( \mu_1 \) and \( \mu_2 \) for \( k \) steps when observed at security level \( \ell_\alpha \). Finally, notation \( \phi_1 \approx_{\ell_\alpha} \phi_2 \), relates security effect \( \phi_1 \) and \( \phi_2 \) for any number of steps at security level \( \ell_\alpha \).

We say that a value is observable at level \( \ell_\alpha \) if, given a security effect \( \phi \), the value is typeable, the security effect is observable, and the label of the value is sublabel of \( \ell_\alpha \). Also, as value \( v \) 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
The level of the value and the bare value are sublabel of store the consistent transitivity operation of the reduction of the application does not hold, then it is not plausible that the security level of argument of a function that expects a value with security level $k$.

$$\langle \phi_1, v_1, \mu_1 \rangle \approx_{\ell_o}^k \langle \phi_2, v_2, \mu_2 \rangle : U \iff \phi_1 \approx_{\ell_o} \phi_2 \land \mu_1 \approx_{\ell_o}^k \mu_2 \land \phi_1 \triangleright v_1 \in T[U] \land$$

$$\left( \text{obs}_{\ell_o}(\phi_1 \triangleright v_1) \lor \lnot \text{obs}_{\ell_o}(\phi_1 \triangleright v_1) \right) \land$$

$$\left( \text{obs}_{\ell_o}(\phi_1 \triangleright v_1) \Rightarrow \text{obsRe}_{U, \ell_o}(\phi_1, v_1, \mu_1, \phi_2, v_2, \mu_2) \right)$$

$$\text{obsRe}_{U, \ell_o}(\phi_1, v_1, \mu_1, \phi_2, v_2, \mu_2) \iff (rval(v_1) = rval(v_2)) \text{ if } U \in \{\text{Bool}, \text{Unit}, \text{Ref}_g, U'\}$$

$$\text{obsRe}_{U, \ell_o}^{U_1} \triangleright_{g_1} \langle \phi_1, v_1, \mu_1, \phi_2, v_2, \mu_2 \rangle \iff \forall j \leq k. \forall U' = U'' \triangleright_{g_2}^{U_2}, U', \forall \phi'_1, \text{ s.t. } \phi_1 \leq_{\ell_o} \phi'_1,$$

$$\epsilon'_1 + U_1 \triangleright_{g_1} \langle \phi_1, v_1, \mu_1 \rangle \leq U', \text{ and } \epsilon'_2 + U'_1 \leq U''', \epsilon'_2 + \phi'_2 \triangleright g \wedge g_2 \leq g_2''', \text{ we have:}$$

$$\forall \phi'_1, \langle \phi_1, v_1, \mu_1 \rangle \approx_{\ell_o}^k \langle \phi_2, v_2, \mu_2 \rangle : U'_1, \text{ dom}(\mu_1) \subseteq \text{ dom}(\mu'_1),$$

$$\langle \phi_1, (\epsilon'_1 \triangleright v_1 @ U'_1) \triangleright_{g_1} \langle \phi_2, (\epsilon'_2 \triangleright v_2 @ U'_2) \triangleright_{g_2} \langle \phi_2, (\epsilon'_2 \triangleright v_2) : C(U''_2 \triangleright g_2)$$

$$\langle \phi_1, \mu_1 \rangle \approx_{\ell_o}^k \langle \phi_2, \mu_2 \rangle : C(U)$$

$$\iff \phi_1 \approx_{\ell_o} \phi_2 \wedge \mu_1 \approx_{\ell_o} \mu_2 \wedge \forall \phi_1', \text{ s.t. } \phi_1 \leq \phi_1' \text{ and } \phi_1' \triangleright t_1 \in T[U] \text{ we have } \forall j < k$$

$$\left( t_1 | \mu_1 \overset{\phi_1'}{\rightarrow} t_1' | \mu_1' \Rightarrow \mu_1' \approx_{\ell_o}^{j-k} \mu_2 \land \right.$$  

$$(\text{irred}(t_1') \Rightarrow \langle \phi_1, t_1', \mu_1' \rangle \approx_{\ell_o}^{j-k} \langle \phi_2, t_2', \mu_2' \rangle : U)$$

$$\mu_1 \approx_{\ell_o}^k \mu_2 \iff \forall \phi_1, \phi_2, j < k, \forall U \in \text{ dom}(\mu_1) \cap \text{ dom}(\mu_2)$$

$$\langle \phi_1 \triangleright \mu_1(o^U), \mu_1 \rangle \approx_{\ell_o}^k \langle \phi_2 \triangleright \mu_2(o^U), \mu_2 \rangle : U$$

$$\phi_1 \approx_{\ell_o} \phi_2 \iff \text{obs}_{\ell_o}(\phi_1, \phi_1, g_1) \lor \lnot \text{obs}_{\ell_o}(\phi_1, \phi_1, g_1)$$

$$\phi_1 \leq_{\ell_o} \phi_2 \iff \text{obs}_{\ell_o}(\phi_2, \phi_2, g_2) \Rightarrow \text{obs}_{\ell_o}(\phi_1, \phi_1, g_1)$$

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

$$\text{obs}_{\ell_o}(\phi \triangleright v) \iff \phi \triangleright v \in T[U] \land \text{obs}_{\ell_o}(\phi) \land \text{label}(U) \leq_{\ell_o} \left( (v = \varepsilon u :: U) \Rightarrow \text{obs}_{\ell_o}(\text{lbll}(\varepsilon)\text{label}(U)) \right)$$

$$\text{obs}_{\ell_o}(\varepsilon g) = \iff \varepsilon \circ \varepsilon' \text{ is defined, where } \varepsilon' = \mathcal{G}_g(\varepsilon, \ell_o)$$

Fig. 30. Gradual security logical relations

are observable, if any value with that underlying evidence and static label, can be used used as an 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 $\varepsilon = \langle [H, \top], [\bot, \top] \rangle$ and static label $g = \varepsilon$. We can construct any value such as $v = \varepsilon \text{true}_\varepsilon :: \text{Bool}_g$. 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 $L \ll H$), because as $\mathcal{G}_g(\varepsilon, \ell_o) = \langle [\bot, L], [L, L] \rangle$, the consistent transitivity operation $([H, \top], [\bot, \top]) \circ \varepsilon : ([\bot, L], [L, L])$ 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 \( \text{Unit} \) if the security effects are related, the stores are related for \( k \) steps, the values can be typed as \( \text{Bool}_g \) 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 \( \text{Unit}_g \) and \( \text{Ref}_g \) analogous to booleans.

Pairs are related at function types similarly to booleans. The difference is that functions can not be compared as booleans. Two functions are related if, given two related values and stores for \( j \leq k \) 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 an 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 \( \rho \vdash \Gamma \), 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 \( \Gamma \vdash \langle \phi_1, \rho_1, \mu_1 \rangle \approx_{k_\ell_o}^k \langle \phi_2, \rho_2, \mu_2 \rangle \), if \( \rho_1 \vdash \Gamma, \mu_1 \approx_{k_\ell_o}^k \mu_2 \) and

\[
\forall x^U \in \Gamma, \langle \phi_1, \rho_1(x^U), \mu_1 \rangle \approx_{k_\ell_o}^k \langle \phi_2, \rho_2(x^U), \mu_2 \rangle : U
\]

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 \( \text{FV}(t) = \{ x^{U_1} \} \) with label(\( U_1 \)) \( \not\subseteq \ell_o \)
- two compatible valid stores \( t^U \triangleright \rho_1, \mu_1 \approx_{k_\ell_o}^k \rho_2, \mu_2 \)

Then for any \( j < k, v_1, v_2 \in T[\text{Unit}_1] \), if both

- \( t^U[v_1/x^{U_1}] \triangleright \mu_1 \xrightarrow{\phi} \rho_1, \mu_1' \)
- \( t^U[v_2/x^{U_1}] \triangleright \mu_2 \xrightarrow{\phi} \rho_2, \mu_2' \)

we have that \( \text{rval}(v'_1) = \text{rval}(v'_2) \), and \( \rho_1' \approx_{k_\ell_o}^k \rho_2' \).

**Proof.** The result follows as a special case of Proposition 6.64 below. \( \square \)

In this theorem, we treat \( t^U \) as a program that takes \( x^{U_1} \) as its input. Furthermore, the security level \( g' = \text{label}(U_1) \) 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 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 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 know equality of raw values at first-order type. Restricting attention to first-order types (i.e., 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 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 error. This change in behavior falls under the notion of 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 \( t^S \mid \mu \xrightarrow{\phi}^k t'^S \mid \mu' \) to describe that configuration \( t^S \mid \mu \) reduces, in at most \( k \) steps, to configuration \( t'^S \mid \mu' \).

**Lemma 6.36.** Consider \( \varepsilon_1 \vdash g \preceq g' \). If \( \forall \varepsilon_2 \) such that \( \varepsilon_2 \vdash g' \preceq \ell_o, \varepsilon_1 \circ_{\varepsilon_2} \varepsilon_3 \vdash g \preceq \ell_o \) is not defined.

Then if \( \varepsilon_3 \vdash g' \preceq g'', \) then \( \forall \varepsilon_4 \) such that \( \varepsilon_4 \vdash g'' \preceq \ell_o \),

\( \varepsilon_1 \circ_{\varepsilon_2} \varepsilon_3 \circ_{\varepsilon_2} \varepsilon_4 \vdash g \preceq \ell_o \) is not defined.

**Proof.** Applying associativity: \( (\varepsilon_1 \circ_{\varepsilon_2} \varepsilon_3) \circ_{\varepsilon_2} \varepsilon_4 = \varepsilon_1 \circ_{\varepsilon_4} (\varepsilon_3 \circ_{\varepsilon_4} \varepsilon_4) \), but \( \varepsilon_3 \circ_{\varepsilon_4} \varepsilon_4 \vdash g'' \preceq \ell_o \), and we know that \( \varepsilon_1 \circ_{\varepsilon_2} \varepsilon_3 \) is not defined \( \forall \varepsilon_4 \) such that \( \varepsilon_4 \vdash g'' \preceq \ell_o \). Therefore \( (\varepsilon_1 \circ_{\varepsilon_2} \varepsilon_3) \circ_{\varepsilon_2} \varepsilon_4 \vdash g \preceq \ell_o \) is not defined and the result holds.

**Lemma 6.37.** Consider \( \varepsilon_1 \vdash g \preceq g' \). If \( \forall \varepsilon_2 \) such that \( \varepsilon_2 \vdash g' \preceq \ell_o, \varepsilon_1 \circ_{\varepsilon_2} \varepsilon_3 \vdash g \preceq \ell_o \) is not defined.

Also \( \varepsilon_0 \vdash g_1 \preceq g_2, \) if \( \varepsilon_3 \vdash g_2 \lor g' \preceq \ell_o, \) then \( (\varepsilon_0 \lor \varepsilon_1) \circ_{\varepsilon_2} \varepsilon_3 \vdash g_1 \lor g \preceq \ell_o \) is not defined.

**Proof.** Let us prove that if \( (\varepsilon_0 \lor \varepsilon_1) \circ_{\varepsilon_2} \varepsilon_3 \vdash g_1 \lor g \preceq \ell_o \) is defined, then \( \varepsilon_1 \circ_{\varepsilon_2} \varepsilon_3 \) is defined.

As \( \lor \) is join monotone \( \exists \varepsilon_0' \) such that \( \varepsilon_0' \vdash g' \preceq g_2 \lor g' \).

Suppose \( \varepsilon_1 = (\langle \ell_{11}, \ell_{12} \rangle, [\ell_{21}, \ell_{22}]), \varepsilon_0 = (\langle \ell_{31}, \ell_{32} \rangle, [\ell_{41}, \ell_{42}]), \varepsilon_0' = (\langle \ell_{51}, \ell_{52} \rangle, [\ell_{61}, \ell_{62}]), \) and \( \varepsilon_3 = (\langle \ell_{71}, \ell_{72} \rangle, [\ell_{81}, \ell_{82}]). \)

As \( \varepsilon_0 \lor \varepsilon_1 = (\langle \ell_{11} \lor \ell_{31}, \ell_{12} \lor \ell_{32} \rangle, [\ell_{21} \lor \ell_{41}, \ell_{22} \lor \ell_{42}]) \) is defined, then \( \ell_{11} \lor \ell_{31} \preceq \ell_{12} \lor \ell_{32} \) and \( \ell_{21} \lor \ell_{41} \preceq \ell_{22} \lor \ell_{42}. \) Also as

\( (\varepsilon_0 \lor \varepsilon_1) \circ_{\varepsilon_2} \varepsilon_3 = (\langle \ell_{11} \lor \ell_{31}, \ell_{12} \lor \ell_{32} \rangle \circ (\langle \ell_{22} \lor \ell_{42}, \ell_{72} \rangle \circ \ell_{82})), \)

is defined then \( \ell_{21} \lor \ell_{41} \lor \ell_{71} \preceq \ell_{22} \lor \ell_{42} \lor \ell_{72}, \ell_{11} \lor \ell_{31} \preceq \ell_{22} \lor \ell_{42} \lor \ell_{72}, \ell_{71} \lor \ell_{72} \preceq \ell_{82} \), and \( \ell_{21} \lor \ell_{41} \lor \ell_{71} \preceq \ell_{82}. \)

If we choose \( \varepsilon_0' \) as the interior of the judgment, then we do not get new information, therefore

\( [\ell_{21}, \ell_{22}] \subseteq [\ell_{51}, \ell_{52}], \) i.e. \( \ell_{51} \preceq \ell_{21} \) and \( \ell_{22} \preceq \ell_{52}. \) Using the same argument \( \ell_{61} \preceq \ell_{71} \) and \( \ell_{72} \preceq \ell_{62}. \)

Then

\[ \varepsilon_0' \circ_{\varepsilon_2} \varepsilon_3 = \Delta^\langle [\ell_{51}, \ell_{52} \lor \ell_{71} \lor \ell_{82}], [\ell_{61} \lor \ell_{71} \lor \ell_{81} \lor \ell_{82}]. \]

which is defined if \( \ell_{51} \preceq \ell_{72}, \ell_{71} \preceq \ell_{82} \) and \( \ell_{51} \preceq \ell_{82}. \) But \( \ell_{51} \preceq \ell_{21} \leq \ell_{41} \lor \ell_{71} \preceq \ell_{22} \lor \ell_{42} \lor \ell_{72} \preceq \ell_{72}, \ell_{51} \preceq \ell_{21} \leq \ell_{21} \lor \ell_{41} \lor \ell_{71} \preceq \ell_{82} \) and \( \ell_{71} \preceq \ell_{21} \lor \ell_{41} \lor \ell_{71} \preceq \ell_{82}. \)

Therefore

\( \varepsilon_0 \circ_{\varepsilon_2} \varepsilon_3 = (\langle \ell_{51}, \ell_{52} \lor \ell_{71} \lor \ell_{82}], [\ell_{51} \lor \ell_{71} \lor \ell_{81} \lor \ell_{82}]. \)

Using the same method, \( \varepsilon_1 \circ_{\varepsilon_2} \varepsilon_3 \) is defined if \( \ell_{21} \preceq \ell_{51} \leq \ell_{22} \lor (\ell_{52} \lor \ell_{72} \lor \ell_{82}), \) \( \ell_{11} \leq \ell_{71} \lor (\ell_{52} 

\) and \( \ell_{11} \leq \ell_{72}. \)

But by definition of \( \ell_{21} \preceq \ell_{22}, \) also \( \ell_{21} \preceq \ell_{52}, \ell_{21} \preceq \ell_{21} \lor \ell_{41} \lor \ell_{71} \leq (\ell_{22} \lor \ell_{42}) \lor \ell_{72} \leq \ell_{72}, \ell_{21} \leq \ell_{21} \lor \ell_{41} \lor \ell_{71} \leq \ell_{82}, \) and \( \ell_{51} \leq \ell_{71} \leq \ell_{72}, \) therefore \( \ell_{21} \lor \ell_{51} \leq \ell_{22} \lor (\ell_{52} \lor \ell_{72} \lor \ell_{82}). \)
Also \( \ell_{11} \leq \ell_{22} \leq \ell_{52}, \ell_{11} \leq \ell_{11} \lor \ell_{31} \leq (\ell_{22} \lor \ell_{42}) \lor \ell_{72} \leq \ell_{72}, \text{ and } \ell_{11} \leq \ell_{11} \lor \ell_{31} \leq \ell_{82}, \) therefore \( \ell_{11} \leq \ell_{22} \lor (\ell_{52} \lor \ell_{72} \lor \ell_{82}), \) and \( \ell_{11} \leq \ell_{82}. \)

Then as \( e_1 \circ \approx (e_0' \circ \approx e_3) \) is defined then if we choose \( e_2 = (e_0' \circ \approx e_3) \vdash g' \leq \ell_0, \) the result holds.

\[ \square \]

**Lemma 6.38 (Associativity).** Consider \( e_1, e_2 \) and \( e_3, \) such that \( e_1 \vdash g_1 \leq g_2, e_2 \vdash g_2 \leq g_3 \) and \( e_3 \vdash g_3 \leq g_4, (e_1 \circ \approx e_2) \circ \approx e_3 = e_1 \circ \approx (e_2 \circ \approx e_3) \)

**Proof.** Suppose \( e_1 = \langle [\ell_{11}, \ell_{12}, [\ell_{21}, \ell_{22}]] \rangle, e_2 = \langle [\ell_{31}, \ell_{32}, [\ell_{41}, \ell_{42}]] \rangle, \) and \( e_3 = \langle [\ell_{51}, \ell_{52}, [\ell_{61}, \ell_{62}]] \rangle. \) Then

\[
(e_1 \circ \approx e_2) \circ \approx e_3 = \Delta^\approx([\ell_{11}, \ell_{12}, [\ell_{21}, \ell_{22}]] \cap [\ell_{31}, \ell_{32}, [\ell_{41}, \ell_{42}]] \circ \approx e_3
\]

\[
\Delta^\approx([\ell_{11}, \ell_{12}, [\ell_{21}, \ell_{22}]] \cap [\ell_{31}, \ell_{32}, [\ell_{41}, \ell_{42}]] \circ \approx e_3 = \Delta^\approx([\ell_{11}, \ell_{12}, [\ell_{21}, \ell_{22}]] \cap [\ell_{31}, \ell_{32}, [\ell_{41}, \ell_{42}]]) \cap [\ell_{51}, \ell_{52}, [\ell_{61}, \ell_{62}])
\]

where \( \ell'_{21} = \ell_{12} \lor (\ell_{22} \land \ell_{32}) \lor \ell_{42} \land \ell_{52} \lor \ell_{62} \) and \( \ell'_{61} = \ell_{11} \lor (\ell_{21} \lor \ell_{31}) \lor \ell_{41} \lor \ell_{51} \lor \ell_{61}. \) But

\[
e_1 \circ \approx (e_2 \circ \approx e_3)
\]

\[
= \Delta^\approx([\ell_{31}, \ell_{32}, [\ell_{41}, \ell_{42}]] \cap [\ell_{51}, \ell_{52}, [\ell_{61}, \ell_{62}]) = \Delta^\approx([\ell_{11}, \ell_{12}, [\ell_{21}, \ell_{22}]] \cap [\ell_{31}, \ell_{32}, [\ell_{41}, \ell_{42}]] \cap [\ell_{51}, \ell_{52}, [\ell_{61}, \ell_{62}]) = \Delta^\approx([\ell_{11}, \ell_{12}, [\ell_{21}, \ell_{22}]] \cap [\ell_{31}, \ell_{32}, [\ell_{41}, \ell_{42}]] \cap [\ell_{51}, \ell_{52}, [\ell_{61}, \ell_{62}])
\]

where \( \ell'_{21} = \ell_{12} \lor (\ell_{22} \land \ell_{32}) \lor \ell_{42} \land \ell_{52} \lor \ell_{62} \) and \( \ell'_{61} = \ell_{11} \lor (\ell_{21} \lor \ell_{31}) \lor \ell_{41} \lor \ell_{51} \lor \ell_{61}, \) and the result holds.

\[ \square \]

**Lemma 6.39.** Consider \( e_1, e_2 \) and \( e_3, \) such that \( e_1 \vdash g_1 \leq g_2, e_2 \vdash g_2 \leq g_3 \) and \( e_3 \vdash g_3 \leq g_4. \) If \( e_1 \triangledown (e_2 \circ \approx e_3) \) is defined, then \( (e_1 \triangledown e_2) \circ \approx (e_3 \triangledown e_3) \) is defined.

**Proof.** By definition of join and consistent transitivity, using the property that the join operator is monotone.

\[ \square \]

**Lemma 6.40.** If \( \nabla e_1, \) such that \( e_1 \vdash g_1 \leq g_2, \) then \( \nabla e_2, \) such that \( e_2 \vdash g_1 \lor g_3 \leq g_2. \)

**Proof.** By definition of join and consistent transitivity, using the property that the join operator is monotone.

\[ \square \]

**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 \( \Gamma \vdash \langle \phi_1, \rho_1, \mu_1 \rangle \approx_{\ell_0}^k \langle \phi_2, \rho_2, \mu_2 \rangle, \) then if \( \forall j \leq k, \) if \( \mu'_1 \approx_{\ell_0}^{j} \mu'_2 \) then \( \Gamma \vdash \langle \phi_1, \rho_1, \mu_1 \rangle \approx_{\ell_0}^{j} \langle \phi_2, \rho_2, \mu_2 \rangle.\)
Proof. By definition of related computations and related stores. The key argument is that given that \( \mu_i \to \mu'_i \) then \( \mu'_i \) 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.

\( \square \)

**Lemma 6.42 (Substitution preserves typing).** If \( \phi \to t^U \in T[U] \) and \( \rho \models FV(t^U) \) then \( \phi \to \rho(t^U) \in T[U] \).

Proof. By induction on the derivation of \( \phi \to t^U \in T[U] \)

**Lemma 6.43 (Reduction preserves relations).** Consider \( \phi_i \leq_{t_o} \phi_i', \phi_i' \triangleright t_i \in T[U], \mu_i \in Store, t_i \vdash \mu_i \) and \( \mu_1 \approx_{t_o} \mu_2 \). Consider \( j < k \), posing \( t_i \vdash \mu_i \phi'_i \triangleright \mu'_i \), we have

\[ \langle \phi_1, t_i, \mu_1 \rangle \approx_{t_o} \langle \phi_2, t_2, \mu_2 \rangle : C(U) \text{ if and only if } \langle \phi_1, t_i', \mu_1' \rangle \approx_{t_o} \langle \phi_2, t_2', \mu_2' \rangle : C(U) \]

Proof. Direct by definition of

\( \langle \phi_1, t_i, \mu_1 \rangle \approx_{t_o} \langle \phi_2, t_2, \mu_2 \rangle : C(U) \) and transitivity of \( \phi'_i \triangleright j \).

**Lemma 6.44 (Ascription preserves relation).** Suppose \( \varepsilon \vdash \epsilon' \leq U \).

(1) If \( \langle \phi_1, \nu, \mu \rangle \approx_{\ell_o} \langle \phi_2, \nu, \mu \rangle 2 : U' \text{ then} \)

\[ \langle \phi_1, \varepsilon \nu_1 :: U, \mu_1 \rangle \approx_{\ell_o} \langle \phi_2, \varepsilon \nu_2 :: U, \mu_2 \rangle : C(U) \]

(2) If \( \langle \phi_1, t, \mu \rangle \approx_{\ell_o} \langle \phi_2, t, \mu \rangle 2 : C(U') \text{ then} \)

\[ \langle \phi_1, \varepsilon \, t_1 :: U, \mu_1 \rangle \approx_{\ell_o} \langle \phi_2, \varepsilon \, t_2 :: U, \mu_2 \rangle : C(U) \]

Proof. Following Zdancewic [2002], the proof proceeds by induction on the judgment \( \varepsilon \vdash \epsilon' \leq U \). 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 \( \langle \varepsilon \nu_1 :: U \rangle \) may not be a value: each value \( \nu_1 \) is either a bare value \( \nu_1 \) or a casted value \( \varepsilon \nu_1 :: U \), with \( \varepsilon_1 \vdash U_1 \leq U \). In the latter case, \( \langle \varepsilon \varepsilon \nu_1 :: U \rangle : U \) either steps to error (in which case the relation is vacuously established), or steps to \( \varepsilon \nu_1 :: U \), 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.

\( \square \)

**Lemma 6.45.** If \( \langle \phi_1, \nu_1, \mu_1 \rangle \approx_{\ell_o} \langle \phi_2, \nu_2, \mu_2 \rangle : U \) and, \( \phi_1 \triangleright \text{uval}(\nu_1) \in T[U_i] \) where \( U_i \leq U \), then \( \forall U', U \leq U', \nu_i \vdash U_i \leq U', \langle \phi_1, \varepsilon \nu_i :: U', \mu_1 \rangle \approx_{\ell_o} \langle \phi_2, \varepsilon \nu_i :: U', \mu_2 \rangle : U', \mu_1 \rangle : U' \).

Proof. Consider \( U' \) and \( \varepsilon \), such that \( \varepsilon \vdash U' \leq U' \). By Lemma 6.44.1, \( \langle \phi_1, \varepsilon \nu_1 :: U', \mu_1 \rangle \approx_{\ell_o} \langle \phi_2, \varepsilon \nu_2 :: U', \mu_2 \rangle : C(U) \). Next we consider the case the evidence combination do not fails. In case of a failure the lemma vacuously holds. Then as \( \phi'_i \triangleright \varepsilon \nu_i :: U' \in T[U'], \varepsilon \nu_i :: U' \mid \mu_i \phi'_i \triangleright \varepsilon \nu_i :: U' \mid \mu_i \) and the result follows using Lemma 6.43 and observational monotonicity of the transitivity (Lemma 6.51).

\( \square \)

**Lemma 6.46 (Downward Closed / Monotonicity).** If \( (1) \langle \phi_1, \nu_1, \mu_1 \rangle \approx_{\ell_o} \langle \phi_2, \nu_2, \mu_2 \rangle : U \text{ then} \)

\[ \forall j \leq k, \langle \phi_1, \nu_1, \mu_1 \rangle \approx_{\ell_o} \langle \phi_2, \nu_2, \mu_2 \rangle : U \]
Let $g_1, g_2, g_3, g_4$ be elements of $O$. Then:

1. $g_1 \bowtie g_2$ implies $g_1 \bowtie g_3$ and $g_2 \bowtie g_4$, i.e. $g_1 \bowtie 0$ or $g_2 \bowtie 0$.
2. $g_1 \bowtie g_2$ implies $g_1 \bowtie g_3$ and $g_2 \bowtie g_4$, i.e. $g_1 \bowtie 0$ or $g_2 \bowtie 0$.
3. $g_1 \bowtie g_2$ implies $g_1 \bowtie g_3$ and $g_2 \bowtie g_4$, i.e. $g_1 \bowtie 0$ or $g_2 \bowtie 0$.
4. $g_1 \bowtie g_2$ implies $g_1 \bowtie g_3$ and $g_2 \bowtie g_4$, i.e. $g_1 \bowtie 0$ or $g_2 \bowtie 0$.

By construction we know that $\ell_1 \bowtie \ell_2$. By $\ell_1 \bowtie \ell_2$ we know that $\ell_1 \bowtie \ell_2$.

If $g_1 = \ell$, then $[\ell_1, \ell_2] = [\ell, \ell]$, therefore $\ell \bowtie \ell$. If $\ell \bowtie \ell$, then $\ell_1 \bowtie \ell$, and the result holds immediately. If $\ell \bowtie \ell$, by construction of evidence we know that it must be the case that $\ell_1 \bowtie \ell_2$, then either

1. $\ell \bowtie \ell$ (which is impossible),
2. $\ell_1 \bowtie \ell \bowtie \ell$ (which is a contradiction by construction of evidence), or
3. $\ell \bowtie \ell \bowtie \ell$ (which contradicts $\ell \bowtie \ell$) or
4. $\ell_1 \bowtie \ell_2$.

so the only possibility is that $\ell_1 \bowtie \ell$, but we know that $\ell_1 \bowtie \ell_2$, i.e. $\ell_1 \bowtie \ell$ and that $\ell \bowtie \ell$, then by transitivity $\ell \bowtie \ell_2$, which is a contradiction so $\ell \bowtie \ell$ case cannot happen.

If $g_1 = g_2$, then $[\ell_1, \ell_2] = [\ell, \ell_2]$. If $\ell \bowtie \ell$, then $\ell_1 \bowtie \ell_2$, and the result holds because it must be the case that $\ell_1 \bowtie \ell_2$, then $\ell_1 \bowtie \ell_2$, and therefore (3) does not hold for $\ell_2$.

If $\ell \bowtie \ell$, then $[\ell_1, \ell_2] = [\ell_1, \ell_2]$. If $\ell \bowtie \ell$, then $\ell_1 \bowtie \ell_2$, and the result holds because (3) does not hold for $\ell_2$.

If $\ell \bowtie \ell$, then $[\ell_1, \ell_2] = [\ell_1, \ell_2]$.

Also consider $\epsilon' = \delta_{\omega}(g_1, \ell_2) = \langle [\ell_1, \ell_2], [\ell_0, \ell_0] \rangle$ and $\epsilon'' = \delta_{\omega}(g_2, \ell_0) = \langle [\ell_1', \ell_2'], [\ell_0, \ell_0] \rangle$. Then $\epsilon' \bowtie \epsilon''$ and $\epsilon' \bowtie \epsilon''$ such that $\epsilon' \bowtie \epsilon''$ and $\epsilon' \bowtie \epsilon''$.

Then $\epsilon' \bowtie \epsilon'' = \epsilon' = \langle [\ell_1, \ell_2], [\ell_0, \ell_0] \rangle$. Also consider $\epsilon' \bowtie \epsilon'' = \delta_{\omega}(g_1, \ell_0) = \langle [\ell_1, \ell_2], [\ell_0, \ell_0] \rangle$ and $\epsilon'' = \delta_{\omega}(g_2, \ell_0) = \langle [\ell_1', \ell_2'], [\ell_0, \ell_0] \rangle$. Then $\epsilon' \bowtie \epsilon'' = \epsilon' = \delta_{\omega}(g_1, \ell_0) = \langle [\ell_1, \ell_2], [\ell_0, \ell_0] \rangle$ and $\epsilon'' = \delta_{\omega}(g_2, \ell_0) = \langle [\ell_1', \ell_2'], [\ell_0, \ell_0] \rangle$.

If $g_1 = \ell_1$ and $g_2 = \ell_2$, then $\ell_1 \bowtie \ell_2 \bowtie \ell_2$ and $\ell_1 \bowtie \ell_2$. Also $\ell_1 \bowtie \ell_1$, $\ell_2 \bowtie \ell_2$, and $\ell_1 \bowtie \ell_1$. Therefore $\ell_1 \bowtie \ell_2 \bowtie \ell_2$ and $\ell_1 \bowtie \ell_1$. We know that

\[ \epsilon' \bowtie \epsilon'' = \delta_{\omega}(g_1, \ell_0) = \langle [\ell_1, \ell_2], [\ell_0, \ell_0] \rangle \] and $\epsilon'' = \delta_{\omega}(g_2, \ell_0) = \langle [\ell_1', \ell_2'], [\ell_0, \ell_0] \rangle$.

We know that

\[ \epsilon' \bowtie \epsilon'' = \delta_{\omega}(g_1, \ell_0) = \langle [\ell_1, \ell_2], [\ell_0, \ell_0] \rangle \] and $\epsilon'' = \delta_{\omega}(g_2, \ell_0) = \langle [\ell_1', \ell_2'], [\ell_0, \ell_0] \rangle$.
Lemma 6.49. Consider $\varepsilon_1 + g_1 \preceq g_2$, $\varepsilon_2 + g_2 \preceq g_3$, and $\varepsilon_3 = \varepsilon_1 \circ_\preceq \varepsilon_2$ such that $\varepsilon_3 + g_3 \preceq g_3$. Then $\operatorname{obs}_{\varepsilon_3}(\varepsilon_3(g_3)) \Rightarrow \operatorname{obs}_{\varepsilon_3}(\varepsilon_3(g_3))$.

Proof. Suppose $\varepsilon_1 = \langle \llbracket \ell_1, \ell_2, [\ell_3, \ell_4] \rrbracket, \varepsilon_2 = \langle \llbracket \ell_5, \ell_6, [\ell_7, \ell_8] \rrbracket, \varepsilon_3 = \bigwedge \varepsilon_2 \langle \llbracket \ell_1, \ell_2, [\ell_3, \ell_4, \ell_5, \ell_6, \ell_7, \ell_8] \rrbracket, \ell_1 \vee \ell_2 \wedge \ell_3 \wedge \ell_7 \leq \ell_3 \leq \ell_4 \vee \ell_5 \wedge \ell_7 \wedge \ell_8 \rangle \rangle$

Notice that as $\ell_3 \leq \ell_1 \vee \ell_3 \leq \ell_5 \leq \ell_7$ then $\varepsilon_1 \leq \varepsilon_3$, and as $\ell_7 \leq \ell_1 \vee \ell_3 \wedge \ell_7 \leq \ell_3 \leq \ell_3 \leq \ell_7$ then $\varepsilon_2 \leq \varepsilon_3$. What have we to prove is equivalent to prove that

$$(\neg \operatorname{obs}_{\varepsilon_3}(\varepsilon_3(g_2)) \vee \neg \operatorname{obs}_{\varepsilon_3}(\varepsilon_3(g_3))) \Rightarrow \neg \operatorname{obs}_{\varepsilon_3}(\varepsilon_3(g_3))$$

If $\neg \operatorname{obs}_{\varepsilon_3}(\varepsilon_3(g_2))$ and as $\varepsilon_1 \leq \varepsilon_3$, then by Lemma 6.47 $\neg \operatorname{obs}_{\varepsilon_3}(\varepsilon_3(g_3))$ and the result holds. Similarly, if $\neg \operatorname{obs}_{\varepsilon_3}(\varepsilon_3(g_2))$ and as $\varepsilon_2 \leq \varepsilon_3$, then by Lemma 6.47 $\neg \operatorname{obs}_{\varepsilon_3}(\varepsilon_3(g_3))$ and the result holds.

Lemma 6.50. Consider $\varepsilon_1 + g_1 \preceq g_2$, $\varepsilon_2 + g_2 \preceq g_3$, and $\varepsilon_3 = \varepsilon_1 \circ_\preceq \varepsilon_2$ such that $\varepsilon_3 + g_3 \preceq g_3$. Then $\operatorname{obs}_{\varepsilon_3}(\varepsilon_3(g_3)) \Rightarrow \operatorname{obs}_{\varepsilon_3}(\varepsilon_3(g_3))$.

Proof. Suppose $\varepsilon_1 = \langle \llbracket \ell_1, \ell_2, [\ell_3, \ell_4] \rrbracket, \varepsilon_2 = \langle \llbracket \ell_5, \ell_6, [\ell_7, \ell_8] \rrbracket, \varepsilon_3 = \bigwedge \varepsilon_2 \langle \llbracket \ell_1, \ell_2, [\ell_3, \ell_4, \ell_5, \ell_6, \ell_7, \ell_8] \rrbracket, \ell_1 \leq \ell_2 \wedge \ell_3 \wedge \ell_7 \leq \ell_3 \leq \ell_4 \vee \ell_5 \wedge \ell_7 \wedge \ell_8 \rangle \rangle$

By definition of the transitivity operator, $\ell_1 \leq \ell_8$, $\ell_1 \leq \ell_4 \wedge \ell_6$, and $\ell_5 \vee \ell_5 \leq \ell_8$. Let us consider $\varepsilon_1 = \llbracket g_2, \varepsilon_3 \rrbracket = \langle \llbracket \ell_1, \ell_2, [\varepsilon_3, \varepsilon_3] \rrbracket, \varepsilon_2 = \llbracket g_3, \varepsilon_3 \rrbracket = \langle \llbracket \ell_5, \ell_6, [\varepsilon_3, \varepsilon_3] \rrbracket \rangle$ We know that

1. $\ell_3 \vee \ell_4 \leq \ell_4 \wedge \ell_2$,
2. $\ell_1 \leq \ell_4 \wedge \ell_2$,
3. $\ell_3 \vee \ell_1 \leq \ell_3$ or
4. $\ell_1 \leq \ell_3$,
5. $\ell_7 \vee \ell_2 \leq \ell_8 \wedge \ell_4$,
(6) $\ell_5 \leqslant \ell_8 \land \ell'_o$, or  
(7) $\ell_7 \lor \ell'_5 \leqslant \ell_o$ or  
(8) $\ell_5 \leqslant \ell_o$.

We have to prove  
(10) $(\ell_1 \lor \ell_3 \lor \ell_5 \lor \ell_7) \lor \ell'_5 \leqslant \ell_8 \land \ell'_o$,  
(11) $\ell_1 \leqslant \ell_8 \land \ell'_o$, or  
(12) $(\ell_1 \lor \ell_3 \lor \ell_5 \lor \ell_7) \lor \ell'_5 \leqslant \ell_o$ or  
(13) $\ell_1 \leqslant \ell_o$.

Notice that if $g_3 = ?$ then $\ell'_6 = \ell_o$ and therefore by (4) $\ell_1 \leqslant \ell'_6$, and by (3), $\ell_3 \leqslant \ell'_6$. Also $\ell'_5 = \bot$ and therefore $\ell'_5 \leqslant \ell_7 \leqslant \ell_8$. If $g_3 = \ell$, then $\ell'_5 = \ell'_6 = \ell$ and $\ell_7 = \ell_8 = \ell$, but we know that $\ell_1 \leqslant \ell_8$, and therefore $\ell_1 \leqslant \ell'_6$ and $\ell'_5 \leqslant \ell_8$. Also as $\ell_3 \leqslant \ell_8$ then $\ell_3 \leqslant \ell'_6$.

We also know that $\ell_3 \lor \ell_5 \leqslant \ell_8$ and by definition of intervals $\ell_7 \leqslant \ell_8$. We know that $\ell_1 \leqslant \ell'_6$. By (5) $\ell_7 \lor \ell'_5 \leqslant \ell'_6$. By (6) $\ell_5 \leqslant \ell'_6$. Also $\ell_3 \leqslant \ell'_6$ and (10) follows.

We know that $\ell_1 \leqslant \ell_8$ and that $\ell_1 \leqslant \ell'_6$ therefore (11) holds. By (4), (3), (7), (8) and because $\ell'_5 \leqslant \ell_o$ by definition of interior, (12) holds. Finally (13) holds by (4).

\[ \text{Lemma 6.51.} \quad \text{Consider} \quad \epsilon_1 + g_1 \preceq g_2, \quad \epsilon_2 + g_2 \preceq g_3, \quad \text{and} \quad \epsilon_3 = \epsilon_1 \circ \epsilon_2 \quad \text{such that} \quad \epsilon_3 + g_1 \preceq g_3. \quad \text{Then} \quad (\neg \text{obs}_{\ell_o}(\epsilon_1 g_2) \lor \neg \text{obs}_{\ell_o}(\epsilon_2 g_3)) \iff \neg \text{obs}_{\ell_o}(\epsilon_3 g_3). \]

**Proof.** Direct by Lemmas 6.49 and 6.50. \[ \square \]

\[ \text{Lemma 6.52.} \quad \text{Consider} \quad \epsilon_1 \text{ and } \epsilon'_1 = \epsilon_2 \lor (\epsilon_1 \circ \epsilon_3), \quad \text{for some} \quad \epsilon_2 \text{ and } \epsilon_3. \quad \text{Then} \quad \epsilon_1 \mid \leqslant \epsilon'_1. \]

**Proof.** Suppose $\epsilon_2 = \langle [\ell_1, \ell_2], [\ell_3, \ell_4] \rangle$, $\epsilon_1 = \langle [\ell_5, \ell_7], [\ell_8] \rangle$, and $\epsilon_3 = \langle [\ell_9, \ell_{10}], [\ell_11, \ell_{12}] \rangle$.  
$\epsilon_1 \circ \epsilon_3 = \Delta (\langle [\ell_3, \ell_5], [\ell_7 \lor \ell_6, \ell_8 \land \ell_10], [\ell_11, \ell_{12}] \rangle, \langle [\ell_5, \ell_6 \land \ell_8 \land \ell_10 \land \ell_12], [\ell_8 \land \ell_7 \lor \ell_6 \lor \ell_9 \lor \ell_11, \ell_4 \land \ell_{12}] \rangle)$. \[ \epsilon_2 \lor (\epsilon_1 \circ \epsilon_3) = \langle [\ell_1 \lor \ell_5, \ell_2 \lor (\ell_6 \land \ell_8 \land \ell_10 \land \ell_12)], [\ell_3 \land \ell_5 \lor \ell_7 \lor \ell_9 \lor \ell_11, \ell_4 \land \ell_{12}] \rangle. \]

But $\ell_7 \leqslant \ell_5 \land \ell_7 \lor \ell_9 \lor \ell_11$ and therefore, $\epsilon_1 \mid \leqslant \epsilon'_1$. \[ \square \]

\[ \text{Lemma 6.53.} \quad \text{Consider} \quad \epsilon_1 + g'_1 \preceq g_1, \quad \epsilon'_1 = \epsilon_2 \lor (\epsilon_1 \circ \epsilon_3) \quad \text{such that} \quad \epsilon'_1 + g'_2 \preceq g_2. \quad \text{Then} \quad \neg \text{obs}_{\ell_o}(\epsilon_1 g_1) \Rightarrow \neg \text{obs}_{\ell_o}(\epsilon'_1 g_2). \]

**Proof.** By Lemma 6.52 and Lemma 6.47 the result holds immediately. \[ \square \]

\[ \text{Lemma 6.54.} \quad \text{Consider} \quad \epsilon_1 + g'_1 \preceq g_1, \quad \epsilon_2 + g'_2 \preceq g_2, \quad \text{and} \quad \epsilon_3 = \epsilon_1 \lor \epsilon_2 \quad \text{such that} \quad \epsilon_3 + g'_1 \lor g'_2 \preceq g_1 \lor g_3. \quad \text{Then} \quad \epsilon_1 \mid \leqslant \epsilon_3. \]

**Proof.** Suppose $\epsilon_1 = \langle [\ell_1, \ell_2], [\ell_3, \ell_4] \rangle$, $\epsilon_2 = \langle [\ell_5, \ell_6], [\ell_7, \ell_8] \rangle$, then $\epsilon_3 = \langle [\ell_1 \lor \ell_5, \ell_2 \lor \ell_6], [\ell_3 \lor \ell_7, \ell_4 \lor \ell_8] \rangle$. As $\ell_3 \leqslant \ell_5 \land \ell_3 \lor \ell_7$ therefore, $\epsilon_1 \mid \leqslant \epsilon_3$ and the result holds. \[ \square \]

\[ \text{Lemma 6.55.} \quad \text{Consider} \quad \epsilon_1 + g'_1 \preceq g_1, \quad \epsilon_2 + g'_2 \preceq g_2, \quad \text{and} \quad \epsilon_3 = \epsilon_1 \lor \epsilon_2 \quad \text{such that} \quad \epsilon_3 + g'_1 \lor g'_2 \preceq g_1 \lor g_2. \quad \text{Then} \quad (\neg \text{obs}_{\ell_o}(\epsilon_1 g_1) \lor \neg \text{obs}_{\ell_o}(\epsilon_2 g_2)) \iff \neg \text{obs}_{\ell_o}(\epsilon_3 (g_1 \lor g_2)). \]

**Proof.** First we prove the $\Rightarrow$ direction. By Lemma 6.54, $\epsilon_1 \mid \leqslant \epsilon_3$. Suppose \text{obs}_{\ell_o}(\epsilon_1 g_1) does not hold (the other case is analogous). Then by Lemma 6.47 the result holds immediately. Then for the $\Leftarrow$ we use Lemma 6.48 and the result holds immediately. \[ \square \]
Lemma 6.56. Consider $\phi' \triangleright t^U \in T[U]$, and $\mu$, such that $t^U \triangleright \mu$ and $\neg \text{obs}_{\ell_o}(\phi')$, and $\forall k > 0$, such that $t^U | \mu \triangleright k \tau^U | \mu'$, then $\forall \phi$,

1. $\forall \phi U' \in \text{dom}(\mu') \setminus \text{dom}(\mu)$, $\neg \text{obs}_{\ell_o}(\phi \triangleright \mu'(\phi U'))$,
2. $\forall \phi U' \in \text{dom}(\mu') \cap \text{dom}(\mu) \land \mu'(\phi U') \neq \mu(\phi U')$,
   
   (a) $\neg \text{obs}_{\ell_o}(\phi \triangleright \mu(\phi U'))$, and
   
   (b) $\neg \text{obs}_{\ell_o}(\phi \triangleright \mu'(\phi U'))$.

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

Case ($t = \epsilon_1 o_{\phi'}^U : \epsilon_2 u$). We are only updating the heap so we only have to prove (a) and (b). Then

$$\epsilon_1 o_{\phi'}^U : \epsilon_2 u \xrightarrow{\phi'} \epsilon_1 \epsilon_2 u \xrightarrow{\mu} \epsilon' \xrightarrow{(u \varphi_o \phi_o \tilde{g} \varphi_o \rho_o g_o) \triangleright U'}$$

where $\epsilon' = (\epsilon_2 \circ^\epsilon \text{iref}(\epsilon_1)) \sim((\phi' \sim \text{ilbl}(\epsilon_1)) \circ^\epsilon \epsilon_3 \circ^\epsilon \text{ilbl}(\text{iref}(\epsilon_1)))$ and if $\mu(\phi U') = \epsilon u \triangleright U'$, then $\phi' : [\leq] \epsilon$. For simplicity let us call $\epsilon_2' = (\epsilon_2 \circ^\epsilon \text{iref}(\epsilon_1))$ and $\epsilon_3' = \epsilon_3 \circ^\epsilon \text{ilbl}(\text{iref}(\epsilon_1))$. We have to prove that (b) $\neg \text{obs}_{\ell_o}(\phi' : \epsilon' \text{label}(U'))$. As $\neg \text{obs}_{\ell_o}(\phi')$, by Lemma 6.55, $\neg \text{obs}_{\ell_o}(\epsilon' \text{label}(U'))$. Next we have to prove that (a) $\text{obs}_{\ell_o}(\phi \triangleright \mu(\phi U'))$ is not defined. Consider that $\mu(\phi U') = \epsilon u \triangleright U'$. We know that $\text{obs}_{\ell_o}(\phi' : \epsilon' \text{ilbl}(U'))$ is not defined, and that $\phi' : [\leq] \epsilon$, therefore by Lemma 6.47, $\text{obs}_{\ell_o}(\epsilon U')$ is not defined, concluding that $\text{obs}_{\ell_o}(\phi \triangleright \mu(\phi U'))$ is not defined as well and the result holds.

Case ($t = \text{ref}_{\epsilon}^U \epsilon_3 u$). We are extending the heap, so we need to only prove (1). Then

$$\text{ref}_{\epsilon}^U \epsilon_3 u \xrightarrow{\phi'} \epsilon_3 u \xrightarrow{\mu} \epsilon' \xrightarrow{(u \varphi_o \phi_o \tilde{g} \varphi_o \rho_o g_o) \triangleright U'}$$

where $\phi U' \notin \text{dom}(\mu)$, $\epsilon' = \epsilon_3 \sim((\phi' \varphi_o \rho_o g_o) \circ^\epsilon \epsilon_3)$. We need to prove that $\text{obs}_{\ell_o}(\phi \triangleright \epsilon'(u \varphi_o \phi_o \rho_o g_o) \triangleright U')$ does not hold. In order to do so, we will show that $\text{obs}_{\ell_o}(\epsilon \text{ilbl}(\epsilon' \text{label}(U'))) \text{ does not holds, which follows directly by Lemma 6.53.}$

$\square$

Lemma 6.57. Consider $\phi'$, such that $\text{obs}_{\ell_o}(\phi' : \phi'_o g_o)$ does not hold, then then $\forall k > 0$, such that

$$t^U | \mu_1 \xrightarrow{\phi'} k t^U | \mu'_1$$

then if $\mu_1 \approx_{\ell_o}^k \mu_2$, then $\mu'_1 \approx_{\ell_o}^k \mu'_2$.

Proof. By Lemma 6.56 we know three things:

1. $\forall \phi U' \in \text{dom}(\mu'_1) \setminus \text{dom}(\mu_1)$, $\text{obs}_{\ell_o}(\phi \triangleright \mu'_1(\phi U'))$ does not hold, i.e. new locations are not observable.
2. $\forall \phi U' \in \text{dom}(\mu'_1) \cap \text{dom}(\mu_1) \land \mu'_1(\phi U') \neq \mu(\phi U')$,
   
   (a) $\text{obs}_{\ell_o}(\phi \triangleright \mu_1(\phi U'))$ does not hold, and
   
   (b) $\text{obs}_{\ell_o}(\phi \triangleright \mu'_1(\phi U'))$ 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 $\mu'_1 \approx_{\ell_o}^k \mu'_2$ and the result holds. $\square$
Lemma 6.58. Consider simple values \( u_i \in \mathcal{T}[U_i] \) and 
\[
\langle \phi_1, \mu_1 \rangle \approx_k^\rho \langle \phi_2, \mu_2 \rangle : U.
\]
If \( \varepsilon_1 \vdash g \leq U \), then 
\[
\langle \phi_1, \mu_1 \rangle \approx_k^\rho \langle \phi_2, \mu_2 \rangle : U.
\]

**Proof.** By definition of related values and Lemma 6.55 (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 imprecision. □

Lemma 6.59. Suppose that \( \phi_i \leq \varepsilon_\mu \phi'_i, \phi'_i \triangleright \text{prot}^{U}_{\varepsilon \phi_1} \phi''_i(e_i \cdot t_{U_i}) \in \mathcal{T}[U \setminus \gamma g_i], \) for \( i \in \{1, 2\} \), where \( \text{obs}_{\varepsilon_\mu}(\phi'_i, \phi''_i, g_i) \) does not hold, and either \( \text{obs}_{\varepsilon_\mu}(\phi_i, e_\phi_1, g_i) \) or \( \text{obs}_{\varepsilon_\mu}(\varepsilon_i, g_i) \) does not hold. Also consider two stores \( \mu_i \) such that \( \mu_1 \approx_k^\rho \mu_2 \).

Then \( \langle \phi_1, \text{prot}^{U}_{\varepsilon \phi_1} \phi'_1(e_i \cdot t_{U_i}) \rangle, \mu_1 \rangle \approx_k^\rho \langle \phi_2, \text{prot}^{U}_{\varepsilon \phi_1} \phi''_2(e_i \cdot t_{U_i}), \mu_2 \rangle \)

**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):

\[
t_{U_i} \mid \mu_i \xrightarrow{\phi'_i} j \quad \varepsilon_i v_i \mid \mu'_i
\]

Therefore:

\[
\text{prot}^{U}_{\varepsilon \phi_1} \phi''_i(e_i \cdot t_{U_i}) \mid \mu'_i \xrightarrow{\phi'_i, j} \text{prot}^{U}_{\varepsilon \phi_1} \phi''_i(e_i' \cdot t_{U_i}) \mid \mu'_i \xrightarrow{\phi'_i, 1} \langle e_i' \setminus \gamma \phi'_i \rangle \langle u_i \setminus \gamma g'_i \rangle \colon U \setminus \gamma g_i : U \setminus \gamma g_i \rangle
\]

As the values can be radically different we have to make sure that both values are not observables. If \( \text{obs}_{\varepsilon_\mu}(\phi_i, e_\phi_1, g_i) \) does not hold then the values are not observables because the security context is not observable. Let us assume that \( \text{obs}_{\varepsilon_\mu}(\phi_i, e_\phi_1, g_i) \), but does not \( \text{obs}_{\varepsilon_\mu}(\varepsilon_i, g_i) \). Then by Lemma 6.55, \( \text{obs}_{\varepsilon_\mu} \langle e_i' \setminus \gamma \phi'_i \rangle \langle \text{label}(U \setminus \gamma g) \rangle \) does not hold, and therefore \( \text{obs}_{\varepsilon_\mu} (\phi_i \triangleright (e_i' \setminus \gamma \phi'_i)(u_i \setminus \gamma g'_i)) \colon U \setminus \gamma g_i \) does not hold.

Now we have to prove that the resulting stores are related. But by Lemma 6.57 the result immediately. □

Lemma 6.60. Suppose that \( \phi_i \leq \varepsilon_\mu \phi'_i, \phi'_i \leq \varepsilon_\mu \phi''_i, \langle \phi_1, t_1, \mu_1 \rangle \approx_k^\rho \langle \phi_2, t_2, \mu_2 \rangle : C(U') \), and that 
\[
\phi'_i \triangleright \text{prot}^{U}_{\varepsilon \phi_1} \phi''_i(e_i \cdot t_{U_i}) \in \mathcal{T}[U \setminus \gamma g_i], \text{ for } i \in \{1, 2\}. \text{If } \phi_1 \approx_k^\rho \phi_2, \text{ and both } \text{obs}_{\varepsilon_\mu}(\varepsilon_i, g_i) \text{ hold or does not hold, then } \langle \phi_1, \text{prot}^{U}_{\varepsilon \phi_1} \phi''_i(e_i \cdot t_{U_i}), \mu_1 \rangle \approx_k^\rho \langle \phi_2, \text{prot}^{U}_{\varepsilon \phi_1} \phi''_2(v_i \cdot t_{U_i}), \mu_2 \rangle : C(U \setminus \gamma g_i)
\]

**Proof.** In case that combining evidence may fail, then the Lemma vacuously holds. Let us assume that combining evidence always successes. Consider \( j < k \), we know by definition of related computations that

\[
t_{U_i} \mid \mu_i \xrightarrow{\phi'_i, j} t'_{U_i} \mid \mu'_i
\]

then \( \mu'_i \approx_k^\rho \mu_2 \), and by Lemma 6.61 \( \mu_i \rightarrow \mu'_i \). If \( t_{U_i} \) are reducible after \( k - 1 \) steps, then the result holds immediately by (Rprot ()). The interesting case if \( t'_{U_i} \) are irreducible after \( j < k \) steps:

Suppose that after \( j \) steps \( t'_{U_i} = v_i \), then \( \langle \phi_1, v_i, \mu'_i \rangle \approx_{k-j}^\rho \langle \phi_2, v_i, \mu_2 \rangle : U' \).
Therefore:

\[
\text{prof}^U_{\epsilon_i} \phi''(\epsilon_i t''') \mid \mu''
\]

\[
\xrightarrow{\phi'} j \quad \text{prof}^U_{\epsilon_i} \phi''(\epsilon_i u_1) \mid \mu''
\]

\[
\xrightarrow{\phi'} 1 \quad (\epsilon_i' \sim \epsilon_i)(u_i \sim g') :: U \sim g \mid \mu''
\]

If both \(\text{obs}_{\epsilon_o}(\phi \triangleright v_i)\) do not hold, then by Lemma 6.63, \(\text{obs}_{\epsilon_o}((\epsilon_i' u_1 :: U')\) also does not hold. Finally by Lemma 6.55 \(\text{obs}_{\epsilon_o}((\epsilon_i' \sim \epsilon_i)(\text{label}(U) \sim g))\) does not hold and therefore the values are related.

Let us consider that both \(\text{obs}_{\epsilon_o}(\phi \triangleright v_i)\) holds and that \(\text{obs}_{\epsilon_o}(\phi \triangleright \epsilon_i' u_1 :: U')\) holds (otherwise we follow by the previous argument). If both \(\text{obs}_{\epsilon_o}(\phi \triangleright \epsilon_i' u_1 :: U')\) do not hold, then the values are not observables because the security contexts are not observable.

Let us assume that both \(\text{obs}_{\epsilon_o}(\phi_1 \triangleright \epsilon_i' g_c)\) hold, but \(\text{obs}_{\epsilon_o}(\epsilon_i' g)\) not. Then by Lemma 6.55, \(\text{obs}_{\epsilon_o}((\epsilon_i' \sim \epsilon_i)(\text{label}(U) \sim g))\) do not hold, and therefore \(\text{obs}_{\epsilon_o}(\phi_1 \triangleright (\epsilon_i' \sim g)):: U \sim g\) do not hold.

If \(\text{obs}_{\epsilon_o}(\phi_1 \triangleright \epsilon_i' g_c)\) and \(\text{obs}_{\epsilon_o}((\epsilon_i' \sim \epsilon_i)(\text{label}(U) \sim g))\) hold, then the result follows by Lemma 6.58, and by backward preservation of the relations (Lemma 6.43).

\(\Box\)

**Lemma 6.61.** Consider term \(\phi \triangleright t^U \in \mathbb{T}[U]\), store \(\mu\) and \(j > 0\), such that \(t^U \mid \mu \xrightarrow{\phi} j t^U \mid \mu'\). Then \(\mu \xrightarrow{\phi'} \mu'\).

**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. \(\Box\)

**Lemma 6.62.** If \(\phi \leq_{\epsilon_o} \phi'\) and \(\phi' \leq_{\epsilon_o} \phi''\), then \(\phi \leq_{\epsilon_o} \phi''\).

**Proof.** Trivial because if \(\phi\) is not observable, then \(\phi'\) is not observable as well by definition of \(\leq_{\epsilon_o}\), and therefore \(\phi''\) must also be not observable. \(\Box\)

**Lemma 6.63.** Consider \(\phi_1 \triangleright v \in \mathbb{T}[U]\), and \(\epsilon \triangleright U \subseteq U'. Suppose \(\epsilon v :: U' \xrightarrow{L} \epsilon' v :: U'\). If \(\neg \text{obs}_{\epsilon_o}(\phi_1 \triangleright v) \lor \neg \text{obs}_{\epsilon_o}(\epsilon v :: U') \iff \neg \text{obs}_{\epsilon_o}(\phi_1 \triangleright \epsilon' v :: U')\).

**Proof.** Direct by Lemma 6.51. \(\Box\)

Next, we present the Noninterference proposition, which naturally implies the Security Type Soundness proposition (Prop 2.24) presented in the paper.

**Proposition 6.64 (Noninterference).** If \(\phi_i' \triangleright \ell \in \mathbb{T}[U],\mu_j \in \text{STORE,} \ell \vdash \mu_i, \Gamma = \text{FV(}\ell\text{)}\), and \(\forall k \geq 0, \phi_i \leq_{\epsilon_o} \phi_i', \Gamma \vdash (\phi_1, \rho_1, \mu_1) \approx_{\epsilon_o}^k (\phi_2, \rho_2, \mu_2)\), then \(\langle \phi_1, \rho_1(\ell) \rangle, \mu_1, \approx_{\epsilon_o} \{ \ell \}^k \langle \phi_2, \rho_2(\ell) \rangle, \mu_2 \rangle : C(U)\).

**Proof.** By induction on the derivation of term \(\ell \in \mathbb{T}[U]\). Let us take an arbitrary index \(k \geq 0\).

**Case (a).** \(\ell = x^U\) so \(\Gamma = \{ x^U \}. \Gamma \vdash (\phi_1, \rho_1, \mu_1) \approx_{\epsilon_o}^k (\phi_2, \rho_2, \mu_2)\) implies by definition that \(\langle \phi_1, \rho_1(x^U) \rangle, \mu_1 \approx_{\epsilon_o}^k (\phi_2, \rho_2(x^U), \mu_2) : U\), and the result holds immediately.

**Case (b).** \(\ell = b_g\). By definition of substitution, \(\rho_1(b_g) = \rho_2(b_g) = b_g\). By definition, \(\langle \phi_1, b_g, \mu_1 \rangle \approx_{\epsilon_o}^k (\phi_2, b_g, \mu_2) : \text{Bool}_r\) as required.
Case (α). \( i = o^{U_1}_{\beta_i} \) where \( U = \text{Ref}_{\beta_i} U_1 \). By definition of substitution, \( \rho_1(o^{U_1}_{\beta_i}) = \rho_2(o^{U_1}_{\beta_i}) = o^{U_1}_{\beta_i} \). We know that \( \phi_1 \triangleright o^{U_1}_{\beta_i} \in \text{T}[\text{Ref}_{\beta_i} U_1] \). By definition of related stores, \( \langle \phi_1, o^{U_1}_{\beta_i}, \mu_1 \rangle \approx^{\ell_o} \langle \phi_2, o^{U_1}_{\beta_i}, \mu_2 \rangle : \text{Ref}_{\beta_i} U_1 \) as required, and the result holds.

---

Case (λ). \( t^U = (\lambda \delta^U \cdot U_1, t^{U_2}) \). Then \( U = U_1 \triangleright g \circ U_2 \).

By definition of substitution, assuming \( x^{U_1} \not\in \text{dom}(\rho_1) \), and Lemma 6.42:

\[
\phi'_1 \triangleright \rho_1(t^{U_1}) = \phi' \triangleright (\lambda \delta^U \cdot U_1, \rho_1(t^{U_2})) \in \text{T}[U]
\]

Consider \( j \leq k, \mu'_1, \mu'_2 \) such that \( \mu_i \rightarrow \mu_i' \) and \( \mu_i' \approx^{\ell_o} \mu'_2 \), and assume two values \( v_1 \) and \( v_2 \) such that \( \langle \phi_1, v_1, \mu_1' \rangle \approx^{\ell_o} \langle \phi_2, v_2, \mu_2' \rangle : U_1' \). Consider \( U' = U_1' \triangleright g \circ U_2' \) and \( \epsilon_1, \epsilon_2, \epsilon_\ell \), such that

\[
\epsilon_1 \triangleright U_1 \triangleright g \circ U_2 \leq U', \quad \epsilon_2 \triangleright U_1' \leq U''', \quad \text{and that} \quad \epsilon_\ell \triangleright \phi'_1 \triangleright \epsilon g'' \leq g'''
\]

For simplicity, let us annotate \( U_1' = U_2'' \triangleright g'' \). We need to show that:

\[
\langle \phi_1, \epsilon_1(\lambda \delta^U \cdot U_1, \rho_1(t^{U_2})), \epsilon_2 v_1, \mu_1' \rangle \approx^{\ell_o} \langle \phi_2, \epsilon_1(\lambda \delta^U \cdot U_1, \rho_2(t^{U_2})), \epsilon_2 v_2, \mu_2' \rangle : C(U_2')
\]

Each \( v_1 \) is either a bare value \( u_i \) or a casted value \( \epsilon_2 u_i :: U_1' \). In the latter case, the application expression combines evidence, which may fail with error. If it succeeds, we call the combined evidence \( \epsilon' \). The application rule then applies: it may fail with error if the evidence \( \epsilon' \) 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:

\[
\epsilon_1(\lambda \delta^U \cdot U_1, \rho_1(t^{U_2})), \epsilon_2 v_1, \mu_1' \triangleright \epsilon_1(\lambda \delta^U \cdot U_1, \rho_1(t^{U_2})), \epsilon_2 v_1, \mu_1'
\]

where \( \epsilon'' = \langle \epsilon'_1(\phi_1, g \circ g), g' \rangle, \epsilon'_1 = (\phi'_1 \circ g \circ \text{idl}(\epsilon_1)) \circ^\kappa \epsilon_\ell \circ^\kappa \text{idl}(\epsilon_1) \). If \( \text{idl}(\epsilon_1) \) do not hold, then by Lemma 6.55, \( \text{idl}(\epsilon_1) \) do not hold. Then \( \phi'_1 \leq \ell_o \phi_1'' \), and by Lemma 6.62, \( \phi_1 \leq \ell_o \phi_1'' \).

\( \epsilon_\ell, \epsilon_p \) and \( \epsilon_{ai} \) are the new evidences for the label, return value and argument, respectively. We then extend the substitutions to map \( x^{U_1} \) to the casted arguments:

\[
\rho_1' = \rho_1\{x^{U_1} \mapsto \epsilon_{ai} u_i :: U_1\}
\]

We know that \( \langle \phi_1, v_1, \mu_1' \rangle \approx^{\ell_o} \langle \phi_2, v_2, \mu_2' \rangle \) and consider \( \phi \triangleright u_i \in \text{T}[U_{ui}] \) then \( \epsilon_{ai} \triangleright U_{ui} \leq U_1 \) and \( \epsilon_{ai} = (\epsilon_2 \circ^\kappa \epsilon_2) \circ^\kappa \text{idl}(\epsilon_1) \), therefore using Lemma 6.45 \( \langle \phi_1, (\epsilon_{ai} u_i :: U_1), \mu_1' \rangle \approx^{\ell_o} \langle \phi_2(\epsilon_{ai} u_2 :: U_1), \mu_2' \rangle : U_1 \)

So as \( \mu_i \rightarrow \mu_i' \) then by Lemma 6.41, \( x^{U_1} \triangleright \langle \phi_1, \rho_1, \mu_1' \rangle \approx^{\ell_o} \langle \phi_2, \rho_2, \mu_2' \rangle \).

We also know that \( \phi'' \triangleright \rho_1(t^{U_2}) \in \text{T}[U_2] \). Then by induction hypothesis:

\[
\langle \phi_1, \rho_1(t^{U_2}), \mu_1' \rangle \approx^{\ell_o} \langle \phi_2, \rho_2(t^{U_2}), \mu_2' \rangle : C(U_2)
\]

Finally, by Lemma 6.60:

\[
\langle \phi_1, \text{prot}_{\phi_1^\mu(t^{U_2})}^\epsilon \phi''(\epsilon \rho_1(t^{U_2})), \mu_1' \rangle \approx^{\ell_o} \langle \phi_2, \text{prot}_{\phi_1^\mu(t^{U_2})}^\epsilon \phi''(\epsilon \rho_2(t^{U_2})), \mu_2' \rangle : C(U_2)
\]

and finally the result holds by backward preservation of the relations (Lemma 6.43).
Case (1). \( t^U = !\text{Ref}\_U \cdot t^{\text{Ref}\_U} \). Then \( U = U_1 \vdash g \).

By definition of substitution:
\[
\rho_i(t^U) = !\text{Ref}\_U \cdot \rho_i(t^{\text{Ref}\_U})
\]

We have to show that
\[
\phi_i \approx_k^{\ell_o} \langle \phi_{1}, !\text{Ref}\_U \cdot \rho_i(t^{\text{Ref}\_U}), \mu_1 \rangle
\]

By Lemma 6.42:
\[
\phi'_i \triangleright !\text{Ref}\_U \cdot \rho_i(t^{\text{Ref}\_U}) \in T[U_1 \vdash g]
\]

By induction hypotheses on the subterm:
\[
\langle \phi_{1}, \rho_i(t^{\text{Ref}\_U}), \mu_1 \rangle \approx_k^{\ell_o} \langle \phi_{2}, \rho_2(t^{\text{Ref}\_U}), \mu_2 \rangle : C(U_1' \vdash g)
\]

Consider \( j < k \), then by definition of related computations
\[
\rho_i(t^{\text{Ref}\_U}) \mid \mu_i \xrightarrow{\phi'_i} j_{t^{\text{Ref}\_U}}(U_1') \mid \mu'_i \implies \mu'_i \approx j_{\ell_o}^{k-j} \mu'_2 \land \text{irred}(t^{\text{Ref}\_U}) \Rightarrow \langle \phi_{1}, t^{\text{Ref}\_U}, \mu_1 \rangle \approx_k^{\ell_o} \langle \phi_{2}, t^{\text{Ref}\_U}, \mu_2 \rangle : U_1'
\]

Where \( U_1' = \text{Ref}_g' \cdot U_1'' \). If terms \( t_1^{\text{Ref}\_U} \) are reducible after \( j = k-1 \) steps, then
\[
!\text{Ref}_U \cdot \rho_i(t^{\text{Ref}\_U}) \mid \mu_i \xrightarrow{\phi'_i} j_{\text{Ref}_U}^{\ell_o} t^{\text{Ref}_U}(U_1') \mid \mu'_i \text{ and the result holds.}
\]

If after at most \( j \) steps \( t_1^{\text{Ref}_U} \) is irreducible it means that for some \( j' \leq j \), \( !\text{Ref}_U \cdot \rho_i(t^{\text{Ref}_U}) \mid \mu_i \xrightarrow{\phi'_i} j'_{\text{Ref}_U} \cdot U_1' \mid \mu'_i \). If \( j' = j \) then we use the same same argument for reducible terms and the result holds.

Let us consider now \( j' < j \). Then \( \langle \phi_{1}, v_1, \mu_1 \rangle \approx_k^{\ell_o} \langle \phi_{2}, v_2, \mu_2 \rangle : U_1' \). By Lemma 6.10, each \( v_i \) is either a location \( (o_i)_{g_i}^{U_i''} \) or a casted location \( \epsilon_i(o_i)_{g_i}^{U_i''} \) \( : U_1' \). Let us assume they both are a casted location (the other cases are analogous). In case a value \( v_i \) is a casted value, then the whole term \( \rho_i(t^{\text{Ref}_U}) \) can take a step by (Rg), combining \( \epsilon_i \) with \( \epsilon_i \). Such a step either fails, or succeeds with a new combined evidence. Therefore, either:
\[
\rho_i(t^{\text{Ref}_U}) \mid \mu_i \xrightarrow{\phi'_i} j \text{ error}
\]
in which case we do not care since we only consider termination-insensitive noninterference, or:
\[
\rho_i(t^{\text{Ref}_U}) \mid \mu_i \xrightarrow{\phi'_i} j^{+1} \xrightarrow{\text{Ref}_U \cdot \alpha_{g_i}^{U_i'''}} \mu'_i \]

with \( v_i' = \mu'_i(o_i)_{g_i}^{U_i'''}) = \epsilon_i u_i' \vdash U_i''' \), \( \phi_i''' = \langle \phi_{1}|e \vdash \text{ilbl}(e)\rangle(\phi_{1|g \vdash \text{Ref}_g \cdot g \vdash g}) \). By Lemma 6.55, if \( \sim \text{obs}_{\alpha}(\phi_i') \text{ then } \sim \text{obs}_{\alpha}(\phi_i'' \rangle \text{ Then by Lemma 6.62, } \phi_i \leq_{\ell_o} \phi_i'. \text { As } \langle \phi_{1}, v_1, \mu_1 \rangle \approx_k^{\ell_o} \langle \phi_{2}, v_2, \mu_2 \rangle : U_1' \text{, then by Lemma 6.51 either both } \text{obs}_{\alpha}(\text{ilbl}(e)(\text{label}(U) \vdash g)) \text{ holds or do not hold. Finally as } \langle \phi_{1}, v_1, \mu_1 \rangle \approx_k^{\ell_o} \langle \phi_{2}, v_2, \mu_2 \rangle : U_1', \text{ by Lemma 6.59,}
\]

\[
\langle \phi_{1}, \text{prot}_{\text{ilbl}(e)_{g_i}^{U_i'''}}(\text{iref}_{\epsilon_i} v_i'), \mu_1 \rangle \approx_{\ell_o} \langle \phi_{2}, \text{prot}_{\text{ilbl}(e)_{g_i}^{U_i'''}}(\text{iref}_{\epsilon_i} v_i'), \mu_2 \rangle : C(U_2')
\]

and finally the result holds by backward preservation of the relations (Lemma 6.43).
Case (\(=\)). \(t^U = \varepsilon_1 U_1^g \varepsilon_2 U_2^g\). Then \(U = \text{Unit}_\bot\).

By definition of substitution:

\[
\rho_1(t^U) = \varepsilon_1 \rho_1(t^{U_1}) \varepsilon_2 \rho_1(t^{U_2})
\]

and Lemma 6.42:

\[
\phi_1' \succ \varepsilon_1 \rho_1(t^{U_1}) \varepsilon_2 \rho_1(t^{U_2}) \in T[\text{Unit}_\bot]
\]

We have to show that

\[
\langle \phi_1, \varepsilon_1 \rho_1(t^{U_1}), \mu_1 \rangle \approx_k \langle \phi_2, \varepsilon_2 \rho_2(t^{U_2}), \mu_2 \rangle : C(U)
\]

By induction hypotheses

\[
\langle \phi_1, \rho_1(t^{U_1}), \mu_1 \rangle \approx_k \langle \phi_2, \rho_2(t^{U_1}), \mu_2 \rangle : C(U_1)
\]

Suppose \(j_1 < k\), and that \(\rho_1(t^{U_1})\) are irreducible after \(j_1\) steps (otherwise, similar to case \(!\), the result holds immediately). Then by definition of related computations:

\[
\rho_1(t^{U_1}) \mid \mu_1 \xrightarrow{\phi_1'} h \cdot v_1 \mid \mu_1' \implies \mu_1' \approx_k \mu_2' \wedge \langle \phi_1, v_1, \mu_1' \rangle \approx_k \langle \phi_2, v_2, \mu_2' \rangle : U_1
\]

By Lemma 6.61 \(\mu_1 \rightarrow \mu_2'\), and \(\mu_2' \approx_k \mu_2''\) then by Lemma 6.41, \(\langle \phi_1, \rho_1(t^{U_1}), \mu_1' \rangle \approx_k \langle \phi_2, \rho_2(t^{U_1}), \mu_2'' \rangle\). By induction hypotheses:

\[
\langle \phi_1, \phi_1(t^{U_1}), \mu_1' \rangle \approx_k \langle \phi_2, \rho_2(t^{U_2}), \mu_2'' \rangle : C(U_2)
\]

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 \(v_1'\):

\[
\rho_1(t^{U_2}) \mid \mu_1' \xrightarrow{\phi_1'} h' \cdot v_1' \mid \mu_2'' \wedge \langle \phi_1, v_1', \mu_1'' \rangle \approx_k \langle \phi_2, v_2, \mu_2'' \rangle : U_2
\]

Then

\[
\rho_1(t^{U_1}) \mid \mu_1 \xrightarrow{\phi_1'} h' \cdot \varepsilon_1 u_1 \varepsilon_2 v_1' : \mu_1'' \approx_k \mu_2''
\]

Now \(v_1\) and \(v_1'\) can be bare values or casted values. In the case of casted values we can combine evidence, which may fail with \textbf{error}. We assume that all evidence combinations succeed, otherwise the relation vacuously holds. As both values \(v_1\) are related at some reference type, then by canonical forms (Lemma 6.10) they both must be locations \(U_i'\) for some \(U_i' \subseteq U_i\). We consider when the values are observable and the locations are identical (otherwise the result is trivial):

\[
\varepsilon_1' U_1^g : \varepsilon_2 U_2^g \mid \mu_1''
\]

Where \(\mu_1'' = \mu_2''[\alpha_U]^g \implies \varepsilon_1' (u_1' \mapsto \phi_1, \varepsilon_2 \mu_2'' : U_2']\). As

\[
\langle \phi_1, v_1', \mu_1' \rangle \approx_k \langle \phi_2, v_2, \mu_2'' \rangle : U_2\text{ then by Lemma 6.45,}
\]

\[
\langle \phi_1, \varepsilon_1' U_1' : U_1', \mu_1' \rangle \approx_k \langle \phi_2, \varepsilon_2' U_2' : U_2', \mu_2'' \rangle : U_2'.\text{ As } \varepsilon_1' + \phi_1, \varepsilon_2 g \iff g \equiv \text{label}(U_1')\text{ and } \varepsilon_1' = \varepsilon_2'.
\]

by Lemma 6.58,

\[
\langle \phi_1, \varepsilon_1' (u_1' \mapsto \phi_1, \varepsilon_2 g \iff g) : U_1', \mu_1' \rangle
\]

\[
\approx_k \langle \phi_2, \varepsilon_2' (u_2' \mapsto \phi_1, \varepsilon_2 g \iff g) : U_2', \mu_2'' \rangle
\]


Also if $\neg \text{obs}_\ell (\phi_1) \Rightarrow \neg \text{obs}_\ell (\phi_1')$ and therefore by monotonicity of the join $\neg \text{obs}_\ell (\epsilon_1' \text{label}(U_1'))$. Therefore if 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 $\forall, \phi_1' \approx_{\epsilon_1'} \phi_2'$.

As every values are related at type Unit, we only have to prove that $\mu'' \approx_{\epsilon_1'} \mu'''$, but using monotonicity (Lemma 6.46), it is trivial to prove that because either both both stores update the same location $\sigma^U_1$ to values that are related, therefore the result holds.

---

**Case (ref ).** $t^U = \text{ref}_{\epsilon_1'}^U \epsilon t^U_1$. Then $U = \text{Ref}_{\bot} U_1$.

By definition of substitution:

$$\rho_i(t^U) = \text{ref}_{\epsilon_1'}^U \epsilon p_i(t^U_1)$$

and Lemma 6.42:

$$\phi_1' \succ \text{ref}_{\epsilon_1'}^U \epsilon p_i(t^U_1) \in T[\text{Ref}_{\bot} U_1]$$

We have to show that

$$\langle \phi_1, \text{ref}_{\epsilon_1'}^U \epsilon p_1(t^U_1), \mu_1 \rangle \approx_{\epsilon_1} \langle \phi_2, \text{ref}_{\epsilon_1'}^U \epsilon p_2(t^U_1), \mu_2 \rangle : C(\text{Ref}_{\bot} U_1)$$

By induction hypotheses:

$$\langle \phi_1, \rho_1(t^U_1), \mu \rangle \approx_{\epsilon_1} \langle \phi_2, \rho_2(t^U_1), \mu \rangle : C(U_1')$$

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

$$\rho_1(t^U_1) \mid \mu_i \xrightarrow{\phi_1'} \text{ref}_{\epsilon_1'}^U \epsilon t^U_1 \mid \mu_1' \Rightarrow \mu_1' \approx_{\epsilon_1} \mu_2' \wedge (\text{irred}(t^U_1) \Rightarrow \langle \phi_1, t^U_1, \mu_1' \rangle \approx_{\epsilon_1} \langle \phi_2, t^U_1, \mu_2' \rangle : U_1')$$

If terms $t^U_1$ are reducible after $j = k - 1$ steps, then

$$\text{ref}_{\epsilon_1'}^U \epsilon p_1(t^U_1) \mid \mu_1 \xrightarrow{\phi_1'} \text{ref}_{\epsilon_1'}^U \epsilon t^U_1 \mid \mu_1'$$

and the result holds.

If after at most $j$ steps $t^U_1$ is irreducible, it means that for some $j' \leq j \text{ ref}_{\epsilon_1'}^U \epsilon p_1(t^U_1) \mid \mu_i \xrightarrow{\phi_i'} \text{ref}_{\epsilon_1'}^U \epsilon v_i \mid \mu_i'$. If $j' = j$ then we use the same same argument for reducible terms and the result holds.

Let us consider now $j' < j$. By Lemma 6.10, each $v_i$ is either a base value $u_i$ or a casted base value $\epsilon_i u_i :: U_i'$. In case a value $v_{ij}$ is a casted value, then the whole term $p_1(t^U)$ can take a step by (Rg), combining $\epsilon$ with $\epsilon_i$. Such a step either fails, or succeeds with a new combined evidence. Therefore, either:

$$\rho_i(t^U) \mid \mu_i \xrightarrow{\phi_i'} \text{error}$$

in which case we do not care since we only consider termination-insensitive noninterference, or:

$$\rho_1(t^U) \mid \mu \xrightarrow{\phi_1'} \text{ref}_{\epsilon_1'}^U \epsilon u_1 \mid \mu_1'$$

with, $\mu_1'' = \mu_1'[\sigma^U_1 \Rightarrow \epsilon_1(u_1 \Rightarrow \phi_1' g_1) :: U_1]$. Where $\epsilon_1'' = \epsilon_1 \wedge (\phi_1' \epsilon \approx \epsilon_1)$. We know that if $u_1 \in T[U_1]$, then $\epsilon_1 \vdash U_1 \leq U_1$. Also, as $(\phi_1, \mu_1) \approx_{\epsilon_1} (\phi_2, \mu_2) : U_1'$ then by Lemma 6.45,

$$\langle \phi_1, \epsilon_1 u_1 :: U_1, \mu_1' \rangle \approx_{\epsilon_1} \langle \phi_2, \epsilon_2 u_2 :: U_1, \mu_2' \rangle : U_1'$$

and as $(\phi_1' \epsilon \approx \epsilon_1) + \phi_1' \epsilon_2 \wedge g_1 \approx \text{label}(U_1)$, then by
Lemma 6.58, Lemma 6.53, and Lemma 6.46,
\langle ϕ_1, ε'_1 \rangle (u_1 \top φ'_1[0]) :: U_1, μ'_1 \approx_{ε'_1}^{k-j-2} \langle ϕ_2, ε'_2 \rangle (u_2 \top φ'_2[0]) :: U_1, μ'_2 : U'_1.

Also if \neg obs_{ε_o}(ϕ_i) ⇒ \neg obs_{ε_o}(ϕ'_i) and therefore by monotonicity of the join \neg obs_{ε_o}(ε'_i[\text{label}]}(U_1)). Therefore if 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, φ'_1 \approx_{ε'_1}^{k} φ''_1, \langle φ''_1, ε'_1 \rangle (u_1 \top φ'_1[0]) :: U_1, μ'_1 \approx_{ε'_1}^{k-j-2} \langle φ''_2, ε'_2 \rangle (u_2 \top φ'_2[0]) :: U_1, μ'_2 : U'_1.

By definition of related stores μ''_1 \approx_{ε'_1}^{k-j-2} μ''_2. Then by Monotonicity of the relation (Lemma 6.46)
μ''_1 \approx_{ε'_1}^{k-j-2} μ''_2 and the result holds.

———

Case (⊕). \textbf{t}^U = ε_1 t^{U_1} \oplus^g ε_2 t^{U_2}

By definition of substitution:
ρ_i(t^U) = ε_1 ρ_i(t^{U_1}) \oplus^g ε_2 ρ_i(t^{U_2})

and Lemma 6.42:
φ'_i = ε_1 ρ_i(t^{U_1}) \oplus^g ε_2 ρ_i(t^{U_2}) \in \mathbb{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:
ρ_i(t^{U_1}) | μ_i \overrightarrow{Δ} h_i v_{11} | μ'_i \implies μ'_1 \approx_{ε'_1}^{k-j_1} μ'_2 \land \langle ϕ_1, v_{11}, μ'_1 \rangle \approx_{ε'_1}^{k-j_1} \langle ϕ_2, v_{21}, μ'_2 : U_1 \rangle

ρ_i(t^{U_2}) | μ'_i \overrightarrow{Δ} h_j v_{12} | μ''_i \implies μ''_1 \approx_{ε'_1}^{k-j_1-j_2} μ''_2 \land \langle ϕ_1, v_{12}, μ'_1 \rangle \approx_{ε'_1}^{k-j_1-j_2} \langle ϕ_2, v_{22}, μ''_2 : U_2 \rangle

By Lemma 6.10, each v_{ij} is either a boolean (b_{ij})_{g_{ij}} or a casted boolean ε_{ij}(b_{ij}) g'_{ij} :: U_{ij}. In case a value v_{ij} is a casted value, then the whole term ρ_i(t^U) can take a step by (Rg), combining ϵ_i with ε_{ij}. Such a step either fails, or succeeds with a new combined evidence. Therefore, either:
ρ_i(t^U) | μ_i \overrightarrow{Δ} h_1 + h_2 \text{error}

in which case we do not care since we only consider termination-insensitive noninterference, or:
\overrightarrow{Δ} h_1 + h_2 + 2 ρ_i(t^U) | μ''_i

\overrightarrow{Δ}^1 ϵ_i(b_{11})_{g'_{11}} \oplus^g ϵ'_2(b_{21})_{g'_{21}} | μ''_i

with b_i = b_{11}[\oplus]b_{12}, ϵ_i = ε_{11} \top v_{12}, and g'_i = g'_{11} \top g'_{12}. It remains to show that:
\langle ϕ_1, ϵ_i(b_{11})_{g'_{11}} :: \text{Bool}_g, μ''_1 \rangle \approx_{ε'_1}^{k-j_1-j_2-3} \langle ϕ_2, ϵ'_2(b_{21})_{g'_{21}} :: \text{Bool}_g, μ''_2 : \text{Bool}_g \rangle

If \neg obs_{ε_o}(ϕ_i), then the result is trivial because the resulting booleans are also related as they are not observable.

If obs_{ε_o}(ϕ_i), then by Lemma 6.45, \langle ϕ_1, ϵ_i(b_{11})_{g'_{11}} :: \text{Bool}_g, μ''_1 \rangle \approx_{ε'_1}^{k} \langle ϕ_2, ϵ'_2(b_{21})_{g'_{21}} :: \text{Bool}_g, μ''_2 : \text{Bool}_g \rangle. If \neg obs_{ε_o}(ilbl(ε'_1)g) or \neg obs_{ε_o}(ilbl(ε'_2)g), then by Lemma 6.55, \neg obs_{ε_o}(ε'_i g) and the result holds. If both obs_{ε_o}(ilbl(ε'_1)g) then b_{11} = b_{21} and b_{12} = b_{22}, so b_1 = b_2, and the result holds.

———
We omit the $\epsilon$ 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:

$$\phi' \triangleright \epsilon_1 \rho_i(t^{U_1}) \epsilon_2 \rho_i(t^{U_2}) \in T[U]$$

We use a similar argument to case $\epsilon$ 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:

$$\rho_i(t^{U_1}) | \mu_i \xrightarrow{\phi_i} j_1 \nu_{i_1} | \mu_i' \implies \mu_i' \approx^{k-j_1} \mu_2' \wedge \langle \phi_1, \nu_{i_1}, \mu_i' \rangle \approx^{k-j_1} \langle \phi_2, \nu_{i_1}, \mu_2' \rangle : U_1$$

$$\rho_i(t^{U_2}) | \mu_i' \xrightarrow{\phi_i} j_2 \nu_{i_2} | \mu_i'' \implies \mu_i'' \approx^{k-j_1} k-j_2 \mu_2'' \wedge \langle \phi_1, \nu_{i_2}, \mu_i'' \rangle \approx^{k-j_1} \langle \phi_2, \nu_{i_2}, \mu_2'' \rangle : U_2$$

Then

$$\rho_i(t^U) | \mu_i \xrightarrow{\phi_i} j_1 + j_2 \epsilon_1 \nu_{i_1} \epsilon_2 \nu_{i_2} | \mu_i'''$$

If $\text{obs}_{\epsilon_1}(\phi_i \triangleright \nu_{i_1})$ then, by definition of $\approx_{\epsilon_1}$ at values of function type, using $\epsilon_1$ and $\epsilon_2$ to justify the subtyping relations, we have:

$$\langle \phi_1, (\epsilon_1 \nu_{i_1} \epsilon_2 \nu_{i_2}), \mu_i' \rangle \approx^{k-j_1} k-j_2 \langle \phi_2, (\epsilon_1 \nu_{i_2} \epsilon_2 \nu_{i_2}), \mu_i'' \rangle : C(U_{i_2} \neg \nu g)$$

Finally, by backward preservation of the relations (Lemma 6.43) the result holds.

If $\neg \text{obs}_{\epsilon_1}(\phi_i \triangleright \nu_{i_1})$, and we assume by canonical forms that $\nu_{i_1} = \epsilon_11(\lambda g \nu t)_{\nu_1} :: U_1$ and that $\nu_{i_2} = \epsilon_12 \nu_{i_2} :: U_2$ (and that evidence combination always succeed or the result holds immediately), then

$$\langle (\epsilon_1 \nu_{i_1} \epsilon_2 \nu_{i_2}), \mu_i' \rangle \xrightarrow{\phi_i'} 1 \langle (\epsilon_1' \nu_{i_1} \epsilon_2' \nu_{i_2}), \mu_i'' \rangle$$

$$\xrightarrow{\epsilon_1' \nu_{i_1} \epsilon_2' \nu_{i_2}} 1 \text{ prot}_{\text{ilbl}(\epsilon_1') \nu_1 \epsilon_2' \nu_{i_2}} \phi_i''(\text{icod}(\epsilon_1') t_1') \mu_i''$$

Where $\epsilon_1' = \epsilon_1 \circ \epsilon_1$, $\epsilon_1' = \epsilon_1 \circ \epsilon_1$, and $\phi_i'' = \langle (\phi_i' \epsilon_1 \epsilon_2' \nu_{i_2}), g_1' \rangle$, $\epsilon_i'' = (\phi_i' \epsilon_1 \epsilon_2' \nu_{i_2} \text{ ilbl}(\epsilon_1') \circ \epsilon_1 \circ \epsilon_1 \text{ ilat}(\epsilon_1'))$.

If $\neg \text{obs}_{\epsilon_1}(\phi_i)$ then $\neg \text{obs}_{\epsilon_1}(\phi_i')$ and by Lemma 6.55 and 6.53, $\neg \text{obs}_{\epsilon_1}(\phi_i'')$. As $\epsilon_1' = \epsilon_1 \circ \epsilon_1$, by Lemma 6.51, either both $\text{ilbl}(\epsilon_1')$ are observable or not (the latter when $\neg \text{obs}_{\epsilon_1}(\text{ilbl}(\epsilon_1') \text{ilbl}(U_{i_1})))$.

If $\neg \text{obs}_{\epsilon_1}(\text{ilbl}(\epsilon_1') \text{ilbl}(U_{i_1}))$ then similar to the context case, $\neg \text{obs}_{\epsilon_1}(\phi_i'')$.

Finally by Lemma 6.59,

$$\langle \phi_1, \text{ prot}_{\text{ilbl}(\epsilon_1') \nu_1 \epsilon_2' \nu_{i_2}} \phi_i''(\text{icod}(\epsilon_1') t_1'), \mu_i'' \rangle \approx^{k-j_1} k-j_2 \langle \phi_2, \text{ prot}_{\text{ilbl}(\epsilon_1') \nu_1 \epsilon_2' \nu_{i_2}} \phi_i''(\text{icod}(\epsilon_1') t_1'), \mu_i'' \rangle : C(U_{i_2} \neg \nu g)$$

Finally, by backward preservation of the relations (Lemma 6.43) the result holds.
Case (if). \( t^U = \text{if}^g \ e_1 t^{U_1} \) then \( \epsilon_2 t^{U_2} \) else \( \epsilon_3 t^{U_3} \), with \( \phi' \triangleright t^{U_1} \in T[U_1], g' = \text{label}(U_1), e'_1 = (\phi'_1 \triangleright \overline{\text{ilbl}}(e'_1)), \phi''_1' = (\epsilon'_2(\phi'_1 \triangleright \overline{g})), (\phi'_1 \triangleright \overline{g}) \) \( \phi''_1 \triangleright t^{U_2} \in T[U_2], \phi''_3 \triangleright t^{U_3} \in T[U_3] \), \( \epsilon_1 \vdash U_1 \leq \text{Bool}_g \) and \( U = (U_2 \lor U_3) \overline{\triangleright} g \).

By definition of substitution:

\[
\rho_i(t^U) = \text{if}^g \ e_1 \rho_i(t^{U_1}) \text{ then } \epsilon_2 \rho_i(t^{U_2}) \text{ else } \epsilon_3 \rho_i(t^{U_3})
\]

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 related computations we have that:

\[
\rho_i(t^{U_1}) \mid \mu_i \xrightarrow{\phi'_1} j_1 \nu_{11} \mid \mu'_1 \implies \rho_i \approx^{k-j_1} \mu'_{2} \land \langle \phi_1 \triangleright \nu_{11}, \mu'_1 \rangle \approx^{k-j_1} \langle \phi_2 \triangleright \nu_{21}, \mu'_2 \rangle : U_1
\]

By Lemma 6.10, each \( \nu_{11} \) is either a boolean \( (b_{11})_{g_{11}} \) or a casted boolean \( \epsilon_{11} (b_{11})_{g_{11}} : U_1 \). In either case, \( U_1 \leq \text{Bool}_{g_i} \) implies \( U_1 = \text{Bool}_{g'_i} \). In case a value \( \nu_{11} \) is a casted value, then the whole term \( \rho_i(t^U) \) can take a step by (Rg), combining \( \epsilon_i \) with \( \epsilon_{11} \). Such a step either fails, or succeeds with a new combined evidence. Therefore, either:

\[
\rho_i(t^U) \mid \mu_i \xrightarrow{\phi'_1} j_i+1 \text{error}
\]

in which case we do not care since we only consider termination-insensitive noninterference, or:

\[
\rho_i(t^U) \mid \mu_i \xrightarrow{\phi'_1} j_i+1 \text{if}^g \ e_1 (b_{11})_{g_{11}} \text{ then } \epsilon_2 \rho_i(t^{U_2}) \text{ else } \epsilon_3 \rho_i(t^{U_3}) \mid \mu'_1
\]

If \( \text{obs}_{\epsilon_o} (\phi \triangleright \nu_{11}) \) does not hold, then by Lemma 6.63 \( \text{obs}_{\epsilon_o} (\phi \triangleright \epsilon_{11} (b_{11})_{g_{11}} : \text{Bool}_{g'_i}) \) is not observable. Let us assume the worst case scenario and that both execution reduce via different branches of the conditional.

Consider \( \phi''_1 = ((\phi'_1 \triangleright \overline{\text{ilbl}}(e'_1)))(\phi'_1 \triangleright \overline{g}), (\phi'_1 \triangleright \overline{g}) \). It is easy to see that if \( \phi_1 \) is not observable, then as \( \leq \epsilon_o \), \( \phi'_1 \) is not observable, and therefore by Lemma 6.55, \( \text{obs}_{\epsilon_o} (\phi''_1 \triangleright \text{ilbl}(e'_1)) \) does not hold. Therefore \( \phi'_1 \leq \epsilon_o \phi''_1 \). If \( \text{obs}_{\epsilon_o} (\epsilon_{11} \text{Bool}_{g'_i}) \) does not hold, then also by Lemma 6.55, \( \text{obs}_{\epsilon_o} (\phi''_1 \triangleright \epsilon_{11} \text{ilbl}(e'_1)) \) does not hold as well. Then

\[
\rho_i(t^U) \mid \mu_i \xrightarrow{\phi'_1} j_i+1 \text{if}^g \ e_1 (b_{11})_{g_{11}} \text{ then } \epsilon_2 \rho_i(t^{U_2}) \text{ else } \epsilon_3 \rho_i(t^{U_3}) \mid \mu'_1
\]

But because \( \text{obs}_{\epsilon_o} (\phi \triangleright \epsilon_{11} (b_{11})_{g_{11}} : \text{ Bool}_{g'_i}) \) does not hold then either \( \text{obs}_{\epsilon_o} (\phi \triangleright \phi'_1 \epsilon_{11} \text{ilbl}(e'_1)) \) does not hold.

Now consider if \( \text{obs}_{\epsilon_o} (\phi \triangleright \nu_{11}) \) holds, then \( \text{obs}_{\epsilon_o} (\phi \triangleright \epsilon_{11} (b_{11})_{g_{11}} : \text{ Bool}_{g'_i}) \) may hold or not. If its not observable we proceed like we just did for the non-observable case. Let us consider that \( \text{obs}_{\epsilon_o} (\phi \triangleright \epsilon_{11} (b_{11})_{g_{11}} : \text{ Bool}_{g'_i}) \) holds.

Then by definition of \( \leq \epsilon_o \) on boolean values, \( b_{11} = b_{21} \). Because \( b_{11} = b_{21} \), both \( \rho_i(t^U) \) and \( \rho_2(t^U) \) step into the same branch of the conditional. Let us assume the condition is true (the other case is similar):

Then by induction hypotheses \( \langle \phi_1, \rho_i(t^{U_1}), \mu'_1 \rangle \approx^{\epsilon_o} \langle \phi_2, \rho_2(t^{U_2}), \mu'_2 \rangle \), then as \( \phi_i \leq \epsilon_o \phi''_1 \), by Lemma 6.60,

\[
\langle \phi_1, \text{prot}_{\text{ilbl}(e'_1) g_{11}} \phi''_1 \epsilon_{11} \rangle \langle \epsilon_2 \rho_i(t^{U_2}), \mu'_2 \rangle \approx^{\epsilon_o} \langle \phi_2, \text{prot}_{\text{ilbl}(e'_1) g_{11}} \phi''_2 \epsilon_{11} \rangle \langle \epsilon_2 \rho_2(t^{U_2}), \mu'_2 \rangle
\]
and the result holds by backward preservation of the relations (Lemma 6.43).

Case \( \text{prot()} \). Direct by using Lemma 6.60.

□

REFERENCES

Alonzo Church. 1940. A Formulation of the Simple Theory of Types. *J. Symbolic Logic* 5, 2 (06 1940), 56–68.