Deriving a Gradual Security Language

Abstract

Abstracting Gradual Typing (AGT) is an approach to systematically construct gradually-typed counterparts for static type disciplines (Garcia et al. 2016). It consists of giving semantics to gradual types in terms of sets of static types, abstracting sets of static types back to gradual types, and using both mappings to lift the static type discipline to a gradual one.

AGT is demonstrably effective for simple type disciplines and pure languages. This paper broadens its scope by considering a language with richer features—general mutable references—a more sophisticated typing discipline—information-flow security—and a subtler notion of gradualization—security information is gradualized but simple type structure is left intact.

Our primary contribution is $\text{GSL}_{\mathsf{Ref}}$, a gradual security language, which blends static checks, where available, and dynamic checks, to enforce information-flow security. This is the first gradual *source* language, where security intent is reflected solely in types: programs do not need explicit runtime casts.

We show that $\text{GSL}_{\mathsf{Ref}}$ satisfies semantic criteria for both security-typed languages and gradually-typed languages. In particular, we prove termination-insensitive non-interference, as well as Siek et al.'s gradual typing criteria.

1. Introduction

Gradual typing (Siek and Taha 2006) has often been viewed as a means to combine the agility of dynamic languages, like Python and Ruby with the reliability of static languages like OCaml and Scala. In fact, static and dynamic are merely relative notions, and several authors have explored a relativistic view of gradual typing. Work on gradual information flow security by Disney and Flanagan (2011) and Fennell and Thiemann (2013) develop languages where only information-flow security properties are subject to a mix of dynamic and static checking. Bañados Schwerter et al. (2014) develop a language where only computational effect capabilities are gradualized. In each of these cases, the "fully-dynamic" corner of the gradual language is not dynamic at all by typical standards, but rather simply typed. However, those languages support seamless migration toward a *more precise* typing discipline that subsumes simple typing.

To elaborate these notions, we revisit the idea of gradual information-flow security. Reprising a scenario due to Disney and Flanagan (2011), consider a simply-typed program that processes human resources data:

```
1  let age: Int = 31
2  let salary: Int = 58000
3  let intToString: Int → String = ...
4  let print: String → Unit = ...
5  print(intToString(salary))
```

Though well-typed, the program has a significant error: it leaks salary data by printing it.

Information-flow security types address this and related correctness errors by enabling a programmer to statically classify program entities with *confidentiality levels*, such that the type system prevents sensitive or high-confidence data from flowing to leaky or low-confidence channels (Volpano et al. 1996). Purely-static security type systems impose a costly all-or-nothing adoption model, but *gradual* security typing supports piecemeal migration toward a static security type discipline, leaving the option to pause as befits the programmer's needs.

In our current example, the problem is that salary is high-confidence, but printing is a low-confidence channel. We can reflect this in the program's types by updating lines 2 and 4.

```
let salary: Int = 58000_H
let print: String_L → Unit = ...
```

In particular, the salary data is annotated with high-confidence H, while the `print` function's input type is given low-confidence L. This program type checks, as before, but triggers a runtime check failure at line 5.

How to fix it? Simply adding more annotations cannot help. In fact, adding reasonable security annotations to lines 2 and 3 escalates the runtime failure to a static type error.

```
let salary: Int_H = 58000_H
let intToString: Int_L → String = ...
```

Indeed, the proper fix is to print the employee's `age`, which is public knowledge, rather than salary. Editing lines 1 and 5 yield a program that both type-checks and runs.

```
let age: Int_L = 31_L
...
print(intToString(age))
```

This program is on the path to full static security typing, if so desired. How does gradual evolution work? Under the hood, simple types like Int desugar to imprecise security types like $\mathsf{Int}_?$. Then a gradual type checker statically enforces the invariants it can, introducing runtime checks where it can not.

Prior work like Disney and Flanagan (2011) and Fennell and Thiemann (2013) provide similar benefits, but with two caveats. First, both systems require the programmer to explicitly introduce runtime casts at boundaries between dynamic and static checking (e.g., lines 2 and 5 after the first change), and as such are rather *cast* languages, akin to the intermediate languages of traditional gradually-typed languages under the model set forth by Siek and Taha (2006). Second, neither work establishes the *gradual guarantee* (Siek et al. 2015), which links the gradual type discipline directly to a fully-static type discipline and ensures that runtime check failures indicate an inherent fault in a program's progression toward the static type discipline. The main contribution of this paper is to develop the first gradual security *source* language that provides these capabilities and guarantees, in a context that also supports mutable references.

The second contribution of this paper is that it systematically derives the gradual language design, starting from a fully static language, by applying the Abstracting Gradual Typing (AGT) methodology (Garcia et al. 2016), which proposes to interpret gradual types as sets of static types. AGT has been shown to be

$\ell, \ell_c \in \text{LABEL}, \quad S \in \text{TYPE}, \quad x \in \text{VAR}, \quad b \in \text{BOOL}, \quad \oplus \in \text{BOOLOP},$
$l \in \text{LOC}, \quad t \in \text{TERM}, \quad r \in \text{RAWVALUE} \quad v \in \text{VALUE}$
$\Gamma \in \text{VAR} \xrightarrow{\text{fin}} \text{TYPE}, \quad \Sigma \in \text{LOC} \xrightarrow{\text{fin}} \text{TYPE}$

S	::=	$\mathsf{Bool}_\ell \mid S \xrightarrow{\ell} {}_\ell S \mid \mathsf{Ref}_\ell\ S \mid \mathsf{Unit}_\ell$	(types)
b	::=	$\mathsf{true} \mid \mathsf{false}$	(Booleans)
r	::=	$b \mid \lambda^{\ell_c} x : S.t \mid \mathsf{unit} \mid l$	(raw values)
v	::=	r_ℓ	(values)
t	::=	$v \mid t\ t \mid t \oplus t \mid \mathsf{if}\ t\ \mathsf{then}\ t\ \mathsf{else}\ t$	
		$\mathsf{ref}^S\ t \mid !t \mid t{:=}t \mid t :: S \mid \mathsf{prot}_\ell t$	(terms)
$\oplus$	::=	$\wedge \mid \vee$	(operations)

Figure 1. $\text{SSL}_{\mathsf{Ref}}$: Syntax

effective for deriving a traditional gradual type system. This paper broadens the scope of AGT by considering a language with richer features—general mutable references—a more sophisticated typing discipline—information-flow security—and a subtler notion of gradualization—security information is gradualized but simple type structure is left intact. This case study shows the power of applying AGT to obtain guidance in the definition of sometimes evasive notions, and uncovers a number of subtleties in the application of the AGT methodology, which did not arise in prior work due to the relative simplicity of the typing discipline. It is also the first study of AGT applied to a language with a rich semantic soundness property, where type safety does not necessarily imply type soundness.

We proceed as follows. Section 2 presents $\text{SSL}_{\mathsf{Ref}}$, a fully static security typed language, with no runtime security checks. We then derive $\text{GSL}_{\mathsf{Ref}}$ by applying AGT, first focusing on gradualizing the static semantics (Sect. 3) and then the dynamic semantics (Sect. 4). We prove that the derived language is not only type safe and satisfies the gradual guarantees of Siek et al. (2015), but also that it enforces *noninterference*, the semantic soundness property of security languages (Sect. 5). This property is established via step-indexed logical relations (Ahmed 2004).

Due to space considerations, the paper only includes selected parts of definitions, and statements of formal properties. The full development with proofs is provided as anonymous supplementary material, cited in this paper as Appendix.

2. Static Security Typing with References

The AGT approach starts from a pre-existing static type discipline, so to that end this section introduces $\text{SSL}_{\mathsf{Ref}}$, a higher-order static security-typed language with references. The language is a straightforward adaptation of prior information-flow security typing disciplines (Fennell and Thiemann 2013; Heintze and Riecke 1998; Zdancewic 2002). The most significant changes include a syntax-directed type system, which is critical to the AGT approach, and a dynamic semantics that tracks confidentiality levels but performs no security checks.

Syntax. Figure 1 presents the syntax of $\text{SSL}_{\mathsf{Ref}}$, at heart a simply-typed higher-order language with references: it includes booleans, functions, unit, mutable references, and type ascription.

The language also includes some security-specific constructs and annotations. The language is parameterized over a lattice of *security labels* $\ell \in \text{LABEL}$ with partial order $\preccurlyeq$, which together denote confidentiality levels (Denning 1976). We denote the greatest and least security labels $\top$ (most private) and $\bot$ (most public), respectively. One examplar security lattice, noted by Zdancewic, is the U.S. Department of Defence classification scheme:

$$\bot = \text{Unclassified} \preccurlyeq \text{Confidential} \preccurlyeq \text{Secret} \preccurlyeq \text{Top Secret} = \top,$$

though in general, a security lattice need not be linearly ordered. Each value is annotated with a security label. Each simple type constructor is annotated with a security label as well, yielding *security types* S. Function abstractions, and their corresponding types, are annotated with an additional security label called the *latent security effect*: we explain its static semantics below.

Figure 1 highlights two forms that arise only at runtime. Mutable reference locations are standard. A *protection term* $\mathsf{prot}_\ell t$ restricts t's security effects. We explain its semantics below.

Static semantics. Figure 2 presents the type system, which is technically a type-and-effect system (Gifford and Lucassen 1986).[1] The typing judgment $\Gamma; \Sigma; \ell_c \vdash t : S$ says that the term t has type S under type environment Γ, store environment Σ, and security effect ℓ_c. The security effect, often called the *program counter* or PC (Denning 1976), reflects the least confidentiality level reference into which a given term may *write*, *i.e.* allocate a new reference or mutate an existing one (Heintze and Riecke 1998). The PC is critical to enforcing information-flow security. The intuition behind it is that a computation can leak information using mutation only if a high-security value is assigned to a low-security reference cell. The type system uses the PC to prevent any high-security computation— especially the branch of an if expression that is chosen based on a high-security Boolean—from ever depending on a computation that may leak its information. In particular, high-security code may not contain subterms or call functions that assign to low-security references.

Rule (Sx) and rule (Sl) type variable and location references, respectively, in the typical way and are unaffected by the PC. Simple values are typed as usual, but their types inherit their labels from the values themselves.

Rule (Sλ) annotates the type of a function with the latent security effect of its body, as is standard for type-and-effect systems. The greatest (i.e., best) security effect of a function can be inferred from the body, but for simplicity this type system, as is common, consults an explicit annotation ℓ'_c to determine the latent effect of a function.

Rule (Sprot) imposes a lowerbound ℓ on the PC effects available to the subterm t. Rule (S$\oplus$) types Boolean operations, setting the confidentiality level of its result to the least upper-bound, or *join*, denoted $\curlyvee$, of the levels of its operands.

Rule (Sapp) enforces the standard function type discipline, but it also imposes security restrictions. First, to prevent mutation-based security leaks, the latent effect ℓ'_c of the operator must be at least as high as the latent effects of the entire application, and at least as high as the confidentiality of the operator itself. Both restrictions are captured with the single label comparison in the premise. Second, to prevent value-based security leaks, the confidentiality of the entire expression must be at least as high as the confidentiality ℓ of the function that results from evaluating the operator. This restriciton is captured using *label stamping* on types (Heintze and Riecke 1998), *e.g.* $\mathsf{Bool}_\ell \curlyvee \ell' = \mathsf{Bool}_{(\ell \curlyvee \ell')}$.

Rule (Sapp) also appeals to the *subtyping* relation induced by the ordering on security labels (Figure 2). Subtyping lets lower-security values flow to higher-security contexts, but not vice-versa. Subtyping is driven by security labels: it is invariant on reference types, covariant on security labels, and contravariant on latent effects (Pottier and Simonet 2003).

Rule (Sif) incorporates the standard structure for a subtype discipline: the type of the expression involves the *subtyping join* $\ddot{\vee}$ of its branches. To protect against *implicit information flows*, which arise when control-flow is dictated by high-security values, the

[1] Throughout, the paper uses color to make distinctions: green is for effects and static information; orange is for the runtime information of the PC; and the blue is for the runtime store.

$\boxed{\Gamma; \Sigma; \ell_c \vdash t : S}$

$$(\text{Sx})\ \frac{x : S \in \Gamma}{\Gamma; \Sigma; \ell_c \vdash x : S} \qquad (\text{Sb})\ \frac{}{\Gamma; \Sigma; \ell_c \vdash b_\ell : \mathsf{Bool}_\ell}$$

$$(\text{Su})\ \frac{}{\Gamma; \Sigma; \ell_c \vdash \mathsf{unit}_\ell : \mathsf{Unit}_\ell} \qquad (\text{Sl})\ \frac{l : S \in \Sigma}{\Gamma; \Sigma; \ell_c \vdash l_\ell : \mathsf{Ref}_\ell\ S}$$

$$(\text{S}\lambda)\ \frac{\Gamma, x : S_1; \Sigma; \ell'_c \vdash t : S_2}{\Gamma; \Sigma; \ell_c \vdash (\lambda^{\ell'_c} x : S_1.t)_\ell : S_1 \xrightarrow{\ell'_c}_\ell S_2}$$

$$(\text{Sprot})\ \frac{\Gamma; \Sigma; \ell_c \curlyvee \ell \vdash t : S}{\Gamma; \Sigma; \ell_c \vdash \mathsf{prot}_\ell t : S \curlyvee \ell}$$

$$(\text{S}\oplus)\ \frac{\Gamma; \Sigma; \ell_c \vdash t_1 : \mathsf{Bool}_{\ell_1} \quad \Gamma; \Sigma; \ell_c \vdash t_2 : \mathsf{Bool}_{\ell_2}}{\Gamma; \Sigma; \ell_c \vdash t_1 \oplus t_2 : \mathsf{Bool}_{(\ell_1 \curlyvee \ell_2)}}$$

$$(\text{Sapp})\ \frac{\begin{array}{c}\Gamma; \Sigma; \ell_c \vdash t_1 : S_{11} \xrightarrow{\ell'_c}_\ell S_{12} \quad \Gamma; \Sigma; \ell_c \vdash t_2 : S_2 \\ S_2 <: S_{11} \qquad \ell_c \curlyvee \ell \preccurlyeq \ell'_c\end{array}}{\Gamma; \Sigma; \ell_c \vdash t_1\ t_2 : S_{12} \curlyvee \ell}$$

$$(\text{Sif})\ \frac{\Gamma; \Sigma; \ell_c \vdash t : \mathsf{Bool}_\ell \quad \Gamma; \Sigma; \ell_c \curlyvee \ell \vdash t_i : S_i}{\Gamma; \Sigma; \ell_c \vdash \mathsf{if}\ t\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 : (S_1 \ddot{\vee} S_2) \curlyvee \ell}$$

$$(\text{S::})\ \frac{\Gamma; \Sigma; \ell_c \vdash t : S_1 \quad S_1 <: S_2}{\Gamma; \Sigma; \ell_c \vdash t :: S_2 : S_2}$$

$$(\text{Sref})\ \frac{\Gamma; \Sigma; \ell_c \vdash t : S' \quad S' <: S \quad \ell_c \preccurlyeq \mathsf{label}(S)}{\Gamma; \Sigma; \ell_c \vdash \mathsf{ref}^S\ t : \mathsf{Ref}_\bot\ S}$$

$$(\text{Sderef})\ \frac{\Gamma; \Sigma; \ell_c \vdash t : \mathsf{Ref}_\ell\ S}{\Gamma; \Sigma; \ell_c \vdash !t : S \curlyvee \ell}$$

$$(\text{Sasgn})\ \frac{\begin{array}{c}\Gamma; \Sigma; \ell_c \vdash t_1 : \mathsf{Ref}_\ell\ S_1 \quad \Gamma; \Sigma; \ell_c \vdash t_2 : S_2 \\ S_2 <: S_1 \qquad \ell_c \curlyvee \ell \preccurlyeq \mathsf{label}(S_1)\end{array}}{\Gamma; \Sigma; \ell_c \vdash t_1 := t_2 : \mathsf{Unit}_\bot}$$

$\boxed{S <: S}$

$$\frac{\ell \preccurlyeq \ell'}{\mathsf{Bool}_\ell <: \mathsf{Bool}_{\ell'}} \qquad \frac{\ell \preccurlyeq \ell'}{\mathsf{Unit}_\ell <: \mathsf{Unit}_{\ell'}}$$

$$\frac{S'_1 <: S_1 \quad S_2 <: S'_2 \quad \ell \preccurlyeq \ell' \quad \ell'_c \preccurlyeq \ell_c}{S_1 \xrightarrow{\ell_c}_\ell S_2 <: S'_1 \xrightarrow{\ell'_c}_{\ell'} S'_2}$$

$$\frac{\ell \preccurlyeq \ell'}{\mathsf{Ref}_\ell\ S <: \mathsf{Ref}_{\ell'}\ S}$$

Figure 2. SSL_{Ref}: Static Semantics

type of the expression is stamped to incorporate the confidentiality level of the branch condition, ℓ. To prevent effect-based leaks, each branch of the conditional is typed with a PC that incorporates the confidentiality of the branch condition.

Rule (S::) is typical for ascription, requiring the ascribed type to be a supertype of the inner term. The SSL_{Ref} language does not have an explicit effect ascription form (Bañados Schwerter et al. 2014), but an effect ascription $t :: \ell_c$ can be simulated using the expression $(\lambda^{\ell_c} x : \mathsf{Unit}_\bot.t)_\bot\ \mathsf{unit}_\bot$.

Rule (Sderef) stamps the confidentiality level of the referenced term on the resulting type. Rule (Sl) is standard.

The last two rules, which perform write effects, are constrained by the security effect of the typing judgment to prevent leaks through the store. Rule (Sref) restricts the confidentiality of the type

$\boxed{t \mid \mu \xrightarrow{\ell_r} t \mid \mu}$ **Notion of Reduction**

$$b_{1\,\ell_1} \oplus b_{2\,\ell_2} \mid \mu \xrightarrow{\ell_r} (b_1\ [\![\oplus]\!]\ b_2)_{(\ell_1 \curlyvee \ell_2)} \mid \mu$$

$$(\lambda^{\ell'_c} x : S.t)_\ell\ v \mid \mu \xrightarrow{\ell_r} \mathsf{prot}_\ell[v/x]t \mid \mu$$

$$\mathsf{if}\ \mathsf{true}_\ell\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \mid \mu \xrightarrow{\ell_r} \mathsf{prot}_\ell t_1 \mid \mu$$

$$\mathsf{if}\ \mathsf{false}_\ell\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \mid \mu \xrightarrow{\ell_r} \mathsf{prot}_\ell t_2 \mid \mu$$

$$\mathsf{prot}_\ell v \mid \mu \xrightarrow{\ell_r} v \curlyvee \ell \mid \mu$$

$$\mathsf{ref}^S\ v \mid \mu \xrightarrow{\ell_r} l_{\ell_r} \mid \mu[l \mapsto v \curlyvee \ell_r] \text{ where } l \notin dom(\mu)$$

$$!l_\ell \mid \mu \xrightarrow{\ell_r} v \curlyvee \ell \mid \mu \text{ where } \mu(l) = v$$

$$l_\ell := v \mid \mu \xrightarrow{\ell_r} \mathsf{unit}_\bot \mid \mu[l \mapsto v \curlyvee \ell_r \curlyvee \ell]$$

$$v :: S \mid \mu \xrightarrow{\ell_r} v \curlyvee label(S) \mid \mu$$

$\boxed{t \mid \mu \xmapsto{\ell_r} t \mid \mu}$ **Reduction**

$$(\text{R}\rightarrow)\ \frac{t_1 \mid \mu_1 \xrightarrow{\ell_r} t_2 \mid \mu_2}{t_1 \mid \mu_1 \xmapsto{\ell_r} t_2 \mid \mu_2} \qquad (\text{R}f)\ \frac{t_1 \mid \mu_1 \xmapsto{\ell_r} t_2 \mid \mu_2}{f[t_1] \mid \mu_1 \xmapsto{\ell_r} f[t_2] \mid \mu_2}$$

$$(\text{Rprot})\ \frac{t_1 \mid \mu_1 \xmapsto{\ell_r \curlyvee \ell} t_2 \mid \mu_2}{\mathsf{prot}_\ell t_1 \mid \mu_1 \xmapsto{\ell_r} \mathsf{prot}_\ell t_2 \mid \mu_2}$$

Figure 3. SSL_{Ref}: Label Tracking Dynamic Semantics

of the stored value S to be at least as high as the current PC, reflecting the restriction imposed by the security effect. The resulting reference has the lowest security $\bot$ because it is both newly minted and cannot leak information: the type of the stored content is already known, and its confidentiality protects it from further prying. Analogously, rule (Sasgn) ensures that the confidentiality of the assigned value is higher than both the current PC and the location itself. The result of assignment has $\bot$ security because the resulting Unit-typed value is newly minted and cannot leak information: its type completely determines its value.

An SSL_{Ref} source program t is well-typed if $\cdot; \cdot; \bot \vdash t : S$.

Dynamic semantics. We now turn to the dynamic semantics of SSL_{Ref}. When security typing is a fully static typing discipline, programs can be run on a standard runtime without any additional security-enforcing machinery. Type *safety*—well-typed terms do not get stuck—is guaranteed by the underlying simple type discipline, because if one eliminates security-related annotations, the language boils down to a run-of-the-mill simply-typed higher-order language with references. However, to establish the *soundness* of security typing—high-confidentiality computations have no effect on low-confidentiality observations—one must characterize computations and their resulting values with respect to their confidentiality. To this end, the dynamic semantics of SSL_{Ref} explicitly *tracks* security labels as programs evaluate, but never *checks* them. The proof of noninterference ensures that any such checks would succeed based solely on static typing, and those checks are no help in proving noninterference, so omitting them substantially simplifies the semantics at no cost. Label tracking amounts to a conservative non-standard semantics for confidentiality levels, whose soundness is confirmed by the noninterference proof.

Figure 3 presents the label-tracking dynamic semantics as a structural operational semantics. Judgment $t_1 \mid \mu_1 \xrightarrow{\ell_r} t_2 \mid \mu_2$ says that a term t_1 and store μ_1 together step under PC ℓ_r to t_2 and μ_2 respectively. The core behavioral semantics of the language are typical for an ML-like language, so we focus on the semantics of security tracking.

The PC, which reflects its static semantics counterpart, represents the security-level of the term currently under evaluation. It is used to track (a conservative approximation of) the security level of reads from and writes to the store, as well as the security of values returned from higher-security contexts to a lower-security ones.

Protection terms $\mathsf{prot}_\ell t$ control the current PC (Fennell and Thiemann 2013). Rule (Rprot) elevates the PC for the dynamic extent of t. Once the computation produces a value, the corresponding notion of reduction stamps the value with the protected label ℓ, in case the contents leak information to a context that lacks the confidentiality of ℓ. Label stamping on values is similar to stamping on types: $r_\ell \curlyvee \ell' = r_{(\ell \curlyvee \ell')}$.

Protection terms do not exist in source programs: they are introduced by control operations, *i.e.* function calls and conditionals. The intuition is that calling a function or destructing a Boolean of confidentiality ℓ may leak information about the identity of the function or Boolean respectively. As such, the context of the resulting computation ought communicate (via mutation) with only reference cells that have high-enough security, and the value of the computation is classified as well.[2]

Apart from prot, all expressions preserve the current program counter for their subterms. We capture this with evaluation frames and the frame rule (Rf):

$$\begin{array}{rcll} f & ::= & \Box \oplus t \mid v \oplus \Box \mid \Box\ t \mid v\ \Box & \text{(frames)} \\ & & \mathsf{if}\ \Box\ \mathsf{then}\ t\ \mathsf{else}\ t \mid \Box :: S & \\ & & \mathsf{ref}^S\ \Box \mid !\Box \mid \Box{:=}t \mid v{:=}\Box & \end{array}$$

The semantics for function calls ignores the latent effect ℓ'_c. That annotation is a future promise to the *type system* that the ensuing computation will not violate the stated confidentiality level, but it does not affect the immediate computation: the label stamp of the function determines the minimum confidentiality.

An ascription $e :: S$ solely relates to the final value flow of an expression, so it merely stamps the resulting value. Ascriptions do not step to protection terms because protection terms impact the ensuing computation.

A Boolean operation stamps the join of the security levels of both operands onto the resulting value. This reflects the possibility that information about both arguments might be leaked.[3]

A value stored in a new reference cell inherits confidentiality from the PC to track leaks. The resulting location ℓ is stamped with the PC, which imposes that confidentiality on any future value assigned to it.

The value retrieved by dereferencing a location inherits the confidentiality of the location. The reasoning corresponds to the final result of a function call: a low-security observer should not notice if the location is replaced with some other high-security location, but the contents of the reference cell may leak that information, so the value is also classified.

Upon assignment, the stored value inherits confidentiality from both the PC and the location. This behavior simultaneously tracks the confidentiality of the location and the induced security effect.

[2] Zdancewic (2002) observes that e.g., $\mathsf{if}\ x\ \mathsf{then}\ e_L\ \mathsf{else}\ e_L$ leaks no information about Boolean $x : \mathsf{Bool}_H$ so could be deemed low-security, but security type systems must be conservative for the sake of tractability.

[3] Though not necessarily: the expression $x \wedge \mathsf{false}_L$ leaks no high-security information about $x : \mathsf{Bool}_H$.

Properties. SSL$_{\mathsf{Ref}}$ is type safe: we establish this using the progress-preservation approach of Wright and Felleisen (1994). Progress mirrors the corresponding argument for the underlying simple type discipline, since the runtime semantics includes no security checks. To prove preservation, on the other hand, we show that after each reduction step the resulting term satisfies the security type system of Figure 2. We omit the statement and proof of type safety because they are commonplace (see Appendix A.2). In the next section we use the type safety argument to derive a dynamic semantics for a gradual security typed language.

The most crucial property of a security-typed language like SSL$_{\mathsf{Ref}}$ is the *soundness* of security typing, *i.e.* noninterference: well-typed programs have no forbidden information flows. We do not formally state and prove noninterference at this point, because it follows from the noninterference property of the gradual version of SSL$_{\mathsf{Ref}}$—both languages provably coincide on fully-static terms.

Having developed a static security-typed language as a basis, we proceed to derive a gradually-typed counterpart that supports the seamless transition between static and dynamic checking of information-flow security, while retaining a simple type discipline.

3. Gradualizing the Static Semantics

This section describes how to derive the static semantics of GSL$_{\mathsf{Ref}}$, a gradually-typed information-flow security language, starting from the definition of SSL$_{\mathsf{Ref}}$. The derivation of GSL$_{\mathsf{Ref}}$ is systematic, following the Abstracting Gradual Typing (AGT) approach by Garcia et al. (2016). Sec. 4 develops the dynamic semantics. The high-level structure of the approach is as follows. First, a notion of *gradual types* is introduced and given meaning formally in terms of the types of the static language (Sec. 3.1). This meaning is used to *lift* first functions and predicates on static types, and their constituents, to *gradual* functions and *consistent* predicates on gradual types, and *their* constituents (Sec. 3.2). These components are used to lift the static type system to a gradual type system (Sect. 3.3). The resulting type system is a conservative extension of the static system, in that it accepts and rejects programs in the static language just as before.

3.1 From Gradual Labels to Gradual Types

To gradualize the security-related part of types, we introduce an unknown security label ? (Fennell and Thiemann 2013). A *gradual label* $\tilde{\ell}$ is either a static label ℓ or the unknown label ?. We define *gradual security types* by annotating each type constructor with a gradual label. The syntax of GSL$_{\mathsf{Ref}}$ is otherwise identical to that of SSL$_{\mathsf{Ref}}$ (see Appendix B.3–Figure 10).

Having established the syntax of GSL$_{\mathsf{Ref}}$, we proceed to develop its semantics, beginning with gradual labels. Prior approaches to gradual security typing treat the unknown label as a new top element in the security lattice, a technique that is reminiscent of Thatte's quasi-static typing approach to combining static and dynamic checking (Thatte 1990), which originally inspired Gradual Typing (Siek and Taha 2006). Instead, following the AGT approach, we directly give meaning to *all* gradual labels in terms of the original security lattice. The key intuition is that a gradual label ℓ represents a definite static security label, while the unknown label ? represents any label whatsoever. We formalize this with a *concretization* function.

Definition 1 (Label Concretization)**.**
Let $\gamma_\ell : \textsc{Glabel} \to \mathcal{P}(\textsc{Label})$ *be defined as follows:*

$$\gamma_\ell(\ell) = \{\, \ell \,\}$$
$$\gamma_\ell(?) = \textsc{Label}$$

Concretization induces a notion of *precision* among gradual labels, which characterizes the static information content of a label.

Definition 2 (Label Precision). *$\tilde{\ell}_1$ is less imprecise than $\tilde{\ell}_2$, notation $\tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2$, if and only if $\gamma_\ell(\tilde{\ell}_1) \subseteq \gamma_\ell(\tilde{\ell}_2)$.*
Equivalently: $\tilde{\ell} \sqsubseteq ?$ and $\tilde{\ell} \sqsubseteq \tilde{\ell}$.

In addition to concretization, the AGT methodology also requires a corresponding *abstraction* function, from arbitrary sets of static labels to single gradual labels, which summarizes the information content of the given static labels as best as possible. The corresponding abstraction function, which is fully determined by definition of concretization, follows:[4]

Definition 3 (Label Abstraction). *Let $\alpha_\ell : \mathcal{P}(\text{LABEL}) \to \text{GLABEL}$ be defined as follows:*

$$\alpha_\ell(\{\,\ell\,\}) = \ell \qquad \alpha_\ell(\emptyset) \text{ is undefined} \qquad \alpha_\ell(\widehat{\ell}) = ? \text{ otherwise}$$

The γ_ℓ and α_ℓ functions share a tight connection, which is reflected in two properties, soundness and optimality. Soundness means that α_ℓ always produces a gradual type whose concretization overapproximates the information in the original set. Optimality means that the α_ℓ function in particular is the best sound approximation function possible, given that it may only use the given gradual labels to represent its summary.

Proposition 1 (α_ℓ is Sound). *If $\widehat{\ell}$ is a non-empty set of labels, then $\widehat{\ell} \subseteq \gamma_\ell(\alpha_\ell(\widehat{\ell}))$.*

Proposition 2 (α_ℓ is Optimal). *If $\widehat{\ell} \subseteq \gamma_\ell(\tilde{\ell})$ then $\alpha_\ell(\widehat{\ell}) \sqsubseteq \tilde{\ell}$.*

We intend the imprecision of gradual security types to simply reflect the imprecision of their labels. To achieve this, we compatibly extend label concretization to types.

Definition 4 (Type Concretization). *Let $\gamma_S : \text{GTYPE} \to \mathcal{P}(\text{TYPE})$ be defined as follows:*

$$\begin{aligned}
\gamma_S(\mathsf{Bool}_{\tilde{\ell}}) &= \{\,\mathsf{Bool}_\ell \mid \ell \in \gamma_\ell(\tilde{\ell})\,\} \\
\gamma_S(\mathsf{Unit}_{\tilde{\ell}}) &= \{\,\mathsf{Unit}_\ell \mid \ell \in \gamma_\ell(\tilde{\ell})\,\} \\
\gamma_S(\widetilde{S}_1 \xrightarrow{\tilde{\ell}_c}_{\tilde{\ell}} \widetilde{S}_2) &= \gamma_S(\widetilde{S}_1)^{\gamma_\ell(\tilde{\ell}_c)}_{\gamma_\ell(\tilde{\ell})} \gamma_S(\widetilde{S}_2) \\
\gamma_S(\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}) &= \{\,\mathsf{Ref}_\ell\ S \mid \ell \in \gamma_\ell(\tilde{\ell}), S \in \gamma_S(\widetilde{S})\,\}
\end{aligned}$$

Concretization once again induces a notion of precision as well as a unique sound and optimal abstraction function.

Definition 5 (Type Precision). *$\widetilde{S}_1$ is less imprecise than $\widetilde{S}_2$, notation $\widetilde{S}_1 \sqsubseteq \widetilde{S}_2$, if and only if $\gamma_S(\widetilde{S}_1) \subseteq \gamma_S(\widetilde{S}_2)$. Inductively:*

$$\frac{\tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2}{\mathsf{Bool}_{\tilde{\ell}_1} \sqsubseteq \mathsf{Bool}_{\tilde{\ell}_2}} \qquad \frac{\widetilde{S}_{11} \sqsubseteq \widetilde{S}_{21} \quad \widetilde{S}_{12} \sqsubseteq \widetilde{S}_{22} \quad \tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2 \quad \tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}}{\widetilde{S}_{11} \xrightarrow{\tilde{\ell}_{c1}}_{\tilde{\ell}_1} \widetilde{S}_{12} \sqsubseteq \widetilde{S}_{21} \xrightarrow{\tilde{\ell}_{c2}}_{\tilde{\ell}_2} \widetilde{S}_{22}}$$

$$\frac{\tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2}{\mathsf{Unit}_{\tilde{\ell}_1} \sqsubseteq \mathsf{Unit}_{\tilde{\ell}_2}} \qquad \frac{\tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2 \quad \widetilde{S}_1 \sqsubseteq \widetilde{S}_2}{\mathsf{Ref}_{\tilde{\ell}_1}\ \widetilde{S}_1 \sqsubseteq \mathsf{Ref}_{\tilde{\ell}_2}\ \widetilde{S}_2}$$

[4] Throughout, we use round hats $\widehat{X}$ to represent sets of X and tilde hats $\widetilde{X}$ to represent gradual X.

Definition 6 (Type Abstraction). *Let the abstraction function $\alpha_S : \mathcal{P}(\text{TYPE}) \to \text{GTYPE}$ be defined as:*

$$\begin{aligned}
\alpha_S(\{\,\overline{\mathsf{Bool}_{\ell_i}}\,\}) &= \mathsf{Bool}_{\alpha_\ell(\{\,\overline{\ell_i}\,\})} \\
\alpha_S(\{\,\overline{\mathsf{Unit}_{\ell_i}}\,\}) &= \mathsf{Unit}_{\alpha_\ell(\{\,\overline{\ell_i}\,\})} \\
\alpha_S(\{\,\overline{S_{i1} \xrightarrow{\ell_{ci}}_{\ell_i} S_{i2}}\,\}) &= \alpha_S(\{\,\overline{S_{i1}}\,\})^{\alpha_\ell(\{\,\overline{\ell_{ci}}\,\})}_{\alpha_\ell(\{\,\overline{\ell_i}\,\})} \alpha_S(\{\,\overline{S_{i2}}\,\}) \\
\alpha_S(\{\,\overline{\mathsf{Ref}_{\ell_i}\ S_i}\,\}) &= \mathsf{Ref}_{\alpha_\ell(\{\,\overline{\ell_i}\,\})}\ \alpha_S(\{\,\overline{S_i}\,\})
\end{aligned}$$

$$\alpha_S(\widehat{S}) \text{ is undefined otherwise}$$

Proposition 3 (α_S is Sound). *If $\widehat{S}$ is a valid set of security types, then $\widehat{S} \subseteq \gamma_S(\alpha_S(\widehat{S}))$.*

Proposition 4 (α_S is Optimal). *If $\widehat{S}$ is a valid set of security types and $\widehat{S} \subseteq \gamma_S(\widetilde{S})$ then $\alpha_S(\widehat{S}) \sqsubseteq \widetilde{S}$.*

We only abstract *valid* sets of security types, *i.e.* in which elements only differ by security labels. For example, abstraction $\alpha_S(\{\,\mathsf{Bool}_{\ell_1}, \mathsf{Int}_{\ell_2}\,\})$ is undefined, indicating a type inconsistency.

3.2 Consistent Predicates and Functions

This section shows how the definitions above can be used to *lift* aspects of $\text{SSL}_{\mathsf{Ref}}$ to their counterparts in $\text{GSL}_{\mathsf{Ref}}$. The static semantics of $\text{SSL}_{\mathsf{Ref}}$ relies on security label ordering and the induced subtyping relation on security types, as well as on joins and meets of security labels and types. The abstract interpretation framework of AGT dictates the *definition* of the lifting of all these operators. We then derive (via inductive reasoning) their equivalent algorithmic and inductive *characterizations*.

In AGT, predicates on static entities (labels ℓ, types S, etc.) are lifted to *consistent* predicates on their gradual counterparts ($\tilde{\ell}$, $\widetilde{S}$, etc.). In essence, each gradual entity represents some set of possible static entities, and a consistent predicate holds among them so long as the underlying static predicate could *plausibly* hold. For instance, consistent ordering on gradual labels is defined as follows:

Definition 7 (Consistent label ordering). *$\tilde{\ell}_1 \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_2$ if and only if $\ell_1 \preccurlyeq \ell_2$ for some $(\ell_1, \ell_2) \in \gamma_\ell(\tilde{\ell}_1) \times \gamma_\ell(\tilde{\ell}_2)$.*

Consistent ordering conservatively extends static label ordering. To see this, recall that every static label also counts as a gradual label, *i.e.*, $\text{LABEL} \subseteq \text{GLABEL}$, and their concretizations are singleton sets. Therefore, $\ell_1 \preccurlyeq \ell_2$ if and only if $\ell_1 \mathrel{\widetilde{\preccurlyeq}} \ell_2$. Conservative extension is key throughout to the concept of graduality.

Consistent ordering holds universally for the unknown label. For example $? \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}$ and vice-versa, since there always exists *some* pair of representative static labels for which the ordering holds. This imprecision introduces slack into the type system, and must be balanced with runtime checks (Sec. 4).

We use the same conception of consistent lifting to define *consistent subtyping*.

Definition 8 (Consistent subtyping). *$\widetilde{S}_1 \mathrel{\widetilde{<:}} \widetilde{S}_2$ if and only if $S_1 <: S_2$ for some $(S_1, S_2) \in \gamma_S(\widetilde{S}_1) \times \gamma_S(\widetilde{S}_2)$. Inductively:*

$$\frac{\tilde{\ell} \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}'}{\mathsf{Bool}_{\tilde{\ell}} \mathrel{\widetilde{<:}} \mathsf{Bool}_{\tilde{\ell}'}} \qquad \frac{\tilde{\ell} \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}'}{\mathsf{Unit}_{\tilde{\ell}} \mathrel{\widetilde{<:}} \mathsf{Unit}_{\tilde{\ell}'}}$$

$$\frac{\tilde{\ell} \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}' \quad \widetilde{S}_1 \mathrel{\widetilde{<:}} \widetilde{S}_2 \quad \widetilde{S}_2 \mathrel{\widetilde{<:}} \widetilde{S}_1}{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1 \mathrel{\widetilde{<:}} \mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S}_2} \qquad \frac{\widetilde{S}'_1 \mathrel{\widetilde{<:}} \widetilde{S}_1 \quad \widetilde{S}_2 \mathrel{\widetilde{<:}} \widetilde{S}'_2 \quad \tilde{\ell} \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}' \quad \tilde{\ell}'_c \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c}{\widetilde{S}_1 \xrightarrow{\tilde{\ell}_c}_{\tilde{\ell}} \widetilde{S}_2 \mathrel{\widetilde{<:}} \widetilde{S}'_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}'} \widetilde{S}'_2}$$

Lifting functions on labels and types follows by abstract interpretation as well. For instance, the join of two gradual labels is the best abstraction of the set of all possible joins of the represented static labels.

Definition 9 (Gradual label join)**.**
$\tilde{\ell}_1 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_2 = \alpha_\ell(\{\, \ell_1 \curlyvee \ell_2 \mid (\ell_1, \ell_2) \in \gamma_\ell(\tilde{\ell}_1) \times \gamma_\ell(\tilde{\ell}_2) \,\})$.

The intuition behind this definition is as follows. The gradual labels $\tilde{\ell}_i$ represent sets of possible static labels. We apply concretization to determine which *sets* of static labels they could plausibly be. Then, by applying the static label join operator $\curlyvee$ to every possible pair of labels, we determine what concrete set of static labels could plausibly be the result. Finally we use abstraction α_ℓ to summarize the result as best as any given gradual label could do. This means that our initial design decision, the meaning of gradual labels, restricts the precision of our static information.

An equational characterization of $\widetilde{\curlyvee}$ sheds significant light on gradual join:

$$\begin{aligned} \top \mathbin{\widetilde{\curlyvee}} ? = ? \mathbin{\widetilde{\curlyvee}} \top &= \top \\ \tilde{\ell} \mathbin{\widetilde{\curlyvee}} ? = ? \mathbin{\widetilde{\curlyvee}} \tilde{\ell} &= ? \text{ if } \tilde{\ell} \neq \top \\ \ell_1 \mathbin{\widetilde{\curlyvee}} \ell_2 &= \ell_1 \curlyvee \ell_2 \end{aligned}$$

The unknown label disappears when joined with $\top$, while it otherwise survives all joins. This is an emergent property of the AGT approach: the authors did not anticipate it.

Aggregate lifting. The static typing rules of Figure 2 do not all use atomic premises consisting of a single operator. For instance, the (Sapp) rule uses the premise $\ell_c \curlyvee \ell \preccurlyeq \ell'_c$, which uses both join and ordering on labels. One might be tempted to lift this premise compositionally as $\tilde{\ell}_c \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \mathbin{\widetilde{\preccurlyeq}} \tilde{\ell}'_c$ but this design loses precision and thereby fails to detect some evident failures.

To prevent this, the combination $\ell_c \curlyvee \ell \preccurlyeq \ell'_c$ must be treated as a ternary predicate, and lifted simultaneously.

Definition 10 (Consistent bounding)**.** $\widetilde{\tilde{\ell}_1 \curlyvee \tilde{\ell}_2 \preccurlyeq \tilde{\ell}_3}$ *if and only if* $\ell_1 \curlyvee \ell_2 \preccurlyeq \ell_3$ *for some* $(\ell_1, \ell_2, \ell_3) \in \gamma_\ell(\tilde{\ell}_1) \times \gamma_\ell(\tilde{\ell}_2) \times \gamma(\tilde{\ell}_3)$

To illustrate, suppose a lattice $\bot \preccurlyeq \mathsf{L} \preccurlyeq \mathsf{H} \preccurlyeq \top$. Then the predicate $\mathsf{H} \mathbin{\widetilde{\curlyvee}} ? \mathbin{\widetilde{\preccurlyeq}} \mathsf{L}$ holds even though there is no static label ℓ such that $\mathsf{H} \curlyvee \ell \preccurlyeq \mathsf{L}$, because $\mathsf{H} \curlyvee \ell$ is by definition at least as high as H. Curiously enough, using the compositional lifting would break neither type safety nor any of the gradual guarantees, but it would break noninterference.

Garcia et al. (2016) explicitly warn against the temptation of blindly using compositional lifting, but their typing discipline ultimately admits compositionality. To the best of our knowledge, this is the first instance of non-compositionality in applying AGT.

3.3 Static Semantics of GSL$_{\mathsf{Ref}}$

The type system of GSL$_{\mathsf{Ref}}$ is directly derived from the SSL$_{\mathsf{Ref}}$ type system (Figure 2) by lifting labels, types, predicates, and functions to their gradual counterparts. For instance:

$$(\widetilde{S}\text{if})\ \frac{\Gamma; \Sigma; \tilde{\ell}_c \vdash t : \mathsf{Bool}_{\tilde{\ell}} \qquad \Gamma; \Sigma; \tilde{\ell}_c \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \vdash t_1 : \widetilde{S}_1 \qquad \Gamma; \Sigma; \tilde{\ell}_c \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \vdash t_2 : \widetilde{S}_2}{\Gamma; \Sigma; \tilde{\ell}_c \vdash \mathsf{if}\ t\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 : (\widetilde{S}_1 \mathbin{\widetilde{\vee}} \widetilde{S}_2) \mathbin{\widetilde{\curlyvee}} \tilde{\ell}}$$

Note however that because of the non-compositional lifting of consistent bounding discussed above, the ($\widetilde{S}$app) rule uses an aggregate lifting of the corresponding premise:

$$(\widetilde{S}\text{app})\ \frac{\Gamma; \Sigma; \tilde{\ell}_c \vdash t_1 : \widetilde{S}_{11} \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_{12} \qquad \Gamma; \Sigma; \tilde{\ell}_c \vdash t_2 : \widetilde{S}_2 \qquad \widetilde{S}_2 \mathbin{\widetilde{\lesssim}} \widetilde{S}_{11} \qquad \widetilde{\tilde{\ell} \curlyvee \tilde{\ell}_c \preccurlyeq \tilde{\ell}'_c}}{\Gamma; \Sigma; \tilde{\ell}_c \vdash t_1\ t_2 : \widetilde{S}_{12} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}}$$

The same care must be taken for defining rule ($\widetilde{S}$asgn). The complete system is given in Appendix B.3–Figure 11. As with SSL$_{\mathsf{Ref}}$, a GSL$_{\mathsf{Ref}}$ source program t is well-typed if $\cdot; \cdot; \bot \vdash t : \widetilde{S}$.

The AGT approach yields a gradual counterpart to an underlying static type system that satisfies a number of desirable properties. First, the gradual type system is a conservative extension of the static type system (we use $\vdash_S$ to denote the SSL$_{\mathsf{Ref}}$ type system); in other words, both systems coincide on fully-annotated terms.

Proposition 5 (Equivalence for fully-annotated terms)**.** *For any* $t \in \textsc{Term}$, $.; \Sigma; \ell_c \vdash_S t : S$ *if and only if* $.; \Sigma; \ell_c \vdash t : S$

More interestingly, the static gradual guarantee of Siek et al. (2015) relates terms of different precision. Precision on terms $t_1 \sqsubseteq t_2$ is the natural lifting of type precision to terms.

Proposition 6 (Static gradual guarantee)**.** *If* $.; .; \tilde{\ell}_{c1} \vdash t_1 : \widetilde{S}_1$, $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$ *and* $t_1 \sqsubseteq t_2$*, then* $.; .; \tilde{\ell}_{c2} \vdash t_2 : \widetilde{S}_2$ *and* $\widetilde{S}_1 \sqsubseteq \widetilde{S}_2$.

4. Gradualizing the Dynamic Semantics

This section develops the dynamic semantics of GSL$_{\mathsf{Ref}}$ using the AGT approach. To produce the dynamic semantics for a gradual language from a static one, AGT views the type safety proof of the static language as defining a reduction relation on static type derivations (Howard 1980) and *lifts* these reductions to gradual type derivations. The result is a dynamic semantics for gradual programs that reifies definite deductions about relationships between static entities like types and labels into dynamic checks. For example, preservation of typeability depends on the transitivity of equality, subtyping, etc. Lifting this to the gradual language introduces runtime checks of *consistent transitivity*, *consistent equality*, *consistent subtyping*, and so forth. We elaborate these concepts below.

These checks arise by deducing *evidence* for consistent judgments during reduction of gradual derivations. We first briefly recall the notion of *evidence* introduced by Garcia et al. (2016), which plays a central role in setting up the dynamic semantics. We then define evidence for security typing, which requires particular consideration in order to retain precise enough knowledge to ensure noninterference.

4.1 Evidence for Consistent Judgments

The type safety proof for SSL$_{\mathsf{Ref}}$ critically relies on the *transitivity* of static predicates like label ordering $\ell_1 \preccurlyeq \ell_2$ and subtyping $S_1 <: S_2$. However, due to the imprecision of gradual labels and types transitivity does not hold for consistent label ordering $\tilde{\ell}_1 \preccurlyeq \tilde{\ell}_2$ and consistent subtyping $\widetilde{S}_1 \mathbin{\widetilde{\lesssim}} \widetilde{S}_2$. For instance, $\mathsf{Bool}_\top \mathbin{\widetilde{\lesssim}} \mathsf{Bool}_?$ and $\mathsf{Bool}_? \mathbin{\widetilde{\lesssim}} \mathsf{Bool}_\bot$, but $\mathsf{Bool}_\top \mathbin{\widetilde{\lesssim}} \mathsf{Bool}_\bot$ is neither true nor desired. In essence, gradual typing is about plausibility, and as a program reduces, the plausibility of it being well-typed might become impossible to justify.

To determine when transitivity can be applied to two consistent judgments, AGT introduces the notion of *evidence*, denoted ε, which represents the most precise knowledge about the plausible static types that support the consistent judgment. In other words, evidence characterizes *why* a given consistent judgment holds. Given two consistent judgments, the transitive judgment holds if their associated evidences can be combined.

The key intuition is that an n-ary consistent judgment on gradual types (labels, etc.) holds if there *exist* n-tuples of static types

(labels, etc.) that 1) fall within the meanings of the given gradual types, and 2) are consistent with all deductions that occurred up to a particular moment at runtime. Evidence abstracts these n-tuples of static types, which evolve as a program reduces. We introduce extended judgments like $\varepsilon \vdash \widetilde{S}_1 \lesssim \widetilde{S}_2$, which associate particular runtime evidence with particular consistent judgments. In the example above, ε represents which pairs of static types (S_1, S_2) are still candidates for satisfying the consistent judgment $\widetilde{S}_2 \lesssim \widetilde{S}_2$.

In essence, a gradual dynamic semantics tries to refute the plausibility of type safety at runtime. We now instantiate these ideas for gradual security typing.

4.2 Precise Evidence for Consistent Security Judgments

Garcia et al. (2016) use a straightforward abstraction of n-ary predicates. Their *coordinate-wise* abstraction represents a set of n-tuples of static types as a single n-tuple of gradual types (Cousot and Cousot 1994), i.e.

$$\gamma^n(\widetilde{S}_1, \widetilde{S}_2, \ldots, \widetilde{S}_n) = \gamma(\widetilde{S}_1) \times \gamma(\widetilde{S}_2) \times \cdots \times \gamma(\widetilde{S}_n).$$

We initially used analogous abstractions to represent evidence for consistent judgments on labels and types, but such abstractions are not precise enough to satisfy the goals of gradual security typing. In particular, the resulting language straightforwardly satisfies dynamic criteria for gradual typing (Prop. 9) but fails to satisfy noninterference (Prop. 10).

The first harbinger of this challenge surfaced in Section 3.2 when we could not apply compositional lifting to label ordering combined with the join operator, such as for the $\ell_c \curlyvee \ell \preccurlyeq \ell_c{}'$ premise in Rule (Sapp) of Figure 2. Too often, the abstraction reverts to the *unknown* label ?, which remembers nothing.

We already consider this combination as a ternary predicate in the static semantics, but as a program reduces, and ordering judgments are combined via (consistent) transitivity arguments, additional joins and meets of labels can occur in a comparison, which threaten to accrete imprecision. To retain sufficent precision to ensure noninterference, we must somehow represent evidence for label ordering predicates of the general form:

$$\widetilde{F_1(\overline{\ell}_i) \preccurlyeq F_2(\overline{\ell}_j)}$$

where a function F denotes a combination of meets and joins of labels.

To this end, we introduce a more precise abstraction for sets of labels—and in turn, types—and use this new abstraction to induce our notions of evidence. Specifically, we use an *interval* abstraction, which is directly analogous to numeric intervals. Intervals are well-behaved for combinations of labels F, and sufficiently precise to establish noninterference.

Definition 11 (Label Interval)**.** *A label interval $\imath = [\ell_1, \ell_2]$ is a bounded gradual label where the static label ℓ_1 is the lower bound and the static label ℓ_2 is the upper bound.*

We formally specify the meaning of label intervals by their concretization to sets of static labels.

Definition 12 (Interval Concretization)**.**
Let $\gamma_\imath : \textsc{Label}^2 \to \mathcal{P}(\textsc{Label})$ be defined as follows:

$$\gamma_\imath([\ell_1, \ell_2]) = \{\ell \mid \ell \in \textsc{Label}, \ell_1 \preccurlyeq \ell \preccurlyeq \ell_2\}$$

The induced optimal abstraction naturally uses meet to determine the lower bound, and the join for the upper bound:

Definition 13 (Interval Abstraction)**.**
Let $\alpha_\imath : \mathcal{P}(\textsc{Label}) \to \textsc{Label}^2$ be defined as follows:

$$\alpha_\imath(\emptyset) \textit{ is undefined}$$
$$\alpha_\imath(\{\overline{\ell_i}\}) = [\curlywedge \overline{\ell_i}, \curlyvee \overline{\ell_i}] \textit{ otherwise}$$

Armed with intervals, we represent label ordering evidence as a coordinate-wise abstraction of intervals. In particular, consistent label ordering predicate is justified by a *label evidence*, which is a pair of intervals $\langle \imath_1, \imath_2 \rangle$. For instance:

$$\langle [\mathsf{L}, \top], [\top, \top] \rangle \vdash \widetilde{\mathsf{L} \curlyvee ? \preccurlyeq \top \curlyvee ?} \qquad \langle [\mathsf{H}, \top], [\mathsf{H}, \top] \rangle \vdash \widetilde{\mathsf{H} \curlyvee ? \preccurlyeq ?}$$

4.3 Evolving Evidence

Following AGT, to establish transitivity of consistent label ordering and consistent subtyping, we define consistent transitivity operators, whose definitions appeal to abstract interpretation. Given a transitive binary type predicate[5] $P \subseteq \textsc{Type}^2$, and two consistent judgments $\varepsilon_{12} \vdash \widetilde{P}(\widetilde{T}_1, \widetilde{T}_2)$ and $\varepsilon_{23} \vdash \widetilde{P}(\widetilde{T}_2, \widetilde{T}_3)$, Garcia et al. (2016) define the consistent transitivity (partial) function $\circ^P$:

$$\begin{aligned}\langle \widetilde{T}_1, \widetilde{T}_{21} \rangle \circ^P \langle \widetilde{T}_{22}, \widetilde{T}_3 \rangle = \\ \alpha^2(\{\langle T_1, T_3 \rangle \in \gamma^2(\widetilde{T}_1, \widetilde{T}_3) \mid \\ \exists T_2 \in \gamma(\widetilde{T}_{21}) \cap \gamma(\widetilde{T}_{22}).\, P(T_1, T_2) \wedge P(T_2, T_3)\}).\end{aligned}$$

That is, consistent transitivity holds if there exists a static type in the intersection of the concretization of both middle gradual types of the evidences for which the static predicates that imply transitivity hold. Note that the resulting evidence is obtained by coordinate-wise abstraction α^2, which abstracts a set of pairs as a pair of abstractions.

Definition 14 (Consistent transitivity for label ordering)**.** *Suppose*

$$\langle \imath_{11}, \imath_{12} \rangle \vdash \widetilde{F_1(\overline{\ell}_i) \preccurlyeq F_2(\overline{\ell}_j)} \qquad \langle \imath_{21}, \imath_{22} \rangle \vdash \widetilde{F_2(\overline{\ell}_j) \preccurlyeq F_3(\overline{\ell}_k)}$$

We deduce evidence for consistent transitivity for label ordering:

$$\langle \imath_{11}, \imath_{12} \rangle \circ^{\preccurlyeq} \langle \imath_{21}, \imath_{22} \rangle \vdash \widetilde{F_1(\overline{\ell}_i) \preccurlyeq F_3(\overline{\ell}_k)}$$

where $\circ^{\preccurlyeq} : \textsc{Label}^2 \times \textsc{Label}^2 \to \textsc{Label}^2$ is defined as:

$$\begin{aligned}\langle \imath_{11}, \imath_{12} \rangle \circ^{\preccurlyeq} \langle \imath_{21}, \imath_{22} \rangle = \\ \alpha_i^2(\{\langle \ell_{11}, \ell_{22} \rangle \in \gamma_i^2(\langle \imath_{11}, \imath_{22} \rangle) \mid \\ \exists \ell \in \gamma_\imath(\imath_{12}) \cap \gamma_\imath(\imath_{21}).\ell_{11} \preccurlyeq \ell \wedge \ell \preccurlyeq \ell_{22}\})\end{aligned}$$

For instance, consider the following consistent judgments:

$$\langle [\mathsf{L}, \top], [\mathsf{H}, \top] \rangle \vdash \widetilde{\mathsf{L} \curlyvee ? \preccurlyeq \mathsf{H} \curlyvee ?} \qquad \langle [\mathsf{H}, \top], [\mathsf{H}, \top] \rangle \vdash \widetilde{\mathsf{H} \curlyvee ? \preccurlyeq ?}$$

We can combine both evidences to justify the consistent transitivity judgment:

$$\langle [\mathsf{L}, \top], [\mathsf{H}, \top] \rangle \circ^{\preccurlyeq} \langle [\mathsf{H}, \top], [\mathsf{H}, \top] \rangle = \langle [\mathsf{L}, \top], [\mathsf{H}, \top] \rangle \vdash \widetilde{\mathsf{L} \curlyvee ? \preccurlyeq ?}$$

The resulting evidence retains knowledge that transitivity involved the security label H. Had gradual labels been used in place of label intervals to define evidence, this information would have been lost, and noninterference subsequently violated.

Finally, to formalize the reduction of gradual programs we leverage the fact that consistent join distributes over evidence. First, we define join for evidence:

[5] Garcia et al. (2016) fail to note that the underlying predicate must be transitive for this definition to make sense.

Definition 15 (Evidence join)**.**

$$\langle \iota_1, \iota_2 \rangle \mathbin{\widetilde{\curlyvee}} \langle \iota_3, \iota_4 \rangle = \langle \iota_1 \mathbin{\widetilde{\curlyvee}} \iota_3, \iota_2 \mathbin{\widetilde{\curlyvee}} \iota_4 \rangle$$

$$\textit{where } [\ell_1, \ell_2] \mathbin{\widetilde{\curlyvee}} [\ell_3, \ell_4] = [\ell_1 \curlyvee \ell_3, \ell_2 \curlyvee \ell_4]$$

The evidence for two consistent label ordering judgments can be joined to justify a new consistent label ordering judgment that joins the left operands and right operands of the original two judgments.

Proposition 7.

If $\varepsilon_1 \vdash \widetilde{F_{11}(\overline{\tilde{\ell}_i}) \preccurlyeq F_{12}(\overline{\tilde{\ell}_j})}$ *and* $\varepsilon_2 \vdash \widetilde{F_{21}(\overline{\tilde{\ell}_i}) \preccurlyeq F_{22}(\overline{\tilde{\ell}_j})}$ *then* $\varepsilon_1 \mathbin{\widetilde{\curlyvee}} \varepsilon_2 \vdash \widetilde{F_{11}(\overline{\tilde{\ell}_i}) \mathbin{\widetilde{\curlyvee}} F_{21}(\overline{\tilde{\ell}_i}) \preccurlyeq F_{12}(\overline{\tilde{\ell}_j}) \mathbin{\widetilde{\curlyvee}} F_{22}(\overline{\tilde{\ell}_j})}$.

Evidence for typing judgments. We define evidence for consistent subtyping as types annotated with label intervals, and consistent transitivity of subtyping using the same techniques as for labels. We omit these definitions for space reasons (see Appendix C).

Algorithmic characterizations. The definitions of consistent transitivity that are follow from applying the AGT methodology involve concretization and abstraction, and do not immediately suggest an algorithm. Appendix C.4 includes such equivalent algorithmic characterizations.

4.4 Towards Reduction Rules: Intrinsic Terms

The runtime semantics of GSL$_{\mathsf{Ref}}$ follows the structure of the type safety proof of SSL$_{\mathsf{Ref}}$: progress determines which subderivation to reduce, and preservation determines which consistent deductions justify the structure of the reduced derivation. 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 the dynamic semantics: they play the same role that cast calculi play in typical presentations of gradual typing (Siek and Taha 2006).

Intrinsically-typed terms $t^{\widetilde{S}}$ comprise a family $\textsc{Term}_{\widetilde{S}}$ of type-indexed sets, such that ill-typed terms do not exist. They are built up from disjoint families $x^{\widetilde{S}} \in \textsc{Var}_{\widetilde{S}}$ and $l^{\widetilde{S}} \in \textsc{Loc}_{\widetilde{S}}$ of intrinsically typed variables and locations respectively.

For each typing rule, we define an intrinsic term formation rule that captures all the information needed to reconstruct the extrinsic typing rule. This way, an intrinsic term is isomorphic to a typing derivation. To understand the structure of intrinsic terms, recall the gradual typing judgment $\Gamma; \Sigma; \tilde{\ell} \vdash t : \widetilde{S}$. Intrinsic variables and locations reflect their typings, so intrinsic terms do not need explicit type environments Γ or store environments Σ. However, each typing judgment depends on a security effect $\tilde{\ell}$, which intrinsic terms must account for.

Additionally, because intrinsic terms represent typing derivations of programs *as they reduce*, this requires accounting for the possibility that runtime values have more precise types that 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 typecheck the program originally. The formation rule must permit this extra subtyping leeway, justified by evidence. This can be observed in the intrinsic term formation rule for applications (Iapp) on Figure 4. $\widetilde{S}_1$ is the runtime type of the function term. We annotate the initial static type information using the @ notation. Note that the evidence ε_ℓ for the label ordering premise is also annotated, since it is needed to reconstruct the derivation. The static PC label $\tilde{\ell}_c$ used to typecheck the source program is ascribed to the term (above the triangle), because it can evolve at runtime to a consistent sub-label $\tilde{\ell}_r$. The evidence of this additional premise ε_r is kept in the intrinsic term.

Figure 4 presents all the intrinsic terms formation rules for GSL$_{\mathsf{Ref}}$. Rules (Ix), (Ib), (Iu) and (Il) are straightforward. Rule (Iλ) uses the same static PC $\tilde{\ell}_c{}'$ for the body because functions are values. In rule (Iprot), labels $\tilde{\ell}$ and $\tilde{\ell}'$ represent the static and dynamic information of the label used to increase the PC in the subterm, respectively. Evidence ε justifies that the type of the subterm is a consistent subtype of $\widetilde{S}$, the static type of the subterm. Label $\tilde{\ell}'_c$ represents the static PC used to typecheck subterm $t^{\widetilde{S}'}$. Evidence ε'_r justifies that the new PC is a sublabel of $\tilde{\ell}'_c$. Rule (I$\oplus$) preserves the static labels join as an annotation, similar to the application rule (Iapp), described previously. The intrinsic term of a conditional, described in Rule (Iif), carries the static information of the label of the conditional term $\tilde{\ell}_1$. Each branch uses the join of the current PC and the actual label of the conditional term. As $\varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c$ and $\mathit{ilbl}(\varepsilon_1) \vdash \mathit{label}(\widetilde{S}_1) \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_1$, by monotonicity of the join, we can combine both evidences to justify that the dynamic new PC is a sublabel of the static one, *i.e.* $\widetilde{\tilde{\ell}_r \curlyvee \mathit{label}(\widetilde{S}_1) \preccurlyeq \tilde{\ell}_c \curlyvee \tilde{\ell}_1}$. We use this new evidence label to construct the intrinsic term of each conditional branch.[6] Evidences ε_2 and ε_3 justify that the type of each branch is a consistent subtype of the join of both types. Rule (I::) is straightforward; no annotation is necessary because the ascription itself carries the static type. Terms for references and dereferencing, (Iref) and (Ideref), carry the static type as annotation. Finally, rule (Iassgn) is built similarly to the application rule (Iapp).

4.5 Reduction

The evaluation rules directly mirror the rules for the statically typed language, except that they must manage evidence at subexpression borders and combine evidence to form new evidence. Whenever combining evidence using consistent transitivity fails, the program ends with an **error**.

The syntax of the intrinsic term language and its evaluation frames are presented in Figure 5. Following Garcia et al. (2016), the values of this language are either raw values or ascribed values. Evidence terms, evidence values and evidence labels correspond to entities augmented with evidence, as introduced in the intrinsic term formation rules.

Reduction takes an intrinsic term and a store, under a runtime PC $\tilde{\ell}_r$. Because the runtime PC is maintained at each step, we adopt a lightweight notation that captures the PC parts of intrinsic terms in the reduction arrows. Reduction therefore has the form $t_1^{\widetilde{S}} \mid \mu_1 \xmapsto{e\ell}_{\tilde{\ell}_c} t_2^{\widetilde{S}} \mid \mu_2$, corresponding to the reduction of the intrinsic term $e\ell \mathbin{\overset{\tilde{\ell}_c}{\triangleright}} t_1^{\widetilde{S}}$ in store μ_1 to the intrinsic term $e\ell \mathbin{\overset{\tilde{\ell}_c}{\triangleright}} t_2^{\widetilde{S}}$ in store μ_2. We note $\textsc{Conf}_{\widetilde{S}}$ the combination of a term $t^{\widetilde{S}}$ (without the PC part) and a store μ.

Reduction rules are presented in Figure 5. Reduction of intrinsic terms is specified using *term frames* f and *evidence frames* g. Rule (Rf) reduces under term frames. Rule (R$\longrightarrow$) reduces a term to either a term or an error using the notion of intrinsic reduction $\longrightarrow$ defined in Figure 6. Similarly Rule (Rg) steps the subterm with the evidence combination reduction $\longrightarrow_c$. Rule (Rprot) allows the protected subterm to step under a higher security level, which may be a sublabel of the one determined statically. Rules (Rprot) and (Rprot$g$) reduce the protected subterm. The propagation of errors is standard, omitted here for brevity (Appendix C.6–Figure 12).

The two notions of reduction $\longrightarrow$ and $\longrightarrow_c$ are defined in Figure 6. We first give a brief overview of each rule, and discuss more

[6] We use evidence inversion functions $\mathit{idom}, \mathit{icod}, \mathit{iref}, \mathit{ilbl}$ and ilat, which manifest the evidence for the inversion principles on consistent subtyping judgments. For instance, starting from the evidence that $\widetilde{S}_1 \lesssim \widetilde{S}_2$, ilbl produces the evidence of the judgment $\mathit{label}(\widetilde{S}_1) \mathrel{\widetilde{\preccurlyeq}} \mathit{label}(\widetilde{S}_2)$.

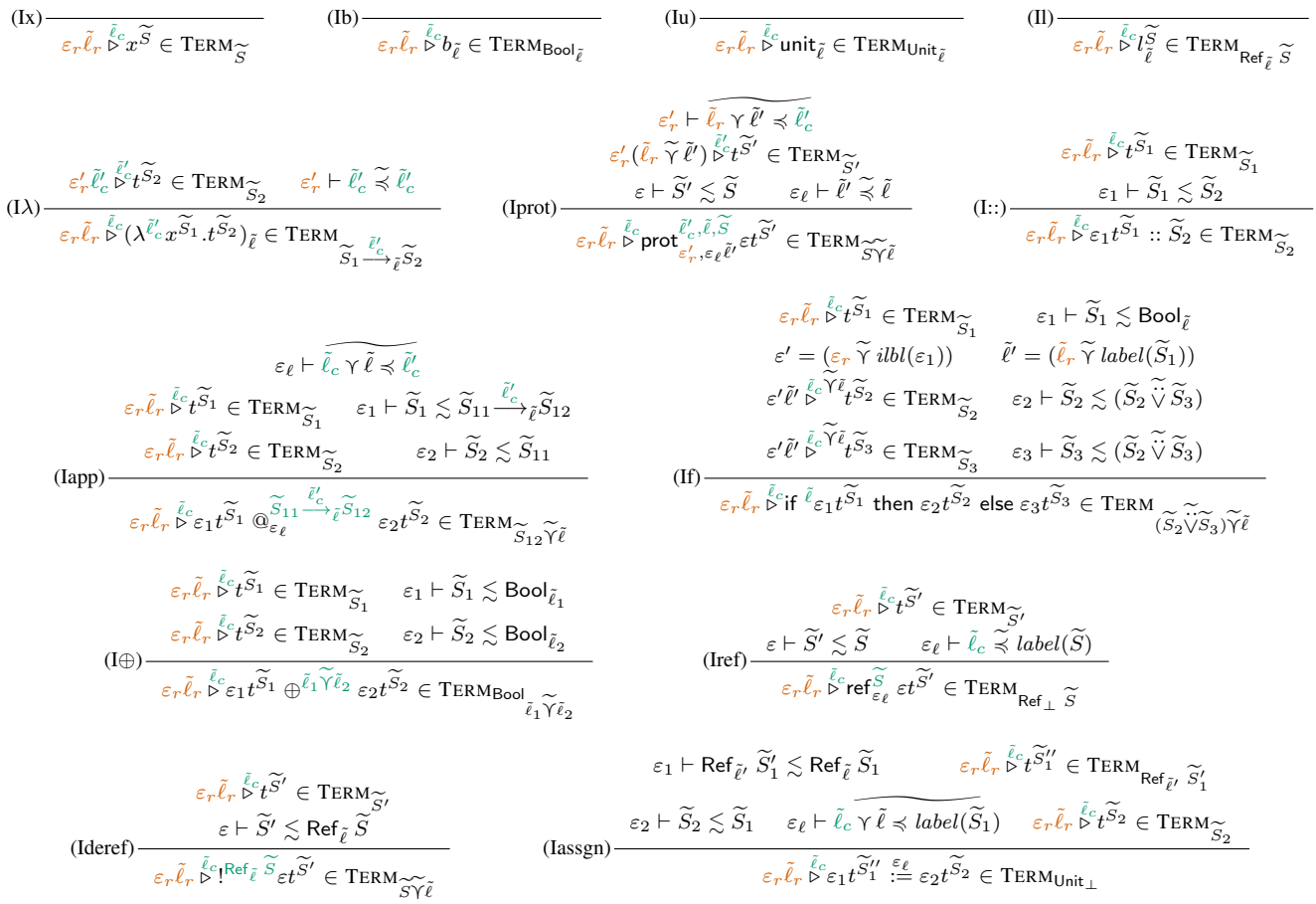

Figure 4. $\text{GSL}_{\mathsf{Ref}}$: Gradual Intrinsic Terms

details in the following paragraph. A Boolean operation is reduced by joining the evidence of both operands to justify the ascription. A protected value is reduced by stamping the actual security label of prot. Evidences are joined to justify the joined ascribed type. Function applications reduce to a protected body or to an error. The term reduces to an error if consistent transitivity does not hold to justify $\widetilde{S}_2 <: \widetilde{S}_{11}$. Conditionals similarly reduce by protecting the chosen branch expression. The new prot term is constructed using the static and dynamic information of the conditional term. References reduce to an ascribed fresh location. As stores need to preserve typing as well, the underlying value is also ascribed the statically-determined type $\widetilde{S}$. Also, as the current PC is stamped on the underlying value, the term may reduce to an error if consistent transitivity does not hold for judgment $\tilde{\ell}_r \preccurlyeq \mathit{label}(\widetilde{S})$. As previously mentioned, when dereferencing a store location, the actual security of the location is stamped on the resulting value. Then the new value must be ascribed the statically-determined type. We instead reduce to a protected expression, which is equivalent and simplifies the proofs.

Assignments may reduce to an ascribed unit value. Similarly to references, the stored value is ascribed the statically determined type $\widetilde{S}$. Therefore consistent transitivity may fail to justify that the actual type of the stored value is a subtype of $\widetilde{S}$. As the value is stamped with actual labels, the term may also reduce to an error if consistent transitivity does not hold for judgment $\tilde{\ell}_r \widetilde{\curlyvee} \ell \lesssim \widetilde{S}$.

4.6 Properties of the Gradual Security Language

Section 3.3 establishes several properties of the static semantics of $\text{GSL}_{\mathsf{Ref}}$. This section establishes two fundamental properties: type safety, and the dynamic gradual guarantee. Section 5 discusses noninterference.

The formulation of type safety relies on the notion of a store that is well-typed with respect to the $t^{\widetilde{S}}$ component of an intrinsic term $t^{\widetilde{S}} \vdash \mu$. This judgment corresponds to traditional store-typing, but appeals to the intrinsically typed labels in the given term.

Proposition 8 (Type Safety). *If $e\ell \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}} \in \text{TERM}_{\widetilde{S}}$ then one of the following is true:*

1. *$t^{\widetilde{S}}$ is a value v;*
2. *if $t^{\widetilde{S}} \vdash \mu$ then $t^{\widetilde{S}} \mid \mu \xrightarrow{e\ell}_{\tilde{\ell}_c} t'^{\widetilde{S}} \mid \mu'$ for some term $e\ell \overset{\tilde{\ell}_c}{\triangleright} t'^{\widetilde{S}} \in \text{TERM}_{\widetilde{S}}$ and some μ' such that $t'^{\widetilde{S}} \vdash \mu'$ and $\mathit{dom}(\mu) \subseteq \mathit{dom}(\mu')$;*
3. *$t^{\widetilde{S}} \mid \mu \xrightarrow{e\ell}_{\tilde{\ell}_c}$ **error**.*

$\varepsilon \in$ Evidence, $et \in$ EvTerm, $ev \in$ EvValue, $e\ell \in$ EvLabel, $\tilde{\ell}_c \in$ GLabel, $\tau \in$ Term$_*$, $v \in$ Value, $u \in$ SimpleValue, $g \in$ EvFrame, $f \in$ TmFrame

$$
\begin{array}{rcl}
\tau & ::= & e\ell \overset{\tilde{\ell}_c}{\triangleright} t \\
\varepsilon & ::= & \overline{\langle \widetilde{S}_i \rangle}^{\,n}_{m} \mid \overline{\langle \tilde{\ell}_i \rangle}^{\,n}_{m} \\
et & ::= & \varepsilon t \\
e\ell & ::= & \varepsilon \tilde{\ell} \\
ev & ::= & \varepsilon u \\
u & ::= & x^{\widetilde{S}} \mid b_{\tilde{\ell}} \mid (\lambda^{\tilde{\ell}_c} x^{\widetilde{S}}.t)_{\tilde{\ell}} \mid l^{\widetilde{S}}_{\tilde{\ell}} \mid \mathsf{unit}_{\tilde{\ell}} \\
v & ::= & u \mid \varepsilon u :: \widetilde{S} \\
g & ::= & \square \oplus^{\tilde{\ell}} et \mid ev \oplus^{\tilde{\ell}} \square \mid \square \,@^{\widetilde{S}}_{\varepsilon}\, et \mid ev \,@^{\widetilde{S}}_{\varepsilon}\, \square \mid \square :: \widetilde{S} \\
 & \mid & \mathsf{if}\ {}^{\tilde{\ell}}\square\ \mathsf{then}\ et\ \mathsf{else}\ et \mid !^{\widetilde{S}}\square \mid \square := \overset{\varepsilon}{} et \mid ev := \overset{\varepsilon}{} \square \\
 & \mid & \mathsf{ref}^{\widetilde{S}}_{\varepsilon}\, \square \\
f & ::= & g[\varepsilon\square] \\
\mu & := & \bullet \mid \mu, l^{\widetilde{S}} \mapsto v
\end{array}
$$

$\boxed{\overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} : \text{Conf}_{\widetilde{S}} \times (\text{Conf}_{\widetilde{S}} \cup \{\,\textbf{error}\,\})}$

$$(\mathrm{R}{\longrightarrow})\ \frac{t^{\widetilde{S}} \mid \mu \ \overset{e\ell}{\longrightarrow}_{\tilde{\ell}_c}\ r \qquad r \in (\text{Conf}_{\widetilde{S}} \cup \{\,\textbf{error}\,\})}{t^{\widetilde{S}} \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ r}$$

$$(\mathrm{R}f)\ \frac{t_1^{\widetilde{S}} \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ t_2^{\widetilde{S}} \mid \mu'}{f[t_1^{\widetilde{S}}] \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ f[t_2^{\widetilde{S}}] \mid \mu'}$$

$$(\mathrm{R}g)\ \frac{et \longrightarrow_c et'}{g[et] \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ g[et'] \mid \mu'}$$

$$(\mathrm{Rprot})\ \frac{t_1^{\widetilde{S}'} \mid \mu \ \overset{\varepsilon'_r(\tilde{\ell}_r \widetilde{\Upsilon} \tilde{\ell}')}{\longmapsto}_{\tilde{\ell}'_c}\ t_2^{\widetilde{S}'} \mid \mu'}{\mathsf{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell \tilde{\ell}'} \varepsilon t_1^{\widetilde{S}'} \mid \mu \ \overset{\varepsilon_r \tilde{\ell}_r}{\longmapsto}_{\tilde{\ell}_c}\ \mathsf{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell \tilde{\ell}'} \varepsilon t_2^{\widetilde{S}'} \mid \mu'}$$

$$(\mathrm{Rprotg})\ \frac{et \longrightarrow_c et'}{\mathsf{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell \tilde{\ell}'} et \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ \mathsf{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell \tilde{\ell}'} et' \mid \mu'}$$

Figure 5. GSL$_{\mathsf{Ref}}$: Intrinsic Reduction (error propagation rules omitted)

The dynamic gradual guarantee (Siek et al. 2015) states that any program that runs without errors would continue to do so if it were given a less precise type.

Proposition 9 (Dynamic guarantee)**.** *Suppose* $t_1^{\widetilde{S}_1} \sqsubseteq t_1^{\widetilde{S}_2}$, $e\ell_1 \sqsubseteq e\ell_2$, $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$, *and* $\mu_1 \sqsubseteq \mu_2$. *If* $t_1^{\widetilde{S}_1} \mid \mu_1 \ \overset{e\ell_1}{\longmapsto}_{\tilde{\ell}_{c1}}\ t_2^{\widetilde{S}_1} \mid \mu_1'$ *then* $t_1^{\widetilde{S}_2} \mid \mu_2 \ \overset{e\ell_2}{\longmapsto}_{\tilde{\ell}_{c2}}\ t_2^{\widetilde{S}_2} \mid \mu_2'$ *where* $t_2^{\widetilde{S}_1} \sqsubseteq t_2^{\widetilde{S}_2}$ *and* $\mu_1' \sqsubseteq \mu_2'$.

5. Noninterference

This section establishes noninterference for GSL$_{\mathsf{Ref}}$ using a logical relations-based approach (Heintze and Riecke 1998; Zdancewic 2002). Because general references introduce nontermination, however, we also apply step-indexing (Ahmed 2004).

Noninterference formalizes the intuition that low-security observers of a computation cannot detect changes in high-security inputs. Therefore noninterference inherently reflects a relationship between different runs of the same program with different inputs. We focus, on *termination-insensitive* noninterference, a standard

$\boxed{\overset{e\ell}{\longrightarrow}_{\tilde{\ell}_c} : \text{Conf}_{\widetilde{S}} \times (\text{Conf}_{\widetilde{S}} \cup \{\,\textbf{error}\,\})}$

$$\varepsilon_1(b_1)_{\tilde{\ell}_1} \oplus^{\tilde{\ell}} \varepsilon_2(b_2)_{\tilde{\ell}_2} \mid \mu \ \overset{e\ell}{\longrightarrow}_{\tilde{\ell}_c}\ (\varepsilon_1 \,\widetilde{\Upsilon}\, \varepsilon_2)(b_1 \,[\![\oplus]\!]\, b_2)_{(\tilde{\ell}_1 \widetilde{\Upsilon} \tilde{\ell}_2)} :: \mathsf{Bool}_{\tilde{\ell}} \mid \mu$$

$$\mathsf{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell \tilde{\ell}'} \varepsilon u \mid \mu \ \overset{e\ell}{\longrightarrow}_{\tilde{\ell}_c}\ (\varepsilon \,\widetilde{\Upsilon}\, \varepsilon_\ell)(u \,\widetilde{\Upsilon}\, \tilde{\ell}') :: \widetilde{S} \,\widetilde{\Upsilon}\, \tilde{\ell} \mid \mu$$

$$\varepsilon_1(\lambda^{\tilde{\ell}_{c1}} x^{\widetilde{S}_{11}}.t^*)_{\tilde{\ell}_1} \,@^{\widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c} \tilde{\ell} \widetilde{S}_2}_{\varepsilon_\ell}\, \varepsilon_2 u \mid \mu \ \overset{\varepsilon_r \tilde{\ell}_r}{\longrightarrow}_{\tilde{\ell}_c}$$

$$\begin{cases} \mathsf{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}_2}_{\varepsilon'_r,ilbl(\varepsilon_1)\tilde{\ell}_1} icod(\varepsilon_1)([(\varepsilon u :: \widetilde{S}_{11})/x^{\widetilde{S}_{11}}]t^*) \mid \mu \\ \textbf{error} \qquad \text{if } \varepsilon \text{ or } \varepsilon'_r \text{ are not defined} \end{cases}$$

$$\text{where} \quad \varepsilon = \varepsilon_2 \circ^{<:} idom(\varepsilon_1)$$
$$\varepsilon'_r = (\varepsilon_r \,\widetilde{\Upsilon}\, ilbl(\varepsilon_1)) \circ^{\preccurlyeq} \varepsilon_\ell \circ^{\preccurlyeq} ilat(\varepsilon_1)$$

$$\mathsf{if}\ {}^{\tilde{\ell}}\varepsilon_1 b_{\tilde{\ell}_1}\ \mathsf{then}\ \varepsilon_2 t^{\widetilde{S}_2}\ \mathsf{else}\ \varepsilon_3 t^{\widetilde{S}_3} \mid \mu \ \overset{\varepsilon_r \tilde{\ell}_r}{\longrightarrow}_{\tilde{\ell}_c}$$

$$\begin{cases} \mathsf{prot}^{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell},\tilde{\ell},(\widetilde{S}_2 \dot{\widetilde{\vee}} \widetilde{S}_3)}_{(\varepsilon_r \widetilde{\Upsilon} ilbl(\varepsilon_1)),ilbl(\varepsilon_1)\tilde{\ell}_1} \varepsilon_2 t^{\widetilde{S}_2} \mid \mu & b = \mathsf{true} \\ \mathsf{prot}^{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell},\tilde{\ell},(\widetilde{S}_2 \dot{\widetilde{\vee}} \widetilde{S}_3)}_{(\varepsilon_r \widetilde{\Upsilon} ilbl(\varepsilon_1)),ilbl(\varepsilon_1)\tilde{\ell}_1} \varepsilon_3 t^{\widetilde{S}_3} \mid \mu & b = \mathsf{false} \end{cases}$$

$$\mathsf{ref}^{\widetilde{S}}_{\varepsilon_\ell} \varepsilon u \mid \mu \ \overset{\varepsilon_r \tilde{\ell}_r}{\longrightarrow}_{\tilde{\ell}_c}$$

$$\begin{cases} l^{\widetilde{S}}_{\perp} \mid \mu[l^{\widetilde{S}} \mapsto \varepsilon'(u \,\widetilde{\Upsilon}\, \tilde{\ell}_r) :: \widetilde{S}] \text{ where } l^{\widetilde{S}} \notin dom(\mu) \\ \textbf{error} \qquad \text{if } (\varepsilon_r \circ^{\preccurlyeq} \varepsilon_\ell) \text{ is not defined} \end{cases}$$

$$\text{where } \varepsilon' = \varepsilon \,\widetilde{\Upsilon}\, (\varepsilon_r \circ^{\preccurlyeq} \varepsilon_\ell)$$

$$!^{\mathsf{Ref}_{\tilde{\ell}}\, \widetilde{S}} \varepsilon l^{\widetilde{S}'}_{\tilde{\ell}'} \mid \mu \ \overset{\varepsilon_r \tilde{\ell}_r}{\longrightarrow}_{\tilde{\ell}_c}\ \mathsf{prot}^{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell},\tilde{\ell},\widetilde{S}}_{(\varepsilon_r \widetilde{\Upsilon} ilbl(\varepsilon)),ilbl(\varepsilon)\tilde{\ell}'} iref(\varepsilon)v \text{ where } \mu(l^{\widetilde{S}'}) = v$$

$$\varepsilon_1 l^{\widetilde{S}}_{\tilde{\ell}} :=^{\varepsilon_\ell} \varepsilon_2 u \mid \mu \ \overset{\varepsilon_r \tilde{\ell}_r}{\longrightarrow}_{\tilde{\ell}_c}$$

$$\begin{cases} \mathsf{unit}_{\perp} \mid \mu[l^{\widetilde{S}} \mapsto \varepsilon'(u \,\widetilde{\Upsilon}\, (\tilde{\ell}_r \,\widetilde{\Upsilon}\, \tilde{\ell})) :: \widetilde{S}] \\ \textbf{error} \qquad \text{if } \varepsilon' \text{ is not defined} \end{cases}$$

$$\text{where } \varepsilon' = (iref(\varepsilon_1) \circ^{<:} \varepsilon_2) \,\widetilde{\Upsilon}\, ((\varepsilon_r \,\widetilde{\Upsilon}\, ilbl(\varepsilon_1)) \circ^{\preccurlyeq} \varepsilon_\ell \circ^{\preccurlyeq} ilbl(iref(\varepsilon_1))$$

$\boxed{\longrightarrow_c : \text{EvTerm} \times (\text{EvTerm} \cup \{\,\textbf{error}\,\})}$

$$\varepsilon_1(\varepsilon_2 v :: \widetilde{S}) \longrightarrow_c \begin{cases} (\varepsilon_2 \circ^{<:} \varepsilon_1)v \\ \textbf{error} \qquad \text{if not defined} \end{cases}$$

Figure 6. GSL$_{\mathsf{Ref}}$: Notions of Intrinsic Reduction

notion for information-flow security languages. The idea is that a low-security observer can monitor at least two channels of information about the final program result (a particular final value), as well as the termination channel (whether the program terminated or not). In this model, the termination channel is ignored: interference between two executions is only acknowledged when both terminate in values and those values are observably different. In line with prior work on gradual security, we consider runtime check errors to be termination-channel observations because in principle the semantics could respond by diverging and directly reporting the error through a secure channel.

Observing values. Noninterference hinges on notions of an observer, observable values, and the observations that she can make in any particular context. Consider, for instance, some value $\vdash v : \widetilde{S}_1 \to_{\tilde{\ell}} \widetilde{S}_2$, and a potential observer with confidentiality level ℓ_o. Two questions arise: can the observer observe the value? And if so, what observations can she make? The logical relations approach identifies the observer with the security label ℓ_o, to formalize the observer, and uses the type $\widetilde{S} = \widetilde{S}_1 \to_{\tilde{\ell}} \widetilde{S}_2$ to formalize the observations ℓ_o can make.

$$\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v_2, \mu_2\rangle : \mathsf{Bool}_{\tilde{\ell}} \iff \varepsilon_r\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i \in \text{TERM}_{\mathsf{Bool}_{\tilde{\ell}}} \wedge \langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle \wedge \mathsf{obs}_{\ell_o}(\varepsilon_{ri}\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i) \implies rval(v_1) = rval(v_2)$$

$$\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v_2, \mu_2\rangle : \mathsf{Unit}_{\tilde{\ell}} \iff \varepsilon_r\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i \in \text{TERM}_{\mathsf{Unit}_{\tilde{\ell}}} \wedge \langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle \wedge \mathsf{obs}_{\ell_o}(\varepsilon_{ri}\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i) \implies rval(v_1) = rval(v_2)$$

$$\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v_2, \mu_2\rangle : \widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_2 \iff$$
$$\varepsilon_r\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i \in \text{TERM}_{\widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_2} \wedge \langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle \wedge \mathsf{obs}_{\ell_o}(\varepsilon_{ri}\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i) \implies$$
$$\forall j \le k. \forall \widetilde{S}' = \widetilde{S}''_1 \xrightarrow{\tilde{\ell}''_c}_{\tilde{\ell}''} \widetilde{S}''_2, \widetilde{S}'_1, \varepsilon_1 \vdash \widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_2 \lesssim \widetilde{S}', \text{ and } \varepsilon_2 \vdash \widetilde{S}'_1 \lesssim \widetilde{S}''_1, \varepsilon_\ell \vdash \tilde{\ell}_c \curlyvee \tilde{\ell} \preccurlyeq \tilde{\ell}'_c, \text{ we have:}$$
$$\forall v'_i, \mu'_i, \langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v'_1, \mu'_1\rangle \approx^j_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v'_2, \mu_2\rangle : \widetilde{S}'_1, dom(\mu_i) \subseteq dom(\mu'_i),$$
$$\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} (\varepsilon_1 v_1 \mathbin{@}^{\widetilde{S}'}_{\varepsilon_\ell} \varepsilon_2 v'_1), \mu'_1\rangle \approx^j_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} (\varepsilon_1 v_2 \mathbin{@}^{\widetilde{S}'}_{\varepsilon_\ell} \varepsilon_2 v'_2), \mu'_2\rangle : \mathcal{C}(\widetilde{S}''_2 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}')$$

$$\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v_2, \mu_2\rangle : \mathsf{Ref}_{\tilde{\ell}}\, \widetilde{S} \iff \varepsilon_r\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i \in \text{TERM}_{\mathsf{Ref}_{\tilde{\ell}}\, \widetilde{S}} \wedge \langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle \wedge \mathsf{obs}_{\ell_o}(\varepsilon_{ri}\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i) \implies rval(v_1) = rval(v_2)$$

$$\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} t_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} t_2, \mu_2\rangle : \mathcal{C}(\widetilde{S}) \iff \varepsilon_r\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} t_i \in \text{TERM}_{\widetilde{S}} \wedge \langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle \wedge$$
$$(t_i \mid \mu_i \overset{\varepsilon_{ri}\tilde{\ell}_{ri}}{\longmapsto}{}^{j}_{\tilde{\ell}_c} t'_i \mid \mu'_i \wedge \langle \tilde{\ell}_{r1}, \mu'_1\rangle \approx^{k-j}_{\ell_o} \langle \tilde{\ell}_{r2}, \mu'_2\rangle \wedge dom(\mu_i) \subseteq dom(\mu'_i) \wedge irred(t'_i)$$
$$\implies \langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} t'_1, \mu'_1\rangle \approx^{k-j}_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} t'_2, \mu'_2\rangle : \widetilde{S})$$

$$\mathsf{obs}_{\ell_o}(\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v) \iff \varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v \in \text{TERM}_{\widetilde{S}} \wedge \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \ell_o \wedge \big((v = \varepsilon u :: \widetilde{S}) \implies ilbl(\varepsilon) \circ^{\preccurlyeq} \varepsilon_2 \text{ is defined}\big)$$
$$\text{where} \quad \varepsilon_2 = \mathcal{I}_{\preccurlyeq}(label(\widetilde{S}), \ell_o)$$

Figure 7. Gradual security logical relations

First, $\tilde{\ell}_0$ cannot make *any* observations if her confidentiality level does not subsume that of the function, *i.e.* $\ell_o \mathrel{\widetilde{\not\preccurlyeq}} \ell$. If clearance is granted, on the other hand, *i.e.* $\ell_o \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}$, then ℓ_o may make observations in accordance with the structure of the type: she may construct another value $v' : \widetilde{S}_1$, apply it to the function, and receive its results. What observations, if any, can ℓ_o make of the result? They are dictated by the type $\widetilde{S}_2$: the cycle begins anew. In short, the security type of a value dictates both an observation protocol and the clearance required to apply it.

We can formalize the initial clearance requirement using an obs_{ℓ_o} predicate that captures what it means for v to be observable at ℓ_o. The following definition reflects the standard conception:

$$\mathsf{obs}_{\ell_o}(v) \iff v \in \text{TERM}_{\widetilde{S}} \wedge label(\widetilde{S}) \mathrel{\widetilde{\preccurlyeq}} \ell_o$$

We extend this notion in two ways.

First, observation becomes somewhat delicate in the gradual checking context. Consider, for instance, a value $\vdash v : \mathsf{Bool}_{\tilde{\ell}}$. The standard observation for such atomic types is to directly observe their identity: either $v = \mathsf{true}$ or $v = \mathsf{false}$. Then one would expect the protocol of the security type $\mathsf{Bool}_{\tilde{\ell}}$ to be "check credentials, then check identity." But what if $v = \varepsilon\mathsf{true}_{\mathsf{H}} :: \mathsf{Bool}_?$? An observer at security level L must *not* observe the underlying high-security value. The key intuition is that her observation should ultimately be equivalent to applying the source language context if $\square :: \mathsf{Bool}_{\mathsf{L}}$ then $\mathsf{true}_{\mathsf{L}}$ else $\mathsf{false}_{\mathsf{L}}$ to the value, thereby asserting credentials and then using them. Doing so would trigger a runtime check error, which amounts to a non-observation.

Second, observation must account for the dynamic security context of the term being executed, $\tilde{\ell}_r$. In essence, observation leaks a value from its context, so we must ensure that the observer has the proper credentials. The full definition of obs_{ℓ_o} (Figure 7) accounts for the dynamic security context clearance $\tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \ell_o$, and checks before accessing values. To address the latter, the expression $\square \circ^{\preccurlyeq} \varepsilon_2$ corresponds to a credential-granting ascription $\square :: \mathsf{Bool}_{\mathsf{L}}$.

Relating values. We define a number of concepts to relate computations to one another (Figure 7). The remainder can be found in Appendix D. The core notion is the ternary relation:

$$\langle \tau_1, \mu_1\rangle \approx^k_{\ell_o} \langle \tau_2, \mu_2\rangle : \widetilde{S}$$

where $\tau_i = \varepsilon_{ri}\ell_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i$ are two intrinsically-typed values.

In essence, the judgment says that from the vantage point of observer ℓ_o, the two value-store configurations $\langle \tau_i, \mu_i\rangle$ are equivalent, where observation is restricted by the actions allowed by simple type structure of $\widetilde{S}$, assuming the proper credentials as imposed by the security labels in $\widetilde{S}$, and finally given only k steps of reduction to work with. We explain these components in more detail.

First, the relation is only defined for pairs of intrinsic terms that have the type $\widetilde{S}$, so type safety ensures that the behaviors dictated by $\widetilde{S}$ will produce defined behavior (including runtime error). This corresponds to the guarantees of simple type safety without security. Second, the security labels embedded in $\widetilde{S}$ determine at each step whether ℓ_o has the credentials to continue observing the pair of terms. If not, then the two terms are equivalent from ℓ_o's vantage.

Finally, a combination of general references and higher-order functions admits nontermination into the language. Thus, to make the relation well-defined, we index it on the number of steps k that the ℓ_o may take. If no inequivalent observations are made after k steps, then the terms are deemed equivalent. Ultimately we require that ℓ_o observes equivalence for any arbitrary number of steps, which implies that nonterminating computations also respect the noninterference guarantees. This is the essence of step-indexing.

This judgment relies on, and is mutually defined with, a variety of related equivalence relations for locations ℓ, stores μ, arbitrary intrinsic terms τ, and substitutions σ. These constructions culminate in a semantics-driven notion of noninterference and a proof that intrinsically-typed terms are sound with respect to it.

Definition 16 (Semantic Security Typing)**.** *For any* $\ell_o \in \text{LABEL}$, $k \geq 0$, $\sigma_1, \sigma_2 \in \text{SUBST}$, *and* $\mu_1, \mu_2 \in \text{STORE}$*:*
$\models \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}} : \widetilde{S}$ *if and only if:*

1. $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}} \in \text{TERM}_{\widetilde{S}}$,
2. $t^{\widetilde{S}} \vdash \mu_1$ *and* $t^{\widetilde{S}} \vdash \mu_2$,
3. $\Gamma = FV(t^{\widetilde{S}})$*, and*
4. $\Gamma \vdash \langle \varepsilon_r \tilde{\ell}_r, \tilde{\ell}_c, \sigma_1, \mu_1 \rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r, \tilde{\ell}_c, \sigma_2, \mu_2 \rangle$ *altogether imply* $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_1(t^{\widetilde{S}}), \mu_1 \rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_2(t^{\widetilde{S}}), \mu_2 \rangle : \mathcal{C}(\widetilde{S})$,

Proposition 10 (Security Type Soundness)**.**
If $\tau \in \text{TERM}_{\widetilde{S}}$*, then* $\models \tau : \widetilde{S}$.

The general intuition behind the semantic typing judgment and type soundness theorem are as follows: The intrinsic term $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}}$ may have free variables, which are intrinsically associated with specific types. In essence, those variables indicate "input parameters" to the term. The two substitutions σ_1 and σ_2 map each variable in the term to particular *related* closed intrinsic values. The analogous relationship holds for the (intrinsically typed) locations in the term and the two related stores μ_1 and μ_2.

Semantic soundness then says that any suitable and related substitutions and stores, when combined with the single term, produce related program configurations at the type $\widetilde{S}$, for any number of steps k, and for any observer ℓ_o. The key property to keep in mind is that low-security observers equate *all* high-security values, so if ℓ_o is, say $\bot$, and any of the term variables or locations has higher-security, then the corresponding substitution and store elements can be wildly different, up to the structures that the logical relation imposes on their types. Tying this all together, the judgment $\models \tau : \widetilde{S}$ says that the term τ respects the security protocol $\widetilde{S}$ for all observers, substitutions, stores, and steps (Ahmed 2004).

Finally, Security Type Soundness says that, so long as you regard semantic soundness as a faithful characterization of noninterference, then the syntactic type system enforces noninterference. The complete definition of the logical relations is given in Appendix D.1–Figure 14, and proofs are in Appendix D.2.

6. Discussion

This work sheds light on the gradual typing programme and on type-based reasoning more broadly. The proposed language designs and their metatheory make a strict distinction between type safety and type soundness. For instance, without the evidence improvements of Sec. 4.2, GSL_{Ref} was already safe—programs did not get stuck—but was not sound—syntactic typing did not enforce the stated information-flow security abstractions. We found this separation insightful: pondering the security logical relation and its overt, operational characterization of security, ultimately informed finer points of SSL_{Ref}'s design. As such, logical relations were not merely a proof technique, but also a design aide.

The GSL_{Ref} language points to some challenges facing language designs that treat unknown information as the top of an existing static subtyping or other ordering relation as per the quasi-static typing (Thatte 1990). Treating gradual types as a lifting of static typing yields a language that avoids the need for user-supplied casts and satisfies compelling criteria for gradual languages, which depend on notions like precision that AGT provides as a matter of course.

Finally, the given language designs challenge and exercise the AGT framework. Garcia et al. (2016) mention the option of choosing alternative abstractions for evidence in a brief footnote. This work affirms the necessity and utility of this control. Our decision to introduce merely an unknown label ? and its meaning necessitated a more precise form of evidence. However, an alternate design for the gradual language could introduce label intervals up-front as the notion of gradual labels and yield a successful design. This extra precision in the source language, however, must be weighed against the additional notational and conceptual complexity of interval labels and types. The GSL_{Ref} source language hides this complexity from the programmer. Such is the richness of the design space for gradual languages.

7. Related Work

As discussed in Section 2, the SSL_{Ref} language design was informed by a number of prior security type system designs.

The GSL_{Ref} language is most closely related to the work of Fennell and Thiemann (2013). In addition to developing a language that combines static and dynamic checking of security, they develop a notion of blame tracking and prove a blame theorem for their semantics. Like our system, they support dynamically changing the security type of a reference cell.

The design of a gradual security-typed language is a novel contribution. Despite the fact that both Disney and Flanagan (2011) and Fennell and Thiemann (2013) have proposed languages for security typing dubbed gradual, they do not propose gradual source languages. Rather, the language designs require explicit security casts—which can also be encoded with a label test expression in Jif (Zheng and Myers 2007). Furthermore, both designs treat an unlabeled type as having the top label, then allowing explicit casts downward in the security lattice. This design is analogous to the internal language of the quasi-static typing approach. In that approach, explicit casts work well, but the external language accepts too many programs. That difficulty was the original motivation for consistency in gradual typing (Siek and Taha 2006).

Thiemann and Fennell (2014) develop a generic approach to gradualize annotated type systems. This is similar to security typing (labels are one kind of annotation), except that they focus on annotations on base types, and the language relies on explicit casts, like the gradual security work discussed above. In addition to gradualization, they also track blame and provide a translation that removes unnecessary casts: the present work does neither.

A variety of *hybrid* information-flow control systems have been investigated, *e.g.* (Buiras et al. 2015; Chandra and Franz 2007; Shroff et al. 2007). These designs combine static and dynamic techniques that buttress one another. Gradual security checking, in contrast, combines static and dynamic checking with the express intent of supporting smooth migration. Russo and Sabelfeld (2010) develops an information-flow system that supports *deferring* some checks to runtime, in the spirit of hybrid typing (Knowles and Flanagan 2010).

8. Conclusion

This paper presents two intimately intertwined language designs, one purely static, the other gradual. Together, the languages expose a smooth continuum between two typing disciplines, in both their static and dynamic semantics. This first gradual security language presents a compelling case study for the AGT approach. Future work includes scaling the security discipline to more advanced features, like declassification (Sabelfeld and Sands 2009). Ultimately this work views gradual typing as a theory of *imprecise typing* rather than dynamic checking, though the latter is an inevitable consequence. We hope that this broader view of gradual typing can expand its its appeal to include static and dynamic language enthusiasts alike. Ultimately, we find that investigating the span between static and dynamic checking disciplines allows each to enrich the other, both materially and conceptually.

References

A. Ahmed. *Semantics of Types for Mutable State*. PhD thesis, Princeton University, Nov. 2004.

F. Bañados Schwerter, R. Garcia, and É. Tanter. A theory of gradual effect systems. In *19th ACM SIGPLAN Conference on Functional Programming (ICFP 2014)*, pages 283–295, Gothenburg, Sweden, Sept. 2014. ACM Press.

P. Buiras, D. Vytiniotis, and A. Russo. HLIO: mixing static and dynamic typing for information-flow control in Haskell. In *20th ACM SIGPLAN Conference on Functional Programming (ICFP 2015)*, pages 289–301, Vancouver, Canada, Sept. 2015. ACM Press.

D. Chandra and M. Franz. Fine-grained information flow analysis and enforcement in a java virtual machine. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 463–475, Dec 2007.

A. Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5(2):56–68, 06 1940.

P. Cousot and R. Cousot. Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages), invited paper. In *1994 International Conference on Computer Languages*, pages 95–112, Toulouse, France, 16–19 May 1994. IEEE Computer Society Press, Los Alamitos, California.

D. E. Denning. A lattice model of secure information flow. *Commun. ACM*, 19(5):236–243, May 1976. ISSN 0001-0782.

T. Disney and C. Flanagan. Gradual information flow typing. In *International Workshop on Scripts to Programs*, 2011.

L. Fennell and P. Thiemann. Gradual security typing with references. In *Computer Security Foundations Symposium*, pages 224–239, June 2013.

R. Garcia, A. M. Clark, and É. Tanter. Abstracting gradual typing. In *43rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2016)*, St Petersburg, FL, USA, Jan. 2016. ACM Press.

D. K. Gifford and J. M. Lucassen. Integrating functional and imperative programming. In *Proceedings of the 1986 ACM Conference on Lisp and Functional Programming*, pages 28–38, Cambridge, MA, USA, Aug. 1986. ACM Press.

N. Heintze and J. G. Riecke. The SLam Calculus: Programming with secrecy and integrity. In *25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 1998)*, pages 365–377, New York, NY, USA, 1998. ACM. ISBN 0-89791-979-3.

W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press, New York, 1980. Reprint of 1969 article.

K. Knowles and C. Flanagan. Hybrid type checking. *ACM Trans. Program. Lang. Syst.*, 32(2), 2010.

F. Pottier and V. Simonet. Information flow inference for ml. *ACM Transactions on Programming Languages and Systems*, 25(1):117–158, Jan. 2003. ISSN 0164-0925.

A. Russo and A. Sabelfeld. Dynamic vs. static flow-sensitive security analysis. In *Proceedings of the 2010 23rd IEEE Computer Security Foundations Symposium*, CSF '10, pages 186–199, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4082-5.

A. Sabelfeld and D. Sands. Declassification: Dimensions and principles. *Journal of Computer Security*, 17(5):517–548, 2009.

P. Shroff, S. Smith, and M. Thober. Dynamic dependency monitoring to secure information flow. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium*, CSF '07, pages 203–217, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2819-8.

J. G. Siek and W. Taha. Gradual typing for functional languages. In *Scheme and Functional Programming Workshop*, pages 81–92, Sept. 2006.

J. G. Siek, M. M. Vitousek, M. Cimini, and J. T. Boyland. Refined criteria for gradual typing. In *1st Summit on Advances in Programming Languages (SNAPL 2015)*, pages 274–293, 2015.

S. Thatte. Quasi-static typing. In *17th ACM Symposium on Principles of Programming Languages (POPL 90)*, pages 367–381, San Francisco, CA, United States, Jan. 1990. ACM Press.

P. Thiemann and L. Fennell. Gradual typing for annotated type systems. In Z. Shao, editor, *23rd European Symposium on Programming Languages and Systems (ESOP 2014)*, volume 8410 of *LNCS*, pages 47–66, Grenoble, France, 2014. Springer-Verlag.

D. Volpano, C. Irvine, and G. Smith. A sound type system for secure flow analysis. *J. Comput. Secur.*, 4(2-3):167–187, Jan. 1996. ISSN 0926-227X.

A. K. Wright and M. Felleisen. A syntactic approach to type soundness. *Journal of Information and Computation*, 115(1):38–94, Nov. 1994.

S. Zdancewic. *Programming Languages for Information Security*. PhD thesis, Cornell University, Aug. 2002.

L. Zheng and A. C. Myers. Dynamic security labels and noninterference. *International Journal of Information Security*, 6(2):67–84, Mar. 2007.

A. Static Security Typing with References

In this section we present some auxiliary definitions (section A.1), and the proof of type preservation for SSL_{Ref} (section A.2.

A.1 Auxiliary Definitions

We define join between types and labels as follows

$$\begin{gathered}\mathsf{Bool}_\ell \curlyvee \ell' = \mathsf{Bool}_{(\ell \curlyvee \ell')} \\ (S_1 \xrightarrow{\ell_c}_\ell S_2) \curlyvee \ell' = S_1 \xrightarrow{\ell_c}_{(\ell \curlyvee \ell')} S_2 \\ \mathsf{Ref}_\ell\ S \curlyvee \ell' = \mathsf{Ref}_{(\ell \curlyvee \ell')}\ S\end{gathered}$$

Figure 8 presents the join and meet type functions.

$\boxed{S \mathbin{\ddot{\vee}} S, S \mathbin{\ddot{\wedge}} S}$

$\ddot{\vee} : \textsc{Type} \times \textsc{Type} \rightharpoonup \textsc{Type}$

$\mathsf{Bool}_\ell \mathbin{\ddot{\vee}} \mathsf{Bool}_{\ell'} = \mathsf{Bool}_{(\ell \curlyvee \ell')}$

$(S_{11} \xrightarrow{\ell_c}_\ell S_{12}) \mathbin{\ddot{\vee}} (S_{21} \xrightarrow{\ell'_c}_{\ell'} S_{22}) = (S_{11} \mathbin{\ddot{\wedge}} S_{21}) \xrightarrow{\ell_c \curlywedge \ell'_c}_{(\ell \curlyvee \ell')} (S_{12} \mathbin{\ddot{\vee}} S_{22})$

$\mathsf{Ref}_\ell\ S \mathbin{\ddot{\vee}} \mathsf{Ref}_{\ell'}\ S = \mathsf{Ref}_{(\ell \curlyvee \ell')}\ S$

$S \mathbin{\ddot{\vee}} S$ undefined otherwise

$\ddot{\wedge} : \textsc{Type} \times \textsc{Type} \rightharpoonup \textsc{Type}$

$\mathsf{Bool}_\ell \mathbin{\ddot{\wedge}} \mathsf{Bool}_{\ell'} = \mathsf{Bool}_{(\ell \curlywedge \ell')}$

$(S_{11} \xrightarrow{\ell_c}_\ell S_{12}) \mathbin{\ddot{\wedge}} (S_{21} \xrightarrow{\ell'_c}_{\ell'} S_{22}) = (S_{11} \mathbin{\ddot{\vee}} S_{21}) \xrightarrow{\ell_c \curlyvee \ell'_c}_{(\ell \curlywedge \ell')} (S_{12} \mathbin{\ddot{\wedge}} S_{22})$

$\mathsf{Ref}_\ell\ S \mathbin{\ddot{\wedge}} \mathsf{Ref}_{\ell'}\ S = \mathsf{Ref}_{(\ell \curlywedge \ell')}\ S$

$S \mathbin{\ddot{\wedge}} S$ undefined otherwise

Figure 8. SSL_{Ref}: Join and meet type functions

Definition 17 (Valid Type Sets)**.**

$$\frac{}{valid(\{\,\overline{\mathsf{Bool}_{\ell_i}}\,\})} \qquad \frac{valid(\{\,\overline{S_{i1}}\,\}) \quad valid(\{\,\overline{S_{i2}}\,\})}{valid(\{\,\overline{S_{i1} \xrightarrow{\ell_{ci}}_{\ell_i} S_{i2}}\,\})}$$

$$\frac{valid(\{\,\overline{S_i}\,\})}{valid(\{\,\overline{\mathsf{Ref}_{\ell_i}\ S_i}\,\})} \qquad \frac{}{valid(\{\,\overline{\mathsf{Unit}_{\ell_i}}\,\})}$$

Definition 18 (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 $\Gamma; \Sigma \vdash \mu$, if $dom(\mu) = dom(\Sigma)$ and $\forall l \in dom(\mu)$, $\Gamma; \Sigma; \bot \vdash \mu(l) : S$ and $S <: \Sigma(l)$.*

A.2 Type Preservation

Lemma 11. *If $\Gamma; \Sigma; \ell_c \vdash t : S$ then $\forall \ell'_c \preccurlyeq \ell_c, \Gamma; \Sigma; \ell'_c \vdash t : S'$ and $S' <: S$.*

Proof. By induction on the derivation of $\Gamma; \Sigma; \ell_c \vdash t : S$.

Case (Sx, Sb, Su, Sl). Trivial because neither the premises and the infered type depend on the PC.

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

$$(\text{S}\oplus)\frac{(\text{Sb})\ \dfrac{}{\Gamma; \Sigma; \ell_c \vdash b_{1\ell_1} : \mathsf{Bool}_{\ell_1}} \quad (\text{Sb})\ \dfrac{}{\Gamma; \Sigma; \ell_c \vdash b_{2\ell_2} : \mathsf{Bool}_{\ell_2}}}{\Gamma; \Sigma; \ell_c \vdash b_{1\ell_1} \oplus b_{2\ell_2} : \mathsf{Bool}_{(\ell_1 \curlyvee \ell_2)}}$$

Suppose ℓ'_c such that $\ell'_c \preccurlyeq \ell_c$, then by induction hypotheses on the premises:

$$(\text{S}\oplus)\frac{(\text{Sb})\ \dfrac{}{\Gamma; \Sigma; \ell'_c \vdash b_{1\ell_1} : \mathsf{Bool}_{\ell'_1}} \quad (\text{Sb})\ \dfrac{}{\Gamma; \Sigma; \ell'_c \vdash b_{2\ell_2} : \mathsf{Bool}_{\ell'_2}}}{\Gamma; \Sigma; \ell'_c \vdash b_{1\ell_1} \oplus b_{2\ell_2} : \mathsf{Bool}_{(\ell'_1 \curlyvee \ell'_2)}}$$

where $\ell'_1 = \ell_1$ and $\ell'_2 = \ell_2$ and the result holds.

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

$$(\text{Sprot})\frac{\dfrac{}{\Gamma; \Sigma; \ell_c \curlyvee \ell \vdash v : S}}{\Gamma; \Sigma; \ell_c \vdash \mathsf{prot}_\ell v : S \curlyvee \ell}$$

Suppose ℓ'_c such that $\ell'_c \preccurlyeq \ell_c$. Considering that $\ell'_c \curlyvee \ell \preccurlyeq \ell_c \ell \preccurlyeq$, then by induction hypotheses on the premise and Lemma 13:

$$(\text{Sprot})\frac{\dfrac{}{\Gamma; \Sigma; \ell'_c \curlyvee \ell \vdash v : S}}{\Gamma; \Sigma; \ell'_c \vdash \mathsf{prot}_\ell v : S \curlyvee \ell}$$

and therefore the result holds.

Case (Sapp). Then $t = t_1\ t_2$ and

$$(\text{Sapp})\frac{(\text{S}\lambda)\ \dfrac{\mathcal{D}_1}{\Gamma; \Sigma; \ell_c \vdash t_1 : S_{11} \xrightarrow{\ell'_c}_\ell S_{12}} \quad \dfrac{\mathcal{D}_2}{\Gamma; \Sigma; \ell_c \vdash t_2 : S_2} \quad \ell_c \curlyvee \ell \preccurlyeq \ell'_c \quad S_2 <: S_{11}}{\Gamma; \Sigma; \ell_c \vdash t_1\ t_2 : S_{12} \curlyvee \ell}$$

Suppose ℓ'_c such that $\ell'_c \preccurlyeq \ell_c$. Then by using induction hypotheses on the premisess, considering $S'_{11} \xrightarrow{\ell''_c}_{\ell} S'_{12} <: S_{11} \xrightarrow{\ell'_c}_{\ell} S_{12}$ and $S'_2 <: S_2$. As $S_2 <: S_{11}$ and $S_{11} <: S'_{11}$ then $S'_2 <: S'_{11}$. Also, by definition of the join operator $\ell'_c \curlyvee \ell \preccurlyeq \ell_c \curlyvee \ell \preccurlyeq \ell'_c$, and then:

$$(\text{Sapp})\frac{(\text{S}\lambda)\ \dfrac{\mathcal{D}_1}{\Gamma; \Sigma; \ell'_c \vdash t_1 : S'_{11} \xrightarrow{\ell''_c}_{\ell'} S'_{12}} \quad \dfrac{\mathcal{D}_2}{\Gamma; \Sigma; \ell'_c \vdash t_2 : S'_2} \quad \ell'_c \curlyvee \ell' \preccurlyeq \ell''_c \quad S'_2 <: S'_{11}}{\Gamma; \Sigma; \ell'_c \vdash t_1\ t_2 : S'_{12} \curlyvee \ell'}$$

Where $S'_{12} \curlyvee \ell' <: S_{12} \curlyvee \ell$ and the result holds.

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

$$(\text{Sif})\frac{\dfrac{\mathcal{D}_0}{\Gamma; \Sigma; \ell_c \vdash \mathsf{true}_\ell : \mathsf{Bool}_\ell} \quad \dfrac{\mathcal{D}_1}{\Gamma; \Sigma; \ell_c \curlyvee \ell \vdash t_1 : S_1} \quad \dfrac{\mathcal{D}_2}{\Gamma; \Sigma; \ell_c \curlyvee \ell \vdash t_2 : S_2}}{\Gamma; \Sigma; \ell_c \vdash \mathsf{if}\ \mathsf{true}_\ell\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 : (S_1 \mathbin{\ddot{\vee}} S_2) \curlyvee \ell}$$

Suppose ℓ'_c such that $\ell'_c \preccurlyeq \ell_c$, and ℓ' such that $\ell' <: \ell$. As $\ell'_c \curlyvee \ell' \preccurlyeq \ell_c \curlyvee \ell$, by induction hypotheses in the premises:

$$(\text{Sif})\frac{\dfrac{\mathcal{D}_0}{\Gamma; \Sigma; \ell'_c \vdash \mathsf{true}_{\ell'} : \mathsf{Bool}_{\ell'}} \quad \dfrac{\mathcal{D}_1}{\Gamma; \Sigma; \ell'_c \curlyvee \ell' \vdash t_1 : S'_1} \quad \dfrac{\mathcal{D}_2}{\Gamma; \Sigma; \ell'_c \curlyvee \ell' \vdash t_2 : S'_2}}{\Gamma; \Sigma; \ell'_c \vdash \mathsf{if}\ \mathsf{true}_{\ell'}\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 : (S'_1 \mathbin{\ddot{\vee}} S'_2) \curlyvee \ell'}$$

where $S'_1 <: S_1$, $S'_2 <: S_2$. Then $(S'_1 \mathbin{\ddot{\vee}} S'_2) \curlyvee \ell' <: (S_1 \mathbin{\ddot{\vee}} S_2) \curlyvee \ell$ and therefore the result holds.

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

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

$$(\text{Sref})\frac{\Gamma; \Sigma; \ell_c \vdash v : S' \quad S' <: S \quad \ell_c \preccurlyeq \mathsf{label}(S)}{\Gamma; \Sigma; \ell_c \vdash \mathsf{ref}^S\ v : \mathsf{Ref}_\bot\ S}$$

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

$$(\text{Sref})\frac{\Gamma; \Sigma; \ell'_c \vdash v : S'' \quad S'' <: S \quad \ell'_c \preccurlyeq \mathsf{label}(S)}{\Gamma; \Sigma; \ell'_c \vdash \mathsf{ref}^S\ v : \mathsf{Ref}_\bot\ S}$$

and the result holds

$$\mathsf{ref}^S\ v \mid \mu \xrightarrow{\ell_r} l_\perp \mid \mu[l \mapsto v \curlyvee \ell_r]$$

where $l \notin dom(\mu)$.
Let us take $\Sigma' = \Sigma, l : S$ and let us call $\mu' = \mu[l \mapsto v \curlyvee \ell_r]$. Then as $dom(\mu) = dom(\Sigma)$ then $dom(\mu') = dom(\Sigma')$. Also, as $\ell_r \preccurlyeq \ell_c \preccurlyeq label(S)$ then by Lemma 13, $\Gamma; \Sigma'; \bot \vdash v : S' \curlyvee \ell_r$ and $S' \curlyvee \ell_r <: \Sigma(l) = S$. Therefore $\Gamma; \Sigma' \vdash \mu'$.

Then

$$\text{(Sl)}\frac{l : S \in \Sigma'}{\Gamma; \Sigma'; \ell_c \vdash l_\perp : \mathsf{Ref}_\perp\ S}$$

and the result holds.

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

$$\text{(Sderef)}\frac{\text{(Sl)}\ \dfrac{l : S \in \Sigma}{\Gamma; \Sigma; \ell_c \vdash l_\ell : \mathsf{Ref}_\ell\ S}}{\Gamma; \Sigma; \ell_c \vdash !l_\ell : S \curlyvee \ell}$$

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

$$\text{(Sderef)}\frac{\text{(Sl)}\ \dfrac{l : S \in \Sigma}{\Gamma; \Sigma; \ell'_c \vdash l_\ell : \mathsf{Ref}_{\ell'}\ S}}{\Gamma; \Sigma; \ell'_c \vdash !l_\ell : S \curlyvee \ell'}$$

where $\ell' = \ell$. and the result holds.

Case (Sassgn). Then $t = l_\ell{:=}v$ and

$$\text{(Sasgn)}\frac{\dfrac{l : S \in \Sigma}{\Gamma; \Sigma; \ell_c \vdash l_\ell : \mathsf{Ref}_\ell\ S} \quad \dfrac{\mathcal{D}}{\Gamma; \Sigma; \ell_c \vdash v : S_2} \quad S_2 <: S \quad \ell_c \curlyvee \ell \preccurlyeq \mathsf{label}(S)}{\Gamma; \Sigma; \ell_c \vdash l_\ell{:=}v : \mathsf{Unit}_\perp}$$

Suppose ℓ'_c such that $\ell'_c \preccurlyeq \ell_c$. Considering that $\ell'_c \curlyvee \ell \preccurlyeq \ell_c \curlyvee \ell \preccurlyeq label(S)$, and $S'_2 <: S_2 <: S$, then:

$$\text{(Sasgn)}\frac{\dfrac{l : S \in \Sigma}{\Gamma; \Sigma; \ell'_c \vdash l_\ell : \mathsf{Ref}_\ell\ S} \quad \dfrac{\mathcal{D}}{\Gamma; \Sigma; \ell'_c \vdash v : S'_2} \quad S'_2 <: S \quad \ell'_c \curlyvee \ell \preccurlyeq \mathsf{label}(S)}{\Gamma; \Sigma; \ell'_c \vdash l_\ell{:=}v : \mathsf{Unit}_\perp}$$

but

$$\frac{}{\mathsf{Unit}_\perp <: \mathsf{Unit}_\perp}$$

and therefore the result holds.

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

$$\text{(S::)}\frac{\dfrac{\mathcal{D}}{\Gamma; \Sigma; \ell_c \vdash v : S_1} \quad S_1 <: S}{\Gamma; \Sigma; \ell_c \vdash v :: S : S}$$

Suppose ℓ'_c such that $\ell'_c \preccurlyeq \ell_c$, then by Lemma 13

$$\text{(S::)}\frac{\dfrac{\mathcal{D}}{\Gamma; \Sigma; \ell'_c \vdash v : S_1} \quad S_1 <: S}{\Gamma; \Sigma; \ell'_c \vdash v :: S : S}$$

and the result holds.

□

Lemma 12 (Substitution)**.** *If* $\Gamma, x : S_1; \Sigma; \ell_c \vdash t : S$ *and* $\Gamma; \Sigma; \ell_c \vdash v : S'_1$ *such that* $S'_1 <: S_1$*, then* $\Gamma; \Sigma; \ell_c \vdash [v/x]t : S'$ *such that* $S' <: S$.

Proof. By induction on the derivation of $\Gamma, x : S_1; \Sigma; \ell_c \vdash t : S$. □

Lemma 13. *If* $\Gamma; \Sigma; \ell_c \vdash v : S$ *then* $\forall \ell'_c$, $\Gamma; \Sigma; \ell'_c \vdash v : S$.

Proof. By induction on the derivation of $\Gamma; \Sigma; \ell_c \vdash v : S$ observing that for values, there is no premise that depends on ℓ_c. □

Proposition 14 ($\longrightarrow$ is well defined)**.** *If* $\Gamma; \Sigma; \ell_c \vdash t : S$, $\Gamma; \Sigma \vdash \mu$ *and* $\forall\ \ell_r$*, such that* $\ell_r \preccurlyeq \ell_c$, $t \mid \mu \xrightarrow{\ell_r} t' \mid \mu'$ *then, for some* $\Sigma' \supseteq \Sigma$, $\Gamma; \Sigma'; \ell_c \vdash t' : S'$*, where* $S' <: S$ *and* $\Gamma; \Sigma' \vdash \mu'$.

Proof.

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

$$\text{(S}\oplus\text{)}\frac{\text{(Sb)}\ \dfrac{}{\Gamma; \Sigma; \ell_c \vdash b_{1\ell_1} : \mathsf{Bool}_{\ell_1}} \quad \text{(Sb)}\ \dfrac{}{\Gamma; \Sigma; \ell_c \vdash b_{2\ell_2} : \mathsf{Bool}_{\ell_2}}}{\Gamma; \Sigma; \ell_c \vdash b_{1\ell_1} \oplus b_{2\ell_2} : \mathsf{Bool}_{(\ell_1 \curlyvee \ell_2)}}$$

Suppose ℓ_r such that $\ell_r \preccurlyeq \ell_c$, then

$$\begin{array}{ll} & b_{1\ell_1} \oplus b_{2\ell_2} \mid \mu \\ \xrightarrow{\ell_r} & (b_1\ [\![\oplus]\!]\ b_2)_{(\ell_1 \curlyvee \ell_2)} \mid \mu \end{array}$$

Then

$$\text{(S}\oplus\text{)}\frac{}{\ell_c \vdash (b_1\ [\![\oplus]\!]\ b_2)_{(\ell_1 \curlyvee \ell_2)} : \mathsf{Bool}_{(\ell_1 \curlyvee \ell_2)}}$$

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

$$\text{(Sprot)}\frac{\dfrac{}{\Gamma; \Sigma; \ell_c \curlyvee \ell \vdash v : S}}{\Gamma; \Sigma; \ell_c \vdash \mathsf{prot}_\ell v : S \curlyvee \ell}$$

Suppose ℓ_r such that $\ell_r \preccurlyeq \ell_c$, then

$$\mathsf{prot}_\ell v \mid \mu \xrightarrow{\ell_r} v \curlyvee \ell \mid \mu$$

But by Lemma 11, $\Gamma; \Sigma; \ell_c \vdash v : S$.

$$\frac{}{\Gamma; \Sigma; \ell_c \vdash v \curlyvee \ell : S \curlyvee \ell}$$

and the result holds.

Case (Sapp). Then $t = (\lambda^{\ell'_c} x : S_{11}.t)_\ell\ v$ and

$$\text{(S}\lambda\text{)}\ \frac{\dfrac{\mathcal{D}_1}{\Gamma, x : S_{11}; \Sigma; \ell'_c \vdash t : S_{12}}}{\Gamma; \Sigma; \ell_c \vdash (\lambda^{\ell'_c} x : S_{11}.t)_\ell : S_{11} \xrightarrow{\ell'_c}_\ell S_{12}}$$

$$\text{(Sapp)}\frac{\dfrac{\mathcal{D}_2}{\Gamma; \Sigma; \ell_c \vdash v : S_2} \quad \ell_c \curlyvee \ell \preccurlyeq \ell'_c \quad S_2 <: S_{11}}{\Gamma; \Sigma; \ell_c \vdash (\lambda^{\ell'_c} x : S_{11}.t)_\ell\ v : S_{12} \curlyvee \ell}$$

Suppose ℓ_r such that $\ell_r \preccurlyeq \ell_c$, and

$$(\lambda^{\ell'_c} x : S.t)_\ell\ v \mid \mu \xrightarrow{\ell_r} \mathsf{prot}_\ell[v/x]t \mid \mu$$

But as $\ell_c \curlyvee \ell \preccurlyeq \ell'_c$ then by Lemma 11, $\Gamma; \Sigma; \ell_c \curlyvee \ell \vdash t : S_{12}$.
By Lemma 12 and Lemma 13, $\Gamma; \Sigma; \ell_c \curlyvee \ell \vdash [v/x]t : S'_{12}$, where $S'_{12} <: S_{12}$. Then

$$\text{(Sprot)}\frac{\dfrac{\mathcal{D}'_1}{\Gamma; \Sigma; \ell_c \curlyvee \ell \vdash [v/x]t : S'_{12}}}{\Gamma; \Sigma; \ell_c \vdash \mathsf{prot}_\ell[v/x]t : S'_{12} \curlyvee \ell}$$

Where $S'_{12} \curlyvee \ell <: S_{12} \curlyvee \ell$ and the result holds.

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

$$(\text{Sif})\ \frac{\dfrac{\mathcal{D}_0}{\Gamma;\Sigma;\ell_c \vdash \mathsf{true}_\ell : \mathsf{Bool}_\ell} \quad \dfrac{\mathcal{D}_1}{\Gamma;\Sigma;\ell_c \curlyvee \ell \vdash t_1 : S_1} \quad \dfrac{\mathcal{D}_2}{\Gamma;\Sigma;\ell_c \curlyvee \ell \vdash t_2 : S_2}}{\Gamma;\Sigma;\ell_c \vdash \mathsf{if}\ \mathsf{true}_\ell\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 : (S_1 \ddot{\vee} S_2) \curlyvee \ell}$$

Suppose ℓ_r such that $\ell_r \preccurlyeq \ell_c$, then if

$$\mathsf{if}\ \mathsf{true}_\ell\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \mid \mu \xrightarrow{\ell_r} \mathsf{prot}_\ell t_1 \mid \mu$$

Then

$$(\text{Sprot})\ \frac{\dfrac{\mathcal{D}_1}{\Gamma;\Sigma;\ell_c \curlyvee \ell \vdash t_1 : S_1}}{\Gamma;\Sigma;\ell_c \vdash \mathsf{prot}_\ell t_1 : S_1 \curlyvee \ell}$$

and by definition of the join operator, $S_1 \curlyvee \ell <: (S_1 \ddot{\vee} S_2) \curlyvee \ell$ and the result holds.

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

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

$$(\text{Sref})\ \frac{\Gamma;\Sigma;\ell_c \vdash v : S' \quad S' <: S \quad \ell_c \preccurlyeq \mathsf{label}(S)}{\Gamma;\Sigma;\ell_c \vdash \mathsf{ref}^S\ v : \mathsf{Ref}_\bot\ S}$$

Suppose ℓ_r such that $\ell_r \preccurlyeq \ell_c$, then

$$\mathsf{ref}^S\ v \mid \mu \xrightarrow{\ell_r} l_\bot \mid \mu[l \mapsto v \curlyvee \ell_r]$$

where $l \notin dom(\mu)$.
Let us take $\Sigma' = \Sigma, l : S$ and let us call $\mu' = \mu[l \mapsto v \curlyvee \ell_r]$. Then as $dom(\mu) = dom(\Sigma)$ then $dom(\mu') = dom(\Sigma')$. Also, as $\ell_r \preccurlyeq \ell_c \preccurlyeq label(S)$ then by Lemma 13, $\Gamma;\Sigma';\bot \vdash v : S' \curlyvee \ell_r$ and $S' \curlyvee \ell_r <: \Sigma(l) = S$. Therefore $\Gamma;\Sigma' \vdash \mu'$.

Then

$$(\text{Sl})\ \frac{l : S \in \Sigma'}{\Gamma;\Sigma';\ell_c \vdash l_\bot : \mathsf{Ref}_\bot\ S}$$

and the result holds.

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

$$(\text{Sderef})\ \frac{(\text{Sl})\ \dfrac{l : S \in \Sigma}{\Gamma;\Sigma;\ell_c \vdash l_\ell : \mathsf{Ref}_\ell\ S}}{\Gamma;\Sigma;\ell_c \vdash !l_\ell : S \curlyvee \ell}$$

Suppose ℓ_r such that $\ell_r \preccurlyeq \ell_c$, then

$$!l_\ell \mid \mu \xrightarrow{\ell_r} v \curlyvee \ell \mid \mu \text{ where } \mu(l) = v$$

Also $\Gamma;\Sigma \vdash \mu$ then $\Gamma;\Sigma;\bot \vdash \mu(l) : S'$ and $S' <: S$. By Lemma 13, $\Gamma;\Sigma;\ell_c \vdash v : S'$

$$\frac{}{\Gamma;\Sigma;\ell_c \vdash v \curlyvee \ell : S' \curlyvee \ell}$$

But $S' \curlyvee \ell <: S \curlyvee \ell$ and the result holds.

Case (Sassgn). Then $t = l_\ell{:=}v$ and

$$(\text{Sasgn})\ \frac{\dfrac{l : S \in \Sigma}{\Gamma;\Sigma;\ell_c \vdash l_\ell : \mathsf{Ref}_\ell\ S} \quad \dfrac{\mathcal{D}}{\Gamma;\Sigma;\ell_c \vdash v : S_2} \quad S_2 <: S \quad \ell_c \curlyvee \ell \preccurlyeq \mathsf{label}(S)}{\Gamma;\Sigma;\ell_c \vdash l_\ell{:=}v : \mathsf{Unit}_\bot}$$

Suppose ℓ_r such that $\ell_r \preccurlyeq \ell_c$, then

$$l_\ell{:=}v \mid \mu \xrightarrow{\ell_r} \mathsf{unit}_\bot \mid \mu[l \mapsto v \curlyvee \ell_r \curlyvee \ell]$$

Let us call $\mu' = \mu[l \mapsto v \curlyvee \ell_r \curlyvee \ell]$. Also $\Gamma;\Sigma \vdash \mu$ then $dom(\mu') = dom(\Sigma)$, and $\Gamma;\Sigma;\ell_c \vdash v : S'$ where $S' <: S$. Therefore $\Gamma;\Sigma;\ell_c \vdash v \curlyvee \ell_r \curlyvee \ell : S' \curlyvee \ell_r \curlyvee \ell$. But $\ell_r \curlyvee \ell \preccurlyeq \ell_c \curlyvee \ell \preccurlyeq label(S)$, then $S' \curlyvee \ell_r \curlyvee \ell <: S$ and therefore $\Gamma;\Sigma \vdash \mu'$. Also

$$(\text{Su})\ \frac{}{\Gamma;\Sigma;\ell_c \vdash \mathsf{unit}_\bot : \mathsf{Unit}_\bot}$$

but

$$\frac{}{\mathsf{Unit}_\bot <: \mathsf{Unit}_\bot}$$

and therefore the result holds.

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

$$(\text{S::})\ \frac{\dfrac{\mathcal{D}}{\Gamma;\Sigma;\ell_c \vdash v : S_1} \quad S_1 <: S}{\Gamma;\Sigma;\ell_c \vdash v :: S : S}$$

Suppose ℓ_r such that $\ell_r \preccurlyeq \ell_c$, then

$$v :: S \mid \mu \xrightarrow{\ell_r} v \curlyvee label(S) \mid \mu$$

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

$$\frac{}{\Gamma;\Sigma;\ell_c \vdash v \curlyvee label(S) : S}$$

and the result holds.

□

Proposition 15 (Canonical forms)**.** *Consider a value v such that $.;\Sigma;\ell_c \vdash v : S$. Then:*

1. *If $S = \mathsf{Bool}_\ell$ then $v = b_\ell$ for some b.*
2. *If $S = \mathsf{Unit}_\ell$ then $v = \mathsf{unit}_\ell$.*
3. *If $S = S_1 \xrightarrow{\ell'_c}_\ell S_2$ then $v = (\lambda^{\ell'_c} x : S_1.t_2)$ for some t_2 and ℓ'_c.*
4. *If $S = \mathsf{Ref}_\ell\ S$ then $v = l_\ell$ for some location l.*

Proof. By inspection of the type derivation rules. □

Proposition 16 (Preservation)**.** *If $\Gamma;\Sigma;\ell_c \vdash t : S$, $\Gamma;\Sigma \vdash \mu$ and $\forall\ \ell_r$, such that $\ell_r \preccurlyeq \ell_c$, $t \mid \mu \overset{\ell_r}{\longmapsto} t' \mid \mu'$ then, for some $\Sigma' \supseteq \Sigma$, $\Gamma;\Sigma';\ell_c \vdash t' : S'$ where $S' <: S$ and $\Gamma;\Sigma' \vdash \mu'$.*

Proof. By induction on the structure of t.

Case (Sb, Su, Sλ, Sl). t is a value.

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

$$(\text{Sprot})\ \frac{\Gamma;\Sigma;\ell_c \curlyvee \ell \vdash t_1 : S_1}{\Gamma;\Sigma;\ell_c \vdash \mathsf{prot}_\ell t_1 : S_1 \curlyvee \ell}$$

By induction hypotheses, one of the following holds:

1. t_1 is a value. Then by (R$\rightarrow$) and Canonical Forms (Lemma 15). $t \mid \mu \overset{\ell_r}{\longmapsto} t' \mid \mu$ and by Prop 14, $\Gamma;\Sigma;\ell_c \vdash t' : S'$ where $S' <: S$ and the result holds.
2. Suppose ℓ_r such that $\ell_r \preccurlyeq \ell_c$, then

$$(\text{Rprot})\ \frac{t_1 \mid \mu \overset{\ell_r \curlyvee \ell}{\longmapsto} t_2 \mid \mu'}{\mathsf{prot}_\ell t_1 \mid \mu \overset{\ell_r}{\longmapsto} \mathsf{prot}_\ell t_2 \mid \mu'}$$

As $\ell_r \preccurlyeq \ell_c$ then $\ell_r \curlyvee \ell \preccurlyeq \ell_c \curlyvee \ell$. Using induction hypotheses $\Gamma;\Sigma;\ell_c \curlyvee \ell \vdash t_2 : S'_1$ where $S'_1 <: S_1$ and $\Gamma;\Sigma \vdash \mu'$. Therefore

$$(\text{Sprot})\ \frac{\Gamma;\Sigma;\ell_c \curlyvee \ell \vdash t_2 : S'_1}{\Gamma;\Sigma;\ell_c \vdash \mathsf{prot}_\ell t_2 : S'_1 \curlyvee \ell}$$

but $S'_1 \curlyvee \ell <: S_1 \curlyvee \ell$ and the result holds.

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

$$(\text{S}\oplus)\ \frac{\Gamma;\Sigma;\ell_c \vdash t_1 : \mathsf{Bool}_{\ell_1} \quad \Gamma;\Sigma;\ell_c \vdash t_2 : \mathsf{Bool}_{\ell_2}}{\Gamma;\Sigma;\ell_c \vdash t_1 \oplus t_2 : \mathsf{Bool}_{(\ell_1 \curlyvee \ell_2)}}$$

By induction hypotheses, one of the following holds:

1. t_1 is a value. Then by induction on t_2 one of the following holds:
   (a) t_2 is a value. Then by Canonical Forms (Lemma 15)

$$(\mathrm{R}\to)\frac{t \mid \mu \xrightarrow{\ell_r} t' \mid \mu}{t \mid \mu \overset{\ell_r}{\longmapsto} t' \mid \mu}$$

   and by Prop 14, $\Gamma; \Sigma; \ell_c \vdash t' : S'$, where $S' <: S$, therefore the result holds.

   (b) $t_2 \mid \mu \overset{\ell_r{}'}{\longmapsto} t_2' \mid \mu'$ for all $\ell_r{}'$ such that $\ell_r{}' \preccurlyeq \ell_c$, in particular we pick $\ell_r{}' = \ell_r$. Then by induction hypothesis, $\Gamma; \Sigma; \ell_c \vdash t_2 : \mathsf{Bool}_{\ell_2'}$, where $\mathsf{Bool}_{\ell_2'} <: \mathsf{Bool}_{\ell_2}$ and $\Gamma; \Sigma \vdash \mu'$.

   Then by (Sf), $t \mid \mu \overset{\ell_r}{\longmapsto} t_1 \oplus t_2' \mid \mu'$ and:

$$(\mathrm{S}\oplus)\frac{\Gamma; \Sigma; \ell_c \vdash t_1 : \mathsf{Bool}_{\ell_1} \qquad \Gamma; \Sigma; \ell_c \vdash t_2' : \mathsf{Bool}_{\ell_2'}}{\Gamma; \Sigma; \ell_c \vdash t_1 \oplus t_2' : \mathsf{Bool}_{(\ell_1 \curlyvee \ell_2')}}$$

   but

$$\frac{(\ell_1 \curlyvee \ell_2') \preccurlyeq (\ell_1 \curlyvee \ell_2)}{\mathsf{Bool}_{(\ell_1 \curlyvee \ell_2')} <: \mathsf{Bool}_{(\ell_1 \curlyvee \ell_2)}}$$

   and the result holds.

2. $t_1 \mid \mu \overset{\ell_r}{\longmapsto} t_1' \mid \mu'$ for all $\ell_r{}'$ such that $\ell_r{}' \preccurlyeq \ell_c$, in particular we pick $\ell_r{}' = \ell_r$. Then by induction hypotheses, $\Gamma; \Sigma; \ell_c \vdash t_1' : \mathsf{Bool}_{\ell_1'}$ where $\mathsf{Bool}_{\ell_1'} <: \mathsf{Bool}_{\ell_1}$, and $\Gamma; \Sigma \vdash \mu'$. Then by (S$f$), $t \mid \mu \overset{\ell_r}{\longmapsto} t_1' \oplus t_2 \mid \mu'$ and:

$$(\mathrm{S}\oplus)\frac{\Gamma; \Sigma; \ell_c \vdash t_1' : \mathsf{Bool}_{\ell_1'} \qquad \Gamma; \Sigma; \ell_c \vdash t_2 : \mathsf{Bool}_{\ell_2}}{\Gamma; \Sigma; \ell_c \vdash t_1' \oplus t_2 : \mathsf{Bool}_{(\ell_1' \curlyvee \ell_2)}}$$

   but

$$\frac{(\ell_1' \curlyvee \ell_2) \preccurlyeq (\ell_1 \curlyvee \ell_2)}{\mathsf{Bool}_{(\ell_1' \curlyvee \ell_2)} <: \mathsf{Bool}_{(\ell_1 \curlyvee \ell_2)}}$$

   and the result holds.

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

$$(\mathrm{Sapp})\frac{\Gamma; \Sigma; \ell_c \vdash t_1 : S_{11} \xrightarrow{\ell_c'} \ell S_{12} \qquad \Gamma; \Sigma; \ell_c \vdash t_2 : S_2 \qquad S_2 <: S_{11} \qquad \ell_c \curlyvee \ell \preccurlyeq \ell_c'}{\Gamma; \Sigma; \ell_c \vdash t_1\ t_2 : S_{12} \curlyvee \ell}$$

By induction hypotheses, one of the following holds:

1. t_1 is a value. Then by Canonical Forms (Lemma 15), and induction on t_2 one of the following holds:
   (a) t_2 is a value. Then by Canonical Forms (Lemma 15)

$$(\mathrm{R}\to)\frac{t \mid \mu \xrightarrow{\ell_r} t' \mid \mu}{t \mid \mu \overset{\ell_r}{\longmapsto} t' \mid \mu}$$

   and by Prop 14 $\Gamma; \Sigma; \ell_c \vdash t' : S'$, where $S' <: S$, therefore the result holds.

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

   Then by (Sf), $t \mid \mu \overset{\ell_r}{\longmapsto} t_1\ t_2' \mid \mu'$. But $S_2' <: S_2 <: S_{11}$ and then:

$$(\mathrm{Sapp})\frac{\Gamma; \Sigma; \ell_c \vdash t_1 : S_{11} \xrightarrow{\ell_c'} \ell S_{12} \qquad \Gamma; \Sigma; \ell_c \vdash t_2' : S_2' \qquad S_2' <: S_{11} \qquad \ell_c \curlyvee \ell \preccurlyeq \ell_c'}{\Gamma; \Sigma; \ell_c \vdash t_1\ t_2 : S_{12} \curlyvee \ell}$$

   and the result holds.

2. $t_1 \mid \mu \overset{\ell_r}{\longmapsto} t_1' \mid \mu'$ for all $\ell_r{}'$ such that $\ell_r{}' \preccurlyeq \ell_c$, in particular we pick $\ell_r{}' = \ell_r$. Then by induction hypotheses, $\Gamma; \Sigma; \ell_c \vdash t_1' : S_{11}' \xrightarrow{\ell_c''} \ell' S_{12}'$ where $S_{11}' \xrightarrow{\ell_c''} \ell' S_{12}' <: S_{11} \xrightarrow{\ell_c'} \ell S_{12}$, and $\Gamma; \Sigma \vdash \mu'$. Then by (S$f$), $t \mid \mu \overset{\ell_r}{\longmapsto} t_1'\ t_2 \mid \mu'$. By definition of subtyping, $S_2 <: S_{11} <: S_{11}'$, $\ell_c' \preccurlyeq \ell_c''$ and $\ell' \preccurlyeq \ell$. Therefore $\ell_c \curlyvee \ell' \preccurlyeq \ell_c \curlyvee \ell \preccurlyeq \ell_c' \preccurlyeq \ell_c''$. Then

$$(\mathrm{Sapp})\frac{\Gamma; \Sigma; \ell_c \vdash t_1' : S_{11}' \xrightarrow{\ell_c''} \ell' S_{12}' \qquad \Gamma; \Sigma; \ell_c \vdash t_2 : S_2 \qquad S_2 <: S_{11}' \qquad \ell_c \curlyvee \ell' \preccurlyeq \ell_c''}{\Gamma; \Sigma; \ell_c \vdash t_1'\ t_2 : S_{12}' \curlyvee \ell'}$$

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

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

$$(\mathrm{Sif})\frac{\Gamma; \Sigma; \ell_c \vdash t_0 : \mathsf{Bool}_\ell \qquad \Gamma; \Sigma; \ell_c \curlyvee \ell \vdash t_1 : S_1 \qquad \Gamma; \Sigma; \ell_c \curlyvee \ell \vdash t_2 : S_2}{\Gamma; \Sigma; \ell_c \vdash \mathsf{if}\ t_0\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 : (S_1 \mathbin{\ddot{\curlyvee}} S_2) \curlyvee \ell}$$

By induction hypotheses, one of the following holds:

1. t_0 is a value. Then by Canonical Forms (Lemma 15)

$$(\mathrm{R}\to)\frac{t \mid \mu \xrightarrow{\ell_r} t' \mid \mu}{t \mid \mu \overset{\ell_r}{\longmapsto} t' \mid \mu}$$

   and by Prop 14, $\Gamma; \Sigma; \ell_c \vdash t' : S'$, where $S' <: S$, therefore the result holds.

2. $t_0 \mid \mu \overset{\ell_r{}'}{\longmapsto} t_0' \mid \mu'$ for all $\ell_r{}'$ such that $\ell_r{}' \preccurlyeq \ell_c$, in particular we pick $\ell_r{}' = \ell_r$. Then by induciton hypothesis, $\Gamma; \Sigma; \ell_c \vdash t_0' : \mathsf{Bool}_{\ell'}$, where $\mathsf{Bool}_{\ell'} <: \mathsf{Bool}_\ell$ and $\Gamma; \Sigma \vdash \mu'$. Then by (S$f$), $t \mid \mu \overset{\ell_r}{\longmapsto} \mathsf{if}\ t_0'\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \mid \mu'$. As $\ell_c \curlyvee \ell' \preccurlyeq \ell_c \curlyvee \ell$, by Lemma 11, $\Gamma; \Sigma; \ell_c \curlyvee \ell' \vdash t_1 : S_1'$ and $\Gamma; \Sigma; \ell_c \curlyvee \ell' \vdash t_2 : S_2'$, where $S_1' <: S_1$ and $S_2' <: S_2$. Therefore:

$$(\mathrm{Sif})\frac{\Gamma; \Sigma; \ell_c \vdash t_0' : \mathsf{Bool}_{\ell'} \qquad \Gamma; \Sigma; \ell_c \curlyvee \ell' \vdash t_1 : S_1' \qquad \Gamma; \Sigma; \ell_c \curlyvee \ell \vdash t_2 : S_2'}{\Gamma; \Sigma; \ell_c \vdash \mathsf{if}\ t_0'\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 : (S_1' \mathbin{\ddot{\curlyvee}} S_2') \curlyvee \ell'}$$

   but by definition of join and subtyping $(S_1' \mathbin{\ddot{\curlyvee}} S_2') \curlyvee \ell' <: (S_1 \mathbin{\ddot{\curlyvee}} S_2) \curlyvee \ell$ and the result holds.

Case (S::). Then $t = t_1 :: S_2$ and

$$(\mathrm{S::})\frac{\Gamma; \Sigma; \ell_c \vdash t_1 : S_1 \qquad S_1 <: S_2}{\Gamma; \Sigma; \ell_c \vdash t_1 :: S_2 : S_2}$$

By induction hypotheses, one of the following holds:

1. t_1 is a value. Then

$$(\mathrm{R}\to)\frac{t \mid \mu \xrightarrow{\ell_r} t' \mid \mu}{t \mid \mu \overset{\ell_r}{\longmapsto} t' \mid \mu}$$

   and by Prop 14, $\Gamma; \Sigma; \ell_c \vdash t' : S'$, where $S' <: S$, therefore the result holds.

2. $t_1 \mid \mu \overset{\ell_r}{\longmapsto} t_1' \mid \mu'$ for all $\ell_r{}'$ such that $\ell_r{}' \preccurlyeq \ell_c$, in particular we pick $\ell_r{}' = \ell_r$. Then by induction hypothesis, $\Gamma; \Sigma; \ell_c \vdash t_1' : S_1'$, where $S_1' <: S_1$ and $\Gamma; \Sigma \vdash \mu'$. Then by (S$f$), $t \mid \mu \overset{\ell_r}{\longmapsto} t_1' :: S_2 \mid \mu'$. Also, $S_1' <: S_1 <: S_2$ and therefore:

$$(\mathrm{S::})\frac{\Gamma; \Sigma; \ell_c \vdash t_1' : S_1' \qquad S_1' <: S_2}{\Gamma; \Sigma; \ell_c \vdash t_1' :: S_2 : S_2}$$

   and the result holds.

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

$$(\text{Sref})\frac{\Gamma;\Sigma;\ell_c \vdash t_1 : S_1' \qquad S_1' <: S_1 \qquad \ell_c \preccurlyeq \mathsf{label}(S_1)}{\Gamma;\Sigma;\ell_c \vdash \mathsf{ref}^{S_1,\ell_c}\ t_1 : \mathsf{Ref}_\bot\ S_1}$$

By induction hypotheses, one of the following holds:

1. t_1 is a value. Then

$$(\text{R}\rightarrow)\frac{t \mid \mu \xrightarrow{\ell_r} t' \mid \mu'}{t \mid \mu \overset{\ell_r}{\longmapsto} t' \mid \mu'}$$

and by Prop 14, $\Gamma;\Sigma;\ell_c \vdash t' : S'$, where $S' <: S$ and $\Gamma;\Sigma \vdash \mu'$, therefore the result holds.

2. $t_1 \mid \mu \overset{\ell_r}{\longmapsto} t_1' \mid \mu'$ for all ${\ell_r}'$ such that ${\ell_r}' \preccurlyeq \ell_c$, in particular we pick ${\ell_r}' = \ell_r$. Then by induction hypothesis, $\Gamma;\Sigma;\ell_c \vdash t_1' : S_1''$ where $S_1'' <: S_1'$ and $\Gamma;\Sigma \vdash \mu'$. Then by (S$f$), $t \mid \mu \overset{\ell_r}{\longmapsto} \mathsf{ref}^{S_1}\ t_1' \mid \mu'$ and:

$$(\text{Sref})\frac{\Gamma;\Sigma;\ell_c \vdash t_1' : S_1'' \qquad S_1'' <: S_1 \qquad \ell_c \preccurlyeq \mathsf{label}(S_1)}{\Gamma;\Sigma;\ell_c \vdash \mathsf{ref}^{S_1}\ t_1' : \mathsf{Ref}_\bot\ S_1}$$

and the result holds.

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

$$(\text{Sderef})\frac{\Gamma;\Sigma;\ell_c \vdash t_1 : \mathsf{Ref}_\ell\ S_1}{\Gamma;\Sigma;\ell_c \vdash !t_1 : S_1 \curlyvee \ell}$$

By induction hypotheses, one of the following holds:

1. t_1 is a value. Then by Canonical Forms (Lemma 15)

$$(\text{R}\rightarrow)\frac{t \mid \mu \xrightarrow{\ell_r} t' \mid \mu}{t \mid \mu \overset{\ell_r}{\longmapsto} t' \mid \mu}$$

and by Prop 14, $\Gamma;\Sigma;\ell_c \vdash t' : S'$, where $S' <: S$, therefore the result holds.

2. $t_1 \mid \mu \overset{\ell_r}{\longmapsto} t_1' \mid \mu'$ for all ${\ell_r}'$ such that ${\ell_r}' \preccurlyeq \ell_c$, in particular we pick ${\ell_r}' = \ell_r$. Then by induction hypothesis, $\Gamma;\Sigma;\ell_c \vdash t_1' : \mathsf{Ref}_{\ell'}\ S_1$ where $\mathsf{Ref}_{\ell'}\ S_1 <: \mathsf{Ref}_\ell\ S_1$ and $\Gamma;\Sigma \vdash \mu'$. Then by (S$f$), $t \mid \mu \overset{\ell_r}{\longmapsto} t' \mid \mu'$ and:

$$(\text{Sderef})\frac{\Gamma;\Sigma;\ell_c \vdash t_1' : \mathsf{Ref}_{\ell'}\ S_1}{\Gamma;\Sigma;\ell_c \vdash !t_1' : S_1 \curlyvee \ell'}$$

but $S_1 \curlyvee \ell' <: S_1 \curlyvee \ell$ and the result holds.

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

$$(\text{Sasgn})\frac{\Gamma;\Sigma;\ell_c \vdash t_1 : \mathsf{Ref}_\ell\ S_1 \qquad \Gamma;\Sigma;\ell_c \vdash t_2 : S_2 \quad S_2 <: S_1 \qquad \ell_c \curlyvee \ell \preccurlyeq \mathsf{label}(S_1)}{\Gamma;\Sigma;\ell_c \vdash t_1 := t_2 : \mathsf{Unit}_\bot}$$

By induction hypotheses, one of the following holds:

1. t_1 is a value. Then by Canonical Forms (Lemma 15), and induction on t_2 one of the following holds:
   (a) t_2 is a value. Then by Canonical Forms (Lemma 15)

$$\frac{t \mid \mu \xrightarrow{\ell_r} t' \mid \mu'}{t \mid \mu \overset{\ell_r}{\longmapsto} t' \mid \mu'}$$

   and by Prop 14, $\Gamma;\Sigma;\ell_c \vdash t' : S'$, where $S' <: S$ and $\Gamma;\Sigma \vdash \mu'$, therefore the result holds.

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

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

$$(\text{Sasgn})\frac{\Gamma;\Sigma;\ell_c \vdash t_1 : \mathsf{Ref}_\ell\ S_1 \qquad \Gamma;\Sigma;\ell_c \vdash t_2' : S_2' \quad S_2' <: S_1 \qquad \ell_c \curlyvee \ell \preccurlyeq \mathsf{label}(S_1)}{\Gamma;\Sigma;\ell_c \vdash t_1 := t_2' : \mathsf{Unit}_\bot}$$

   and the result holds.

2. $t_1 \mid \mu \overset{\ell_r}{\longmapsto} t_1' \mid \mu'$ for all ${\ell_r}'$ such that ${\ell_r}' \preccurlyeq \ell_c$, in particular we pick ${\ell_r}' = \ell_r$. Then by induction hypotheses, $\Gamma;\Sigma;\ell_c \vdash t_1' : \mathsf{Ref}_{\ell'}\ S_1$, where $\mathsf{Ref}_{\ell'}\ S_1 <: \mathsf{Ref}_\ell\ S_1$ and $\Gamma;\Sigma \vdash \mu'$. Then by (S$f$), $t \mid \mu \overset{\ell_r}{\longmapsto} t_1' := t_2 \mid \mu'$. As $\ell' \preccurlyeq \ell$ then $\ell_c \curlyvee \ell' \preccurlyeq \ell_c \curlyvee \ell \preccurlyeq \mathsf{label}(S_1)$, and therefore:

$$(\text{Sasgn})\frac{\Gamma;\Sigma;\ell_c \vdash t_1' : \mathsf{Ref}_{\ell'}\ S_1 \qquad \Gamma;\Sigma;\ell_c \vdash t_2 : S_2 \quad S_2 <: S_1 \qquad \ell_c \curlyvee \ell' \preccurlyeq \mathsf{label}(S_1)}{\Gamma;\Sigma;\ell_c \vdash t_1 := t_2 : \mathsf{Unit}_\bot}$$

   and the result holds.

□

B. Gradualizing the Static Semantics

In this section we present proofs and auxiliary definitions related to gradualizing $\text{SSL}_{\mathsf{Ref}}$. In section B.1, we show the proof of optimality and soundness of the abstraction. In section B.2, we present some auxiliary definitions needed in gradualizing $\text{SSL}_{\mathsf{Ref}}$. Finally in section B.4, we present the proof for the Static Gradual Guarantee.

B.1 From Gradual Labels to Gradual Types

Proposition 1 (α_ℓ is Sound)**.** *If $\widehat{\ell}$ is a non-empty set of labels, then $\widehat{\ell} \subseteq \gamma_\ell(\alpha_\ell(\widehat{\ell}))$.*

Proof. By case analysis on the structure of $\widehat{\ell}$. If $\widehat{\ell} = \{\,\ell\,\}$ then $\gamma_\ell(\alpha_\ell(\{\,\ell\,\})) = \gamma_\ell(\ell) = \{\,\ell\,\} = \widehat{\ell}$, otherwise
$\gamma_\ell(\alpha_\ell(\widehat{\ell})) = \gamma_\ell(?) = \textsc{Label} \supseteq \widehat{\ell}$. □

Proposition 2 (α_ℓ is Optimal)**.** *If $\widehat{\ell} \subseteq \gamma_\ell(\tilde{\ell})$ then $\alpha_\ell(\widehat{\ell}) \sqsubseteq \tilde{\ell}$.*

Proof. By case analysis on the structure of $\tilde{\ell}$. If $\tilde{\ell} = \ell$, $\gamma_\ell(\tilde{\ell}) = \{\,\ell\,\}$; $\widehat{\ell} \subseteq \{\,\ell\,\}, \widehat{\ell} \neq \emptyset$ implies $\alpha_\ell(\widehat{\ell}) = \alpha_\ell(\{\,\ell\,\}) = \ell \sqsubseteq \tilde{\ell}$ (if $\widehat{\ell} = \emptyset$, $\alpha_\ell(\widehat{\ell})$ is undefined). If $\tilde{\ell} = ?$, $\tilde{\ell}' \sqsubseteq \tilde{\ell}$ for all $\tilde{\ell}'$. □

Proposition 3 (α_S is Sound)**.** *If $\widehat{S}$ is a valid set of security types, then $\widehat{S} \subseteq \gamma_S(\alpha_S(\widehat{S}))$.*

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

$$\widehat{dom}(\widehat{S}) \sqsubset \widehat{S}$$
$$\widehat{cod}(j\widehat{S}) \sqsubset \widehat{S}$$

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

$$\widehat{dom}(\widehat{S}) = \{\, dom(S) \mid S \in \widehat{S} \,\}.$$

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

Case ($\{\,\overline{\mathsf{Bool}_{\ell_i}}\,\}$).

$$\begin{aligned}\gamma_S(\alpha_S(\{\,\overline{\mathsf{Bool}_{\ell_i}}\,\})) &= \gamma_S(\mathsf{Bool}_{\alpha_\ell(\{\,\overline{\ell_i}\,\})}) \\ &= \{\,\mathsf{Bool}_\ell \mid \ell \in \gamma_\ell(\alpha_\ell(\{\,\overline{\ell_i}\,\}))\,\} \\ &\supseteq \{\,\overline{\mathsf{Bool}_{\ell_i}}\,\} \text{ by soundness of } \alpha_\ell.\end{aligned}$$

Case $(\{\overline{S_{i1} \xrightarrow{\ell_c i} \ell_i S_{i2}}\})$.

$$\begin{aligned}
&\gamma_S(\alpha_S(\{\overline{S_{i1} \xrightarrow{\ell_c i} \ell_i S_{i2}}\})) \\
&= \gamma_S(\alpha_S(\{\overline{S_{i1}}\}))^{\alpha_\ell(\{\overline{\ell_c i}\})}_{\alpha_\ell(\{\overline{\ell_i}\})}\alpha_S(\{\overline{S_{i2}}\})) \\
&= \gamma_S(\alpha_S(\{\overline{S_{i1}}\}))^{\gamma_\ell(\alpha_\ell(\{\overline{\ell_c i}\}))}_{\gamma_\ell(\alpha_\ell(\{\overline{\ell_i}\}))}\gamma_S(\alpha_S(\{\overline{S_{i2}}\})) \\
&\supseteq \{\overline{S_{i1} \xrightarrow{\ell_c i} \ell_i S_{i2}}\}
\end{aligned}$$

by induction hypothesis on $\{\overline{S_{i1}}\}$ and $\{\overline{S_{i2}}\}$, and soundness of α_ℓ.

Case $(\{\overline{\mathsf{Ref}_{\ell_i}\ S_i}\})$.

$$\begin{aligned}
&\gamma_S(\alpha_S(\{\overline{\mathsf{Ref}_{\ell_i}\ S_i}\})) \\
&= \gamma_S(\mathsf{Ref}_{\alpha_\ell(\{\overline{\ell_i}\})}\ \alpha_S(\{\overline{S_i}\})) \\
&= \{\mathsf{Ref}_\ell\ S \mid \ell \in \gamma_\ell(\alpha_\ell(\{\overline{\ell_i}\})), S \in \gamma_S(\alpha_S(\{\overline{S_i}\}))\} \\
&\supseteq \{\overline{\mathsf{Ref}_{\ell_i}\ S_i}\}
\end{aligned}$$

by induction hypothesis on $\{\overline{S_i}\}$ and soundness of α_ℓ.

□

Proposition 4 (α_S is Optimal)**.** *If $\widehat{S}$ is a valid set of security types and $\widetilde{S} \subseteq \gamma_S(\widetilde{S})$ then $\alpha_S(\widehat{S}) \sqsubseteq \widetilde{S}$.*

Proof. By induction on the structure of $\widetilde{S}$.

Case $(\mathsf{Bool}_{\tilde{\ell}})$. $\gamma_S(\mathsf{Bool}_{\tilde{\ell}}) = \{\mathsf{Bool}_\ell \mid \ell \in \gamma_\ell(\tilde{\ell})\}$
So $\widehat{S} = \{\mathsf{Bool}_\ell \mid \ell \in \widehat{\ell}\}$ for some $\widehat{\ell} \subseteq \gamma_\ell(\tilde{\ell})$. By optimality of α_ℓ, $\alpha_\ell(\widehat{\ell}) \sqsubseteq \tilde{\ell}$, so $\alpha_S(\{\mathsf{Bool}_\ell \mid \ell \in \widehat{\ell}\}) = \mathsf{Bool}_{\alpha_\ell(\widehat{\ell})} \sqsubseteq \mathsf{Bool}_{\tilde{\ell}}$.

Case $(\widetilde{S}_1 \to_{\tilde{\ell}} \widetilde{S}_2)$. $\gamma_S(\widetilde{S}_1 \xrightarrow{\tilde{\ell}_c}_{\tilde{\ell}} \widetilde{S}_2) = \gamma_S(\widetilde{S}_1)^{\gamma_\ell(\tilde{\ell}_c)}_{\gamma_\ell(\tilde{\ell})}\gamma_S(\widetilde{S}_2)$.
So $\widehat{S} = \{\overline{S_{1i} \xrightarrow{\ell_c i}_{\tilde{\ell}_i} S_{2i}}\}$, with $\{\overline{S_{1i}}\} \subseteq \gamma_S(\widetilde{S}_1)$, $\{\overline{S_{1i}}\} \subseteq \gamma_S(\widetilde{S}_2)$, $\{\overline{\ell_c i}\} \subseteq \gamma_\ell(\tilde{\ell}_c)$ and $\{\overline{\ell_c i}\} \subseteq \gamma_\ell(\tilde{\ell})$. By induction hypothesis, $\alpha_S(\{\overline{S_{1i}}\}) \sqsubseteq \widetilde{S}_1$ and $\alpha_S(\{\overline{S_{2i}}\}) \sqsubseteq \widetilde{S}_2$, and by optimality of α_ℓ, $\alpha_\ell(\{\overline{\ell_c i}\}) \sqsubseteq \tilde{\ell}_c$ and $\alpha_\ell(\{\overline{\ell_i}\}) \sqsubseteq \tilde{\ell}$. Hence $\alpha_S(\{\overline{S_{1i} \xrightarrow{\ell_c i} \ell_i S_{2i}}\}) =$
$\alpha_S(\{\overline{S_{1i}}\})^{\alpha_\ell(\{\overline{\ell_c i}\})}_{\alpha_\ell(\{\overline{\ell_i}\})}\alpha_S(\{\overline{S_{2i}}\}) \sqsubseteq \widetilde{S}_1 \xrightarrow{\tilde{\ell}_c}_{\tilde{\ell}} \widetilde{S}_2$.

Case $(\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S})$. $\gamma_S(\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}) = \{\mathsf{Ref}_\ell\ S \mid \ell \in \gamma_\ell(\tilde{\ell}), S \in \gamma_S(\widetilde{S})\}$
So $\widehat{S} = \{\mathsf{Ref}_\ell\ S \mid \ell \in \widehat{\ell}, S \in \{\overline{S_i}\}\}$ for some $\{\overline{S_i}\} \subseteq \gamma_S(\widetilde{S})$ and some $\widehat{\ell} \subseteq \gamma_\ell(\tilde{\ell})$. By induction hypothesis $\alpha_S(\{\overline{S_i}\}) \sqsubseteq \widetilde{S}$ and by optimality of α_ℓ, $\alpha_\ell(\widehat{\ell}) \sqsubseteq \tilde{\ell}$, so $\alpha_S(\{\mathsf{Ref}_\ell\ S \mid \ell \in \widehat{\ell}, S \in \{\overline{S_i}\}\}) = \mathsf{Ref}_{\alpha_\ell(\widehat{\ell})}\ \alpha_S(\{\overline{S_i}\}) \sqsubseteq \mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}$.

□

B.2 Auxiliary Definitions

Definition 19 (Gradual label meet)**.**
$\tilde{\ell}_1 \mathbin{\tilde{\curlywedge}} \tilde{\ell}_2 = \alpha_\ell(\{\ell_1 \curlywedge \ell_2 \mid (\ell_1, \ell_2) \in \gamma_\ell(\tilde{\ell}_1) \times \gamma_\ell(\tilde{\ell}_2)\})$.

Algorithmically:

$$\begin{aligned}
\bot \mathbin{\tilde{\curlywedge}} ? = ? \mathbin{\tilde{\curlywedge}} \bot &= \bot \\
\tilde{\ell} \mathbin{\tilde{\curlywedge}} ? = ? \mathbin{\tilde{\curlywedge}} \tilde{\ell} &= ? \text{ if } \tilde{\ell} \neq \bot \\
\ell_1 \mathbin{\tilde{\curlywedge}} \ell_2 &= \ell_1 \curlywedge \ell_2
\end{aligned}$$

$\boxed{\widetilde{S} \mathbin{\tilde{\curlyvee}} \widetilde{S},\ \widetilde{S} \mathbin{\tilde{\curlywedge}} \widetilde{S}}$

$\mathbin{\tilde{\curlyvee}} : \textsc{Type} \times \textsc{Type} \rightharpoonup \textsc{Type}$
$\mathsf{Bool}_{\tilde{\ell}} \mathbin{\tilde{\curlyvee}} \mathsf{Bool}_{\tilde{\ell}'} = \mathsf{Bool}_{(\tilde{\ell} \mathbin{\tilde{\curlyvee}} \tilde{\ell}')}$
$(\widetilde{S}_{11} \xrightarrow{\tilde{\ell}_c}_{\tilde{\ell}} \widetilde{S}_{12}) \mathbin{\tilde{\curlyvee}} (\widetilde{S}_{21} \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}'} \widetilde{S}_{22}) = (\widetilde{S}_{11} \mathbin{\tilde{\curlywedge}} \widetilde{S}_{21}) \xrightarrow{\tilde{\ell}_c \mathbin{\tilde{\curlywedge}} \tilde{\ell}'_c}_{(\tilde{\ell} \mathbin{\tilde{\curlyvee}} \tilde{\ell}')} (\widetilde{S}_{12} \mathbin{\tilde{\curlyvee}} \widetilde{S}_{22})$
$\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S} \mathbin{\tilde{\curlyvee}} \mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S}' = \mathsf{Ref}_{(\tilde{\ell} \mathbin{\tilde{\curlyvee}} \tilde{\ell}')}\ \widetilde{S} \sqcap \widetilde{S}'$
$\widetilde{S} \mathbin{\tilde{\curlyvee}} \widetilde{S}$ undefined otherwise

$\mathbin{\tilde{\curlywedge}} : \textsc{Type} \times \textsc{Type} \rightharpoonup \textsc{Type}$
$\mathsf{Bool}_{\tilde{\ell}} \mathbin{\tilde{\curlywedge}} \mathsf{Bool}_{\tilde{\ell}'} = \mathsf{Bool}_{(\tilde{\ell} \mathbin{\tilde{\curlywedge}} \tilde{\ell}')}$
$(\widetilde{S}_{11} \xrightarrow{\tilde{\ell}_c}_{\tilde{\ell}} \widetilde{S}_{12}) \mathbin{\tilde{\curlywedge}} (\widetilde{S}_{21} \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}'} \widetilde{S}_{22}) = (\widetilde{S}_{11} \mathbin{\tilde{\curlyvee}} \widetilde{S}_{21}) \xrightarrow{\tilde{\ell}_c \mathbin{\tilde{\curlyvee}} \tilde{\ell}'_c}_{(\tilde{\ell} \mathbin{\tilde{\curlywedge}} \tilde{\ell}')} (\widetilde{S}_{12} \mathbin{\tilde{\curlywedge}} \widetilde{S}_{22})$
$\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S} \mathbin{\tilde{\curlywedge}} \mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S}' = \mathsf{Ref}_{(\tilde{\ell} \mathbin{\tilde{\curlywedge}} \tilde{\ell}')}\ \widetilde{S} \sqcap \widetilde{S}'$
$\widetilde{S} \mathbin{\tilde{\curlywedge}} \widetilde{S}$ undefined otherwise

Figure 9. $\text{GSL}_{\mathsf{Ref}}$: consistent join and consistent meet

$\tilde{\ell} \in \textsc{GLabel}, \quad \widetilde{S} \in \textsc{GType}, \quad x \in \textsc{Var}, \quad b \in \textsc{Bool}, \quad \oplus \in \textsc{BoolOp}$
$l \in \textsc{Loc}, \quad t \in \textsc{GTerm}, \quad r \in \textsc{RawValue} \quad v \in \textsc{Value}$
$\Gamma \in \textsc{Var} \xrightarrow{\text{fin}} \textsc{GType}, \quad \Sigma \in \textsc{Loc} \xrightarrow{\text{fin}} \textsc{GType}$

$\widetilde{S}$	::=	$\mathsf{Bool}_{\tilde{\ell}} \mid \widetilde{S} \xrightarrow{\tilde{\ell}_c}_{\tilde{\ell}} \widetilde{S} \mid \mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S} \mid \mathsf{Unit}_{\tilde{\ell}}$	(gradual types)
$\tilde{\ell}$	::=	$\ell \mid ?$	(gradual labels)
b	::=	$\mathsf{true} \mid \mathsf{false}$	(Booleans)
r	::=	$b \mid \lambda^{\tilde{\ell}_c} x : \widetilde{S}.t \mid \mathsf{unit} \mid l$	(base values)
v	::=	$r_{\tilde{\ell}}$	(values)
t	::=	$v \mid t\ t \mid t \oplus t \mid \mathsf{if}\ t\ \mathsf{then}\ t\ \mathsf{else}\ t$	(terms)
		$\mathsf{ref}^{\widetilde{S}}\ t \mid !t \mid t := t \mid \mathsf{prot}_{\tilde{\ell}} t$	
$\oplus$	::=	$\wedge \mid \vee$	(operations)

Figure 10. $\text{GSL}_{\mathsf{Ref}}$: Syntax

Also, we introduce a function *label*, which yields the security label of a given type:

$$label : \textsc{GType} \to \textsc{Label}$$

$$label(\mathsf{Bool}_{\tilde{\ell}}) = \tilde{\ell} \qquad label(\mathsf{Unit}_{\tilde{\ell}}) = \tilde{\ell}$$

$$label(\widetilde{S}_1 \to_{\tilde{\ell}} \widetilde{S}_2) = \tilde{\ell} \qquad label(\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}) = \tilde{\ell}$$

B.3 Syntax and Static Semantics

In this section we present the syntax and static semantics of $\text{GSL}_{\mathsf{Ref}}$. The syntax of $\text{GSL}_{\mathsf{Ref}}$ is given in Figure 10 and is otherwise identical to that of $\text{SSL}_{\mathsf{Ref}}$. Figure 11 presents the type system of $\text{GSL}_{\mathsf{Ref}}$. Each typing rule is derived from a corresponding $\text{SSL}_{\mathsf{Ref}}$ rule (Figure 2) by lifting labels, types, predicates, and functions to their gradual counterparts.

B.4 Static Criteria for Gradual Typing

In this section we present the proof of Static Gradual Guarantee for $\text{GSL}_{\mathsf{Ref}}$.

Proposition 5 (Equivalence for fully-annotated terms)**.** *For any $t \in \textsc{Term}$, $.; \Sigma; \ell_c \vdash_S t : S$ if and only if $.; \Sigma; \ell_c \vdash t : S$*

$\boxed{\Gamma;\Sigma;\tilde{\ell} \vdash t : \widetilde{S}}$

$$(\widetilde{S}\text{x})\ \frac{x : \widetilde{S} \in \Gamma}{\Gamma;\Sigma;\tilde{\ell}_c \vdash x : \widetilde{S}} \qquad (\widetilde{S}\text{b})\ \frac{}{\Gamma;\Sigma;\tilde{\ell}_c \vdash b_{\tilde{\ell}} : \mathsf{Bool}_{\tilde{\ell}}}$$

$$(\widetilde{S}\text{u})\ \frac{}{\Gamma;\Sigma;\tilde{\ell}_c \vdash \mathsf{unit}_{\tilde{\ell}} : \mathsf{Unit}_{\tilde{\ell}}} \qquad (\widetilde{S}\text{l})\ \frac{l : \widetilde{S} \in \Sigma}{\Gamma;\Sigma;\tilde{\ell}_c \vdash l_{\tilde{\ell}} : \mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}}$$

$$(\widetilde{S}\lambda)\ \frac{\Gamma, x : \widetilde{S}_1;\Sigma;\tilde{\ell}'_c \vdash t : \widetilde{S}_2}{\Gamma;\Sigma;\tilde{\ell}_c \vdash (\lambda^{\tilde{\ell}'_c} x : \widetilde{S}_1.t)_{\tilde{\ell}} : \widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_2}$$

$$(\widetilde{S}\text{prot})\ \frac{\Gamma;\Sigma;\tilde{\ell}_c \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \vdash t : \widetilde{S}}{\Gamma;\Sigma;\tilde{\ell}_c \vdash \mathsf{prot}_{\tilde{\ell}} t : \widetilde{S} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}}$$

$$(\widetilde{S}\oplus)\ \frac{\Gamma;\Sigma;\tilde{\ell}_c \vdash t_1 : \mathsf{Bool}_{\tilde{\ell}_1} \qquad \Gamma;\Sigma;\tilde{\ell}_c \vdash t_2 : \mathsf{Bool}_{\tilde{\ell}_2}}{\Gamma;\Sigma;\tilde{\ell}_c \vdash t_1 \oplus t_2 : \mathsf{Bool}_{(\tilde{\ell}_1 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_2)}}$$

$$(\widetilde{S}\text{app})\ \frac{\begin{array}{c}\Gamma;\Sigma;\tilde{\ell}_c \vdash t_1 : \widetilde{S}_{11} \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_{12} \qquad \Gamma;\Sigma;\tilde{\ell}_c \vdash t_2 : \widetilde{S}_2 \\ \widetilde{S}_2 \lesssim \widetilde{S}_{11} \qquad \widetilde{\tilde{\ell} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_c \preccurlyeq \tilde{\ell}'_c}\end{array}}{\Gamma;\Sigma;\tilde{\ell}_c \vdash t_1\ t_2 : \widetilde{S}_{12} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}}$$

$$(\widetilde{S}\text{if})\ \frac{\begin{array}{c}\Gamma;\Sigma;\tilde{\ell}_c \vdash t : \mathsf{Bool}_{\tilde{\ell}} \\ \Gamma;\Sigma;\tilde{\ell}_c \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \vdash t_1 : \widetilde{S}_1 \qquad \Gamma;\Sigma;\tilde{\ell}_c \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \vdash t_2 : \widetilde{S}_2\end{array}}{\Gamma;\Sigma;\tilde{\ell}_c \vdash \mathsf{if}\ t\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 : (\widetilde{S}_1 \mathbin{\widetilde{\vee}} \widetilde{S}_2) \mathbin{\widetilde{\curlyvee}} \tilde{\ell}}$$

$$(\widetilde{S}\text{::})\ \frac{\Gamma;\Sigma;\tilde{\ell}_c \vdash t : \widetilde{S}_1 \qquad \widetilde{S}_1 \lesssim \widetilde{S}_2}{\Gamma;\Sigma;\tilde{\ell}_c \vdash t :: \widetilde{S}_2 : \widetilde{S}_2}$$

$$(\widetilde{S}\text{ref})\ \frac{\Gamma;\Sigma;\tilde{\ell}_c \vdash t : \widetilde{S}' \qquad \widetilde{S}' \lesssim \widetilde{S} \qquad \tilde{\ell}_c \mathbin{\widetilde{\preccurlyeq}} \mathsf{label}(\widetilde{S})}{\Gamma;\Sigma;\tilde{\ell}_c \vdash \mathsf{ref}^{\widetilde{S}}\ t : \mathsf{Ref}_{\perp}\ \widetilde{S}}$$

$$(\widetilde{S}\text{deref})\ \frac{\Gamma;\Sigma;\tilde{\ell}_c \vdash t : \mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}}{\Gamma;\Sigma;\tilde{\ell}_c \vdash !t : \widetilde{S} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}}$$

$$(\widetilde{S}\text{asgn})\ \frac{\begin{array}{c}\Gamma;\Sigma;\tilde{\ell}_c \vdash t_1 : \mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1 \qquad \Gamma;\Sigma;\tilde{\ell}_c \vdash t_2 : \widetilde{S}_2 \\ \widetilde{S}_2 \lesssim \widetilde{S}_1 \qquad \widetilde{\tilde{\ell} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_c \preccurlyeq \mathit{label}(\widetilde{S}_1)}\end{array}}{\Gamma;\Sigma;\tilde{\ell}_c \vdash t_1 := t_2 : \mathsf{Unit}_{\perp}}$$

Figure 11. $\text{GSL}_{\mathsf{Ref}}$: Static Semantics

Proof. By induction over the typing derivations. The proof is trivial because static types are given singleton meanings via concretization. □

Definition 20 (Term precision)**.**

$$(Px)\ \frac{}{x \sqsubseteq x} \qquad (Pb)\ \frac{\tilde{\ell} \sqsubseteq \tilde{\ell}'}{b_{\tilde{\ell}} \sqsubseteq b_{\tilde{\ell}'}} \qquad (Pu)\ \frac{\tilde{\ell} \sqsubseteq \tilde{\ell}'}{\mathsf{unit}_{\tilde{\ell}} \sqsubseteq \mathsf{unit}_{\tilde{\ell}'}}$$

$$(P\lambda)\ \frac{\begin{array}{c}t \sqsubseteq t' \qquad \tilde{\ell}'_c \sqsubseteq \tilde{\ell}''_c \\ \widetilde{S}_1 \sqsubseteq \widetilde{S}'_1 \qquad \tilde{\ell} \sqsubseteq \tilde{\ell}'\end{array}}{(\lambda^{\tilde{\ell}'_c} x : \widetilde{S}_1.t)_{\tilde{\ell}} \sqsubseteq (\lambda^{\tilde{\ell}''_c} x : \widetilde{S}'_1.t')_{\tilde{\ell}'}}$$

$$(Pprot)\ \frac{t \sqsubseteq t' \qquad \tilde{\ell} \sqsubseteq \tilde{\ell}'}{\mathsf{prot}_{\tilde{\ell}} t \sqsubseteq \mathsf{prot}_{\tilde{\ell}'} t'} \qquad (P\oplus)\ \frac{t_1 \sqsubseteq t'_1 \qquad t_2 \sqsubseteq t'_2}{t_1 \oplus t_2 \sqsubseteq t'_1 \oplus t'_2}$$

$$(Papp)\ \frac{t_1 \sqsubseteq t'_1 \qquad t_2 \sqsubseteq t'_2}{t_1\ t_2 \sqsubseteq t'_1\ t'_2}$$

$$(Pif)\ \frac{t \sqsubseteq t \qquad t_1 \sqsubseteq t'_1 \qquad t_2 \sqsubseteq t'_2}{\mathsf{if}\ t\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \sqsubseteq \mathsf{if}\ t'\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2}$$

$$(P::)\ \frac{t \sqsubseteq t' \qquad \widetilde{S} \sqsubseteq \widetilde{S}'}{t :: \widetilde{S} \sqsubseteq t' :: \widetilde{S}'} \qquad (Pref)\ \frac{t \sqsubseteq t' \qquad \widetilde{S} \sqsubseteq \widetilde{S}'}{\mathsf{ref}^{\widetilde{S}}\ t \sqsubseteq \mathsf{ref}^{\widetilde{S}'}\ t'}$$

$$(Pderef)\ \frac{t \sqsubseteq t'}{!t \sqsubseteq !t'} \qquad (Pasgn)\ \frac{t_1 \sqsubseteq t'_1 \qquad t_2 \sqsubseteq t'_2}{t_1 := t_2 \sqsubseteq t'_1 := t'_2}$$

Definition 21 (Type environment precision)**.**

$$\frac{}{\cdot \sqsubseteq \cdot} \qquad \frac{\Gamma \sqsubseteq \Gamma' \qquad \widetilde{S} \sqsubseteq \widetilde{S}'}{\Gamma, x : \widetilde{S} \sqsubseteq \Gamma', x : \widetilde{S}'}$$

Lemma 17. *If* $\Gamma;\cdot;\tilde{\ell}_c \vdash t : \widetilde{S}$ *and* $\Gamma \sqsubseteq \Gamma'$*, then* $\Gamma';\cdot;\tilde{\ell}_c \vdash t : \widetilde{S}'$ *for some* $\widetilde{S} \sqsubseteq \widetilde{S}'$*.*

Proof. Simple induction on typing derivations. □

Lemma 18. *If* $\widetilde{S}_1 <: \widetilde{S}_2$ *and* $\widetilde{S}_1 \sqsubseteq \widetilde{S}'_1$ *and* $\widetilde{S}_2 \sqsubseteq \widetilde{S}'_2$ *then* $\widetilde{S}'_1 <: \widetilde{S}'_2$*.*

Proof. By definition of $<:$, there exists $\langle S_1, S_2\rangle \in \gamma^2(\widetilde{S}_1, \widetilde{S}_2)$ such that $S_1 <: S_2$. $\widetilde{S}_1 \sqsubseteq \widetilde{S}'_1$ and $\widetilde{S}_2 \sqsubseteq \widetilde{S}'_2$ mean that $\gamma(\widetilde{S}_1) \subseteq \gamma(\widetilde{S}'_1)$ and $\gamma(\widetilde{S}_2) \subseteq \gamma(\widetilde{S}'_2)$, therefore $\langle S_1, S_2\rangle \in \gamma^2(\widetilde{S}'_1, \widetilde{S}'_2)$. □

Lemma 19. *If* $\widetilde{\tilde{\ell}_1 \mathbin{\curlyvee} \tilde{\ell}_2 \preccurlyeq \tilde{\ell}_3}$*,* $\tilde{\ell}_1 \sqsubseteq \tilde{\ell}'_1$*,* $\tilde{\ell}_2 \sqsubseteq \tilde{\ell}'_2$ *and* $\tilde{\ell}_3 \sqsubseteq \tilde{\ell}'_3$*, then* $\widetilde{\tilde{\ell}'_1 \mathbin{\curlyvee} \tilde{\ell}'_2 \preccurlyeq \tilde{\ell}'_3}$*.*

Proof. By definition of the consistent judgment, there exists $\langle \ell_1, \ell_2, \ell_3\rangle \in \gamma^3(\tilde{\ell}_1, \tilde{\ell}_2, \tilde{\ell}_3)$ such that $\ell_1 \mathbin{\curlyvee} \ell_2 \preccurlyeq \ell_3$. $\tilde{\ell}_1 \sqsubseteq \tilde{\ell}'_1$, $\tilde{\ell}_2 \sqsubseteq \tilde{\ell}'_2$ and $\tilde{\ell}_3 \sqsubseteq \tilde{\ell}'_3$ mean that $\gamma(\tilde{\ell}_1) \subseteq \gamma(\tilde{\ell}'_1)$, $\gamma(\tilde{\ell}_2) \subseteq \gamma(\tilde{\ell}'_2)$ and $\gamma(\tilde{\ell}_3) \subseteq \gamma(\tilde{\ell}'_3)$ respectively. Therefore $\langle \ell_1, \ell_2, \ell_3\rangle \in \gamma^3(\tilde{\ell}_1, \tilde{\ell}_2, \tilde{\ell}_3)$. □

Lemma 20. *If* $\tilde{\ell}_1 \mathbin{\widetilde{\preccurlyeq}} \tilde{\ell}_2$*,* $\tilde{\ell}_1 \sqsubseteq \tilde{\ell}'_1$ *and* $\tilde{\ell}_2 \sqsubseteq \tilde{\ell}'_2$*, then* $\tilde{\ell}'_1 \mathbin{\widetilde{\preccurlyeq}} \tilde{\ell}'_2$*.*

Proof. Using almost identical argument of Lemma 19 □

Proposition 6 (Static gradual guarantee)**.** *If* $.;.;\tilde{\ell}_{c1} \vdash t_1 : \widetilde{S}_1$*,* $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$ *and* $t_1 \sqsubseteq t_2$*, then* $.;.;\tilde{\ell}_{c2} \vdash t_2 : \widetilde{S}_2$ *and* $\widetilde{S}_1 \sqsubseteq \widetilde{S}_2$*.*

Proof. We prove the property on opens terms instead of closed terms: If $\Gamma;.;\tilde{\ell}_{c1} \vdash t_1 : \widetilde{S}_1$, $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$ and $t_1 \sqsubseteq t_2$ then $\Gamma;.;\tilde{\ell}_{c2} \vdash t_2 : \widetilde{S}_2$ and $\widetilde{S}_1 \sqsubseteq \widetilde{S}_2$.

The proof proceed by induction on the typing derivation.

Case (Sx, Sb, Su). Trivial by definition of $\sqsubseteq$ using $(Px), (Pb), (Pu)$ respectively.

Case (Sλ). Then $t_1 = (\lambda^{\tilde{\ell}'_c} x : \widetilde{S}_1.t)_{\tilde{\ell}}$ and $\widetilde{S}_1 = \widetilde{S}'_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}'_2$. By $(\widetilde{S}\lambda)$ we know that:

$$(\widetilde{S}\lambda)\frac{\Gamma, x : \widetilde{S}'_1; .; \tilde{\ell}'_c \vdash t : \widetilde{S}'_2}{\Gamma; .; \tilde{\ell}_{c1} \vdash (\lambda^{\tilde{\ell}'_c} x : \widetilde{S}'_1.t)_{\tilde{\ell}} : \widetilde{S}'_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}'_2} \quad (1)$$

Consider $\tilde{\ell}_{c2}$ such that $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$ and t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = (\lambda^{\tilde{\ell}'_c} x : \widetilde{S}''_1.t')'_{\tilde{\ell}}$ and therefore

$$(\widetilde{S}\lambda)\frac{\begin{array}{c} t \sqsubseteq t' \\ \tilde{\ell}'_c \sqsubseteq \tilde{\ell}''_c \qquad \widetilde{S}'_1 \sqsubseteq \widetilde{S}''_1 \qquad \tilde{\ell} \sqsubseteq \tilde{\ell}' \end{array}}{(\lambda^{\tilde{\ell}'_c} x : \widetilde{S}'_1.t)_{\tilde{\ell}} \sqsubseteq (\lambda^{\tilde{\ell}''_c} x : \widetilde{S}''_1.t')_{\tilde{\ell}'}} \quad (2)$$

Using induction hypotheses on the premise of 1, $\Gamma, x : \widetilde{S}'_1; .; \tilde{\ell}_{c2} \vdash t' : \widetilde{S}''_2$ with $\widetilde{S}'_2 \sqsubseteq \widetilde{S}''_2$. By Lemma 17, $\Gamma, x : \widetilde{S}'_1; .; \tilde{\ell}_{c2} \vdash t' : \widetilde{S}'''_2$ where $\widetilde{S}''_2 \sqsubseteq \widetilde{S}'''_2$. Then we can use rule $(\widetilde{S}\lambda)$ to derive:

$$(\widetilde{S}\lambda)\frac{\Gamma, x : \widetilde{S}''_1; .; \tilde{\ell}''_c \vdash t' : \widetilde{S}'''_2}{\Gamma; .; \tilde{\ell}_{c1} \vdash (\lambda^{\tilde{\ell}''_c} x : \widetilde{S}'_1.t')_{\tilde{\ell}'} : \widetilde{S}''_1 \xrightarrow{\tilde{\ell}''_c}_{\tilde{\ell}'} \widetilde{S}'''_2}$$

Where $\widetilde{S}_2 \sqsubseteq \widetilde{S}''_2$. Using the premise of 2 and the definition of type precision we can infer that

$$\widetilde{S}'_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}'_2 \sqsubseteq \widetilde{S}''_1 \xrightarrow{\tilde{\ell}''_c}_{\tilde{\ell}'} \widetilde{S}'''_2$$

and the result holds.

Case $(\widetilde{S}\text{l})$. This case can not happen because initial programs do not contain locations.

Case $(\widetilde{S}\text{prot})$. Then $t_1 = \mathsf{prot}_{\tilde{\ell}} t$ and $\widetilde{S}_1 = \widetilde{S} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}$. By $(\widetilde{S}\text{prot})$ we know that:

$$(\widetilde{S}\text{prot})\frac{\Gamma; .; \tilde{\ell}_{c1} \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \vdash t : \widetilde{S}}{\Gamma; .; \tilde{\ell}_{c1} \vdash \mathsf{prot}_{\tilde{\ell}} t : \widetilde{S} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}} \quad (3)$$

Consider $\tilde{\ell}_{c2}$ such that $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$ and t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = \mathsf{prot}_{\tilde{\ell}'} t'$ and therefore

$$(\text{Pprot})\frac{t \sqsubseteq t' \qquad \tilde{\ell} \sqsubseteq \tilde{\ell}'}{\mathsf{prot}_{\tilde{\ell}} t \sqsubseteq \mathsf{prot}_{\tilde{\ell}'} t'} \quad (4)$$

By definition of join on consistent labels, $\tilde{\ell}_{c1} \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \sqsubseteq \tilde{\ell}_{c2} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'$. Using induction hypotheses on the premises of 3, we can use rule $(\widetilde{S}\text{prot})$ to derive:

$$(\widetilde{S}\text{prot})\frac{\Gamma; .; \tilde{\ell}_{c2} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}' \vdash t' : \widetilde{S}'}{\Gamma; .; \tilde{\ell}_{c2} \vdash \mathsf{prot}_{\tilde{\ell}'} t' : \widetilde{S}' \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'}$$

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

$$\widetilde{S} \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \sqsubseteq \widetilde{S}' \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'$$

and the result holds.

Case $(\widetilde{S}\oplus)$. Then $t_1 = t'_1 \oplus t'_2$ and $\widetilde{S}_1 = \mathsf{Bool}_{(\tilde{\ell}_1 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_2)}$. By $(\widetilde{S}\oplus)$ we know that:

$$(\widetilde{S}\oplus)\frac{\Gamma; .; \tilde{\ell}_{c1} \vdash t'_1 : \mathsf{Bool}_{\tilde{\ell}_1} \qquad \Gamma; .; \tilde{\ell}_{c1} \vdash t'_2 : \mathsf{Bool}_{\tilde{\ell}_2}}{\Gamma; .; \tilde{\ell}_{c1} \vdash t'_1 \oplus t'_2 : \mathsf{Bool}_{(\tilde{\ell}_1 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_2)}} \quad (5)$$

Consider $\tilde{\ell}_{c2}$ such that $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$ 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

$$(\text{P}\oplus)\frac{t'_1 \sqsubseteq t''_1 \qquad t'_2 \sqsubseteq t''_2}{t'_1 \oplus t'_2 \sqsubseteq t''_1 \oplus t''_2} \quad (6)$$

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

$$(\widetilde{S}\oplus)\frac{\Gamma; .; \tilde{\ell}_{c2} \vdash t''_1 : \mathsf{Bool}_{\tilde{\ell}'_1} \qquad \Gamma; .; \tilde{\ell}_{c2} \vdash t''_2 : \mathsf{Bool}_{\tilde{\ell}'_2}}{\Gamma; .; \tilde{\ell}_{c2} \vdash t''_1 \oplus t''_2 : \mathsf{Bool}_{(\tilde{\ell}''_1 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}''_2)}}$$

Where $\tilde{\ell}'_1 \sqsubseteq \tilde{\ell}''_1$ and $\tilde{\ell}'_2 \sqsubseteq \tilde{\ell}''_2$. Using the premise of 6 and the definition of type precision we can infer that

$$\frac{(\tilde{\ell}'_1 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'_2) \sqsubseteq (\tilde{\ell}''_1 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}''_2)}{\mathsf{Bool}_{(\tilde{\ell}'_1 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'_2)} \sqsubseteq \mathsf{Bool}_{(\tilde{\ell}''_1 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}''_2)}}$$

and the result holds.

Case $(\widetilde{S}\text{app})$. Then $t_1 = t'_1\ t'_2$ and $\widetilde{S}_1 = \widetilde{S}_{12} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}$. By $(\widetilde{S}\text{app})$ we know that:

$$(\widetilde{S}\text{app})\frac{\begin{array}{cc} \Gamma; .; \tilde{\ell}_{c1} \vdash t'_1 : \widetilde{S}_{11} \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_{12} & \Gamma; .; \tilde{\ell}_{c1} \vdash t'_2 : \widetilde{S}'_2 \\ \widetilde{S}'_2 \lesssim \widetilde{S}_{11} & \widetilde{\tilde{\ell} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_{c1}} \preccurlyeq \tilde{\ell}'_c \end{array}}{\Gamma; .; \tilde{\ell}_{c1} \vdash t'_1\ t'_2 : \widetilde{S}_{12} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}} \quad (7)$$

Consider $\tilde{\ell}_{c2}$ such that $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$ 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

$$(\text{Papp})\frac{t'_1 \sqsubseteq t''_1 \qquad t'_2 \sqsubseteq t''_2}{t'_1\ t'_2 \sqsubseteq t''_1\ t''_2} \quad (8)$$

Using induction hypotheses on the premises of 7, $\Gamma; .; \tilde{\ell}_{c2} \vdash t''_1 : \widetilde{S}'_{11} \xrightarrow{\tilde{\ell}''_c}_{\tilde{\ell}'} \widetilde{S}'_{12}$ and $\Gamma; .; \tilde{\ell}_{c2} \vdash t''_2 : \widetilde{S}''_2$, where $\widetilde{S}'_2 \sqsubseteq \widetilde{S}''_2$, $\widetilde{S}_{11} \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_{12} \sqsubseteq \widetilde{S}'_{11} \xrightarrow{\tilde{\ell}''_c}_{\tilde{\ell}'} \widetilde{S}'_{12}$. By Lemma 18, $\widetilde{S}''_2 \lesssim \widetilde{S}'_{11}$. By definition of precision of types, $\tilde{\ell}'_c \sqsubseteq \tilde{\ell}''_c$ and $\tilde{\ell} \sqsubseteq \tilde{\ell}'$, therefore by Lemma 19, $\widetilde{\tilde{\ell}' \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_{c2}} \preccurlyeq \tilde{\ell}''_c$. Then we can use rule $(\widetilde{S}\text{app})$ to derive:

$$(\widetilde{S}\text{app})\frac{\begin{array}{cc} \Gamma; .; \tilde{\ell}_{c2} \vdash t''_1 : \widetilde{S}'_{11} \xrightarrow{\tilde{\ell}''_c}_{\tilde{\ell}'} \widetilde{S}'_{12} & \Gamma; .; \tilde{\ell}_{c2} \vdash t''_2 : \widetilde{S}''_2 \\ \widetilde{S}''_2 \lesssim \widetilde{S}'_{11} & \widetilde{\tilde{\ell}' \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_{c2}} \preccurlyeq \tilde{\ell}''_c \end{array}}{\Gamma; .; \tilde{\ell}_{c2} \vdash t''_1\ t''_2 : \widetilde{S}'_{12} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'}$$

Using the definition of type precision we can infer that

$$\widetilde{S}_{12} \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \sqsubseteq \widetilde{S}'_{12} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'$$

and the result holds.

Case $(\widetilde{S}\text{if})$. Then $t_1 = \mathsf{if}\ t\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2$ and $\widetilde{S}_1 = (\widetilde{S}_1 \mathbin{\widetilde{\vee}} \widetilde{S}_2) \mathbin{\widetilde{\curlyvee}} \tilde{\ell}$. By $(\widetilde{S}\text{if})$ we know that:

$$(\widetilde{S}\text{if})\frac{\begin{array}{c} \Gamma; .; \tilde{\ell}_{c1} \vdash t : \mathsf{Bool}_{\tilde{\ell}} \\ \Gamma; .; \tilde{\ell}_{c1} \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \vdash t_1 : \widetilde{S}_1 \qquad \Gamma; .; \tilde{\ell}_{c1} \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \vdash t_2 : \widetilde{S}_2 \end{array}}{\Gamma; .; \tilde{\ell}_{c1} \vdash \mathsf{if}\ t\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 : (\widetilde{S}_1 \mathbin{\widetilde{\vee}} \widetilde{S}_2) \mathbin{\widetilde{\curlyvee}} \tilde{\ell}} \quad (9)$$

Consider $\tilde{\ell}_{c2}$ such that $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$ and t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = \mathsf{if}\ t'\ \mathsf{then}\ t'_1\ \mathsf{else}\ t'_2$ and therefore

$$(\text{Pif})\frac{t \sqsubseteq t \qquad t_1 \sqsubseteq t'_1 \qquad t_2 \sqsubseteq t'_2}{\mathsf{if}\ t\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \sqsubseteq \mathsf{if}\ t'\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2} \quad (10)$$

Consider any ℓ' such that $\ell \sqsubseteq \ell'$. As $\tilde{\ell}_{c1} \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \sqsubseteq \tilde{\ell}_{c2} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'$ then we can use induction hypotheses on the premises of 9 and derive:

$$(\widetilde{S}\text{if})\frac{\Gamma;.;\tilde{\ell}_{c2} \vdash t' : \mathsf{Bool}_{\tilde{\ell}'} \quad \Gamma;.;\tilde{\ell}_{c2} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}' \vdash t'_1 : \widetilde{S}'_1 \quad \Gamma;.;\tilde{\ell}_{c2} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}' \vdash t'_2 : \widetilde{S}'_2}{\Gamma;.;\tilde{\ell}_{c2} \vdash \mathsf{if}\ t'\ \mathsf{then}\ t'_1\ \mathsf{else}\ t'_2 : (\widetilde{S}'_1 \mathbin{\widetilde{\vee}} \widetilde{S}'_2) \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'}$$

Where $\widetilde{S}_1 \sqsubseteq \widetilde{S}'_1$ and $\widetilde{S}_2 \sqsubseteq \widetilde{S}'_2$. Using the definition of type precision we can infer that

$$(\widetilde{S}_1 \mathbin{\widetilde{\vee}} \widetilde{S}_2) \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \sqsubseteq (\widetilde{S}'_1 \mathbin{\widetilde{\vee}} \widetilde{S}'_2) \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'$$

and the result holds.

Case ($\widetilde{S}$::). Then $t_1 = t :: \widetilde{S}_2$ and $\widetilde{S}_1 = \widetilde{S}_2$. By ($\widetilde{S}$::) we know that:

$$(\widetilde{S}::)\frac{\Gamma;.;\tilde{\ell}_{c1} \vdash t : \widetilde{S}_1 \quad \widetilde{S}_1 \lesssim \widetilde{S}_2}{\Gamma;.;\tilde{\ell}_{c1} \vdash t :: \widetilde{S}_2 : \widetilde{S}_2} \tag{11}$$

Consider $\tilde{\ell}_{c2}$ such that $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$ and t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = t' :: \widetilde{S}'_2$ and therefore

$$(\text{P::})\frac{t \sqsubseteq t' \quad \widetilde{S} \sqsubseteq \widetilde{S}'}{t :: \widetilde{S} \sqsubseteq t' :: \widetilde{S}'} \tag{12}$$

Using induction hypotheses on the premises of 11, $\Gamma;.;\tilde{\ell}_c \vdash t' : \widetilde{S}'_1$ where $\widetilde{S}'_1 \sqsubseteq \widetilde{S}_1$. We can use rule ($\widetilde{S}$::) and Lemma 18 to derive:

$$(\widetilde{S}::)\frac{\Gamma;.;\tilde{\ell}_{c2} \vdash t' : \widetilde{S}'_1 \quad \widetilde{S}'_1 \lesssim \widetilde{S}'_2}{\Gamma;.;\tilde{\ell}_{c2} \vdash t' :: \widetilde{S}'_2 : \widetilde{S}'_2}$$

Where $\widetilde{S}_2 \sqsubseteq \widetilde{S}'_2$ and the result holds.

Case ($\widetilde{S}$ref). Then $t_1 = \mathsf{ref}^{\widetilde{S}}\ t$ and $\widetilde{S}_1 = \mathsf{Ref}_{\tilde{\ell}_c}\ \widetilde{S}$. By ($\widetilde{S}$ref) we know that:

$$(\widetilde{S}\text{ref})\frac{\Gamma;.;\tilde{\ell}_{c1} \vdash t : \widetilde{S}' \quad \widetilde{S}' \lesssim \widetilde{S} \quad \tilde{\ell}_{c1} \mathbin{\widetilde{\preccurlyeq}} \mathsf{label}(\widetilde{S})}{\Gamma;.;\tilde{\ell}_{c1} \vdash \mathsf{ref}^{\widetilde{S}}\ t : \mathsf{Ref}_{\perp}\ \widetilde{S}} \tag{13}$$

Consider $\tilde{\ell}_{c2}$ such that $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$ and t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = \mathsf{ref}^{\widetilde{S}'}\ t$ and therefore

$$(\text{Pref})\frac{t \sqsubseteq t' \quad \widetilde{S} \sqsubseteq \widetilde{S}'}{\mathsf{ref}^{\widetilde{S}}\ t \sqsubseteq \mathsf{ref}^{\widetilde{S}'}\ t'} \tag{14}$$

Using induction hypotheses on the premises of 13, we can use rule ($\widetilde{S}$ref) and Lemma 18 and 20 to derive:

$$(\widetilde{S}\text{ref})\frac{\Gamma;.;\tilde{\ell}_{c2} \vdash t' : \widetilde{S}'' \quad \widetilde{S}'' \lesssim \widetilde{S}' \quad \tilde{\ell}_{c2} \mathbin{\widetilde{\preccurlyeq}} \mathsf{label}(\widetilde{S}')}{\Gamma;.;\tilde{\ell}_{c2} \vdash \mathsf{ref}^{\widetilde{S}'}\ t' : \mathsf{Ref}_{\perp}\ \widetilde{S}'}$$

Where $\widetilde{S} \sqsubseteq \widetilde{S}$ and $\widetilde{S}' \sqsubseteq \widetilde{S}''$. Using the the definition of type precision we can infer that

$$\frac{\widetilde{S} \sqsubseteq \widetilde{S}'}{\mathsf{Ref}_{\perp}\ \widetilde{S} \sqsubseteq \mathsf{Ref}_{\perp}\ \widetilde{S}'}$$

and the result holds.

Case ($\widetilde{S}$deref). Then $t_1 = !t$ and $\widetilde{S}_1 =$. By ($\widetilde{S}$deref) we know that:

$$(\widetilde{S}\text{deref})\frac{\Gamma;.;\tilde{\ell}_{c1} \vdash t : \mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}}{\Gamma;.;\tilde{\ell}_{c1} \vdash !t : \widetilde{S} \mathbin{\widetilde{\curlyvee}} \tilde{\ell}} \tag{15}$$

Consider $\tilde{\ell}_{c2}$ such that $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$ and t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = !t'$ and therefore

$$(\text{Pderef})\frac{t \sqsubseteq t'}{!t \sqsubseteq !t'} \tag{16}$$

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

$$(\widetilde{S}\text{deref})\frac{\Gamma;.;\tilde{\ell}_{c2} \vdash t' : \mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S}'}{\Gamma;.;\tilde{\ell}_{c2} \vdash !t' : \widetilde{S}' \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'}$$

Where $\tilde{\ell} \sqsubseteq \tilde{\ell}'$ and $\widetilde{S} \sqsubseteq \widetilde{S}'$. Using the premise of 16 and the definition of type precision we can infer that

$$\widetilde{S} \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \sqsubseteq \widetilde{S}' \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'$$

and the result holds.

Case ($\widetilde{S}$asgn). Then $t_1 = t'_1 := t'_2$ and $\widetilde{S}_1 =$. By ($\widetilde{S}$asgn) we know that:

$$(\widetilde{S}\text{asgn})\frac{\Gamma;.;\tilde{\ell}_{c1} \vdash t'_1 : \mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1 \quad \Gamma;.;\tilde{\ell}_{c1} \vdash t'_2 : \widetilde{S}_2 \quad \widetilde{S}_2 \lesssim \widetilde{S}_1 \quad \tilde{\ell} \mathbin{\curlyvee} \tilde{\ell}_{c1} \preccurlyeq \mathit{label}(\widetilde{S}_1)}{\Gamma;.;\tilde{\ell}_{c1} \vdash t'_1 := t'_2 : \mathsf{Unit}_{\perp}} \tag{17}$$

Consider $\tilde{\ell}_{c2}$ such that $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$ 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

$$(\text{Pasgn})\frac{t'_1 \sqsubseteq t''_1 \quad t'_2 \sqsubseteq t''_2}{t'_1 := t'_2 \sqsubseteq t''_1 := t''_2} \tag{18}$$

Using induction hypotheses on the premises of 17, $\Gamma;.;\tilde{\ell}_{c2} \vdash t'_1 : \mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S}'_1$ and $\Gamma;.;\tilde{\ell}_{c2} \vdash t'_2 : \widetilde{S}'_2$, where $\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1 \sqsubseteq \mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S}'_1$ and $\widetilde{S}_2 \sqsubseteq \widetilde{S}_2$. By definition of precision on types and Lemma 18, $\widetilde{S}'_2 \sqsubseteq \widetilde{S}'_1$. Also, as, $\tilde{\ell} \sqsubseteq \tilde{\ell}'$ and $\widetilde{S}_1 \sqsubseteq \widetilde{S}'_1$, by Lemma 19, $\tilde{\ell}' \mathbin{\curlyvee} \tilde{\ell}_{c2} \preccurlyeq \mathit{label}(\widetilde{S}'_1)$. Then we can use rule ($\widetilde{S}$asgn) to derive:

$$(\widetilde{S}\text{asgn})\frac{\Gamma;.;\tilde{\ell}_{c2} \vdash t'_1 : \mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S}'_1 \quad \Gamma;.;\tilde{\ell}_{c2} \vdash t'_2 : \widetilde{S}'_2 \quad \widetilde{S}'_2 \lesssim \widetilde{S}'_1 \quad \tilde{\ell}' \mathbin{\curlyvee} \tilde{\ell}_{c2} \preccurlyeq \mathit{label}(\widetilde{S}'_1)}{\Gamma;.;\tilde{\ell}_{c2} \vdash t''_1 := t''_2 : \mathsf{Unit}_{\perp}}$$

Using the definition of type precision we can infer that

$$\mathsf{Unit}_{\perp} \sqsubseteq \mathsf{Unit}_{\perp}$$

and the result holds.

□

C. Gradualizing the Dynamic Semantics

In this section we present the formalization of the evidences for $\text{GSL}_{\mathsf{Ref}}$. Section presents the structure of evidence and the abstraction and concretization functions. In section C.2, we show how to calculate the initial evidence. In particular we give definition for the interior of consistent judgments for labels and types. In section C.2, we present how to evolve evidence. We define the consistent transitivity operator, the meet operator and join of evidences. In section C.4, we present the algorithmic definitions of interior and consistent transitivity. Finally, in section C.5, we present some of the proofs of the propositions for evidence presented.

C.1 Precise Evidence for Consistent Security Judgments

Definition 22 (Interval)**.** *An interval is a bounded unknown label* $[\ell_1, \ell_2]$ *where* ℓ_1 *is the upper bound and* ℓ_2 *is the lower bound.*

$$\begin{array}{lcll} \imath \in \text{LABEL}^2 & & & \\ \imath & ::= & [\ell, \ell] & \textit{(interval)} \end{array}$$

Definition 23 (Interval Concretization). *Let* $\gamma_\imath : \textsc{Label}^2 \to \mathcal{P}(\textsc{Label})$ *be defined as follows:*

$$\gamma_\imath([\ell_1, \ell_2]) = \{\ell \mid \ell \in \textsc{Label}, \ell_1 \preccurlyeq \ell \preccurlyeq \ell_2\}$$

We can only concretize *valid* intervals:

Definition 24 (Valid Gradual Label)**.**

$$\frac{\ell_1 \preccurlyeq \ell_2}{valid([\ell_1, \ell_2])}$$

Definition 25 (Label Evidence Concretization)**.** *Let* $\gamma_{\varepsilon_\ell} : \textsc{Label}^4 \to \mathcal{P}(\textsc{Label}^2)$ *be defined as follows:*

$$\gamma_{\varepsilon_\ell}(\langle \imath_1, \imath_2 \rangle) = \{\langle \ell_1, \ell_2 \rangle \mid \ell_1 \in \gamma_\imath(\imath_1), \ell_2 \in \gamma_\imath(\imath_2)\}$$

Definition 26 (Interval Abstraction)**.** *Let* $\alpha_\ell : \mathcal{P}(\textsc{Label}) \to \textsc{Label}^2$ *be defined as follows:*

$$\alpha_\imath(\emptyset) \textit{ is undefined}$$
$$\alpha_\imath(\{\overline{\ell_i}\}) = [\lambda\overline{\ell_i}, \gamma\overline{\ell_i}] \textit{ otherwise}$$

Definition 27 (Label Evidence Abstraction)**.** *Let* $\alpha_{\varepsilon_\ell} : \mathcal{P}(\textsc{Label}^2) \to \textsc{Label}^4$ *be defined as follows:*

$$\alpha_{\varepsilon_\ell}(\emptyset) \textit{ is undefined}$$
$$\alpha_{\varepsilon_\ell}(\{\overline{\langle \ell_{1i}, \ell_{2i} \rangle}\}) = \langle \alpha_\imath(\{\overline{\ell_{1i}}\}), \alpha_\imath(\{\overline{\ell_{2i}}\}) \rangle \textit{ otherwise}$$

Definition 28 (Type Evidence)**.** *An evidence type is a gradual type labeled with an interval:*

$$E \in \textsc{GEType}, \quad \imath \in \textsc{Label}^2$$
$$E \quad ::= \quad \mathsf{Bool}_\imath \mid E \xrightarrow{\imath}_\imath E \mid \mathsf{Ref}_\imath\ E \mid \mathsf{Unit}_\imath \quad \textit{(evidence types)}$$

Definition 29 (Type Evidence Concretization)**.** *Let* $\gamma_E : \textsc{GEType} \to \mathcal{P}(\textsc{Type})$ *be defined as follows:*

$$\gamma_E(\mathsf{Bool}_\imath) = \{\, \mathsf{Bool}_\ell \mid \ell \in \gamma_\imath(\imath) \,\}$$
$$\gamma_E(E_1 \xrightarrow{\imath_2}_{\imath_1} E_2) = \gamma_E(E_1)^{\gamma_\imath(\imath_2)}_{\gamma_\imath(\imath_1)\gamma_E(E_2)}$$
$$\gamma_E(\mathsf{Ref}_\imath\ E) = \{\, \mathsf{Ref}_\ell\ S \mid \ell \in \gamma_\imath(\imath), S \in \gamma_E(E) \,\}$$

Definition 30 (Evidence Concretization)**.** *Let* $\gamma_{\varepsilon_\ell} : \textsc{GEType}^2 \to \mathcal{P}(\textsc{Type}^2)$ *be defined as follows:*

$$\gamma_{\varepsilon_\ell}(\langle E_1, E_2 \rangle) = \{\langle S_1, S_2 \rangle \mid S_1 \in \gamma_E(E_1), S_2 \in \gamma_E(E_2)\}$$

Definition 31 (Type Evidence Abstraction)**.** *Let the abstraction function* $\alpha_E : \mathcal{P}(\textsc{Type}) \to \textsc{GEType}$ *be defined as:*

$$\alpha_E(\{\overline{\mathsf{Bool}_{\ell_i}}\}) = \mathsf{Bool}_{\alpha_\imath(\{\overline{\ell_i}\})}$$
$$\alpha_E(\{\overline{S_{i1} \xrightarrow{\ell_{ci}}_{\ell_i} S_{i2}}\}) = \alpha_E(\{\overline{S_{i1}}\})^{\alpha_\imath(\{\overline{\ell_{ci}}\})}_{\alpha_\imath(\{\overline{\ell_i}\})} \alpha_E(\{\overline{S_{i2}}\})$$
$$\alpha_E(\{\overline{\mathsf{Ref}_{\ell_i}\ S_i}\}) = \mathsf{Ref}_{\alpha_\imath(\{\overline{\ell_i}\})}\ \alpha_E(\{\overline{S_i}\})$$
$$\alpha_E(\widehat{S}) \textit{ is undefined otherwise}$$

Definition 32 (Evidence Abstraction)**.** *Let* $\alpha_\varepsilon : \mathcal{P}(\textsc{Type}^2) \to \textsc{GEType}^2$ *be defined as follows:*

$$\alpha_\varepsilon(\emptyset) \textit{ is undefined}$$
$$\alpha_\varepsilon(\{\overline{\langle S_{1i}, S_{2i} \rangle}\}) = \langle \alpha_E(\{\overline{S_{1i}}\}), \alpha_E(\{\overline{S_{2i}}\}) \rangle \textit{ otherwise}$$

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

Definition 33 (Valid Type Sets)**.**

$$\frac{}{valid(\{\overline{\mathsf{Bool}_{\ell_i}}\})} \qquad \frac{valid(\{\overline{S_{i1}}\}) \qquad valid(\{\overline{S_{i2}}\})}{valid(\{\overline{S_{i1} \xrightarrow{\ell_{ci}}_{\ell_i} S_{i2}}\})}$$

$$\frac{valid(\{\overline{S_i}\})}{valid(\{\overline{\mathsf{Ref}_{\ell_i}\ S_i}\})} \qquad \frac{}{valid(\{\overline{\mathsf{Unit}_{\ell_i}}\})}$$

Proposition 21 ($\alpha_\imath$ is Sound)**.** *If* $\widehat{\ell}$ *is not empty, then* $\widehat{\ell} \subseteq \gamma_\imath(\alpha_\imath(\widehat{\ell}))$.

Proposition 22 ($\alpha_\imath$ is Optimal)**.** *If* $\widehat{\ell} \subseteq \gamma_\imath(\imath)$ *then* $\alpha_\imath(\widehat{\ell}) \sqsubseteq \imath$.

Proposition 23 (α_E is Sound)**.** *If* $valid(\widehat{S})$ *then* $\widehat{S} \subseteq \gamma_E(\alpha_E(\widehat{S}))$.

Proposition 24 (α_E is Optimal)**.** *If* $valid(\widehat{S})$ *and* $\widehat{S} \subseteq \gamma_E(E)$ *then* $\alpha_E(\widehat{S}) \sqsubseteq E$.

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

Definition 34 (Interval and Type Evidence Precision)**.**

1. $\imath_1$ *is less imprecise than* $\imath_2$*, notation* $\imath_1 \sqsubseteq \imath_2$*, if and only if* $\gamma_{\varepsilon_\ell}(\imath_1) \subseteq \gamma_{\varepsilon_\ell}(\imath_2)$*; inductively:*

$$\frac{\ell_3 \preccurlyeq \ell_1 \qquad \ell_2 \preccurlyeq \ell_4}{[\ell_1, \ell_2] \sqsubseteq [\ell_3, \ell_4]}$$

2. E_1 *is less imprecise than* E_2*, notation* $E_1 \sqsubseteq E_2$*, if and only if* $\gamma_E(E_1) \subseteq \gamma_E(E_2)$*; inductively:*

$$\frac{\imath_1 \sqsubseteq \imath_2}{\mathsf{Bool}_{\imath_1} \sqsubseteq \mathsf{Bool}_{\imath_2}} \qquad \frac{E_{11} \sqsubseteq E_{21} \qquad E_{12} \sqsubseteq E_{22} \qquad \imath_1 \sqsubseteq \imath_2 \qquad \imath_1' \sqsubseteq \imath_2'}{E_{11} \xrightarrow{\imath_1'}_{\imath_1} E_{12} \sqsubseteq E_{21} \xrightarrow{\imath_2'}_{\imath_2} E_{22}}$$

$$\frac{\imath_1 \sqsubseteq \imath_2 \qquad E_1 \sqsubseteq E_2}{\mathsf{Ref}_{\imath_1}\ E_1 \sqsubseteq \mathsf{Ref}_{\imath_2}\ E_2}$$

C.2 Initial evidence

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

Definition 35 (Interior of label ordering)**.** *Let* $F_1 : \textsc{Label}^n \longrightarrow \textsc{Label}$ *and* $F_2 : \textsc{Label}^m \longrightarrow \textsc{Label}$ *be functions over labels. The interior of the judgment* $\overbrace{F_1(\overline{\tilde{\ell}_i) \preccurlyeq F_2(\overline{\tilde{\ell}_j})}}$*, notation* $\mathcal{I}^{F_1,F_2}_{\preccurlyeq}(\overline{\tilde{\ell}_i}, \overline{\tilde{\ell}_j})$*, is defined as follows:*

$$\mathcal{I}^{F_1,F_2}_{\preccurlyeq}(\tilde{\ell}_1, ...\tilde{\ell}_n, \tilde{\ell}_{n+1}, ...\tilde{\ell}_{n+m}) =$$
$$\alpha_{\varepsilon_\ell}(\{\langle F_1(\overline{\ell_i}), F_2(\overline{\ell_j}) \rangle \mid \langle \overline{\ell_i} \rangle \in \gamma^n_\ell(\overline{\tilde{\ell}_{i[1/n]}}),$$
$$\langle \overline{\ell_j} \rangle \in \gamma^m_\ell(\overline{\tilde{\ell}_{i[n+1/m]}}) \mid F_1(\overline{\ell_i}) \preccurlyeq F_2(\overline{\ell_j})\})$$

Suppose $F_1 = F_{11}$

Definition 36 (Interior of subtyping)**.** *Let* $F_1 : \textsc{Type}^n \longrightarrow \textsc{Type}$ *and* $F_2 : \textsc{Type}^m \longrightarrow \textsc{Type}$ *be functions over types. The interior of the judgment* $\overbrace{F_1(\overline{\widetilde{S}_i) \preccurlyeq F_2(\overline{\widetilde{S}_j})}}$*, notation* $\mathcal{I}^{F_1,F_2}_{<:}(\overline{\widetilde{S}_i}, \overline{\widetilde{S}_j})$*, is defined as follows:*

$$\mathcal{I}^{F_1,F_2}_{<:}(\widetilde{S}_1, ...\widetilde{S}_n, \widetilde{S}_{n+1}, ...\widetilde{S}_{n+m}) =$$
$$\alpha_{\varepsilon_\ell}(\{\langle F_1(\overline{S_i}), F_2(\overline{S_j}) \rangle \mid \langle \overline{S_i} \rangle \in \gamma^n_S(\overline{\widetilde{S}_{i[1/n]}}),$$
$$\langle \overline{S_j} \rangle \in \gamma^m_S(\overline{\widetilde{S}_{i[n+1/m]}}) \mid F_1(\overline{S_i}) <: F_2(\overline{S_j})\})$$

C.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, $\circ^{\preccurlyeq}$ and $\circ^{<:}$ respectively, to combine evidences as follows:

Definition 37 (Consistent transitivity for label ordering)**.** *Suppose*

$$\overbrace{\langle \imath_{11}, \imath_{12}\rangle \vdash F_1(\overline{\tilde{\ell}_i}) \preccurlyeq F_2(\overline{\tilde{\ell}_j})} \qquad \overbrace{\langle \imath_{21}, \imath_{22}\rangle \vdash F_2(\overline{\tilde{\ell}_j}) \preccurlyeq F_3(\overline{\tilde{\ell}_k})}$$

We deduce evidence for consistent transitivity for label ordering:

$$\overbrace{\langle \imath_{11}, \imath_{12}\rangle \circ^{\preccurlyeq} \langle \imath_{21}, \imath_{22}\rangle \vdash F_1(\overline{\tilde{\ell}_i}) \preccurlyeq F_3(\overline{\tilde{\ell}_k})}$$

where $\circ^{\preccurlyeq} : \textsc{Label}^2 \times \textsc{Label}^2 \to \textsc{Label}^2$ *is defined as:*

$$\begin{aligned}\langle \imath_{11}, \imath_{12}\rangle \circ^{\preccurlyeq} \langle \imath_{21}, \imath_{22}\rangle = \\ \alpha_{\varepsilon_\ell}(\{\langle \ell_{11}, \ell_{22}\rangle \in \gamma_{\varepsilon_\ell}(\langle \imath_{11}, \imath_{22}\rangle) \mid \\ \exists \ell \in \gamma_\imath(\imath_{12}) \cap \gamma_\imath(\imath_{21}).\ell_{11} \preccurlyeq \ell \wedge \ell \preccurlyeq \ell_{22}\})\end{aligned}$$

Proposition 25. $\gamma_\imath(\imath_1 \sqcap \imath_2) = \gamma_\imath(\imath_1) \cap \gamma_\imath(\imath_2)$.

where $\imath \sqcap \imath' = \alpha_\ell(\gamma_\ell(\imath) \cap \gamma_\ell(\imath'))$.

Proposition 26.

$$\langle \imath_1, \imath_{21}\rangle \circ^{\preccurlyeq} \langle \imath_{22}, \imath_3\rangle = \triangle^{\preccurlyeq}(\imath_1, \imath_{21} \sqcap \imath_{22}, \imath_3)$$

where

$$\begin{aligned}\triangle^{\preccurlyeq}(\imath_1, \imath_2, \imath_3) = \alpha_\varepsilon(\{\langle \ell_1, \ell_3\rangle \in \gamma_\varepsilon(\langle \imath_1, \imath_3\rangle) \mid \\ \exists \ell_2 \in \gamma_\imath(\imath_2).\ell_1 \preccurlyeq \ell_2 \wedge \ell_2 \preccurlyeq \ell_3\})\end{aligned}$$

Definition 38 (Consistent transitivity for subtyping)**.** *Suppose*

$$\overbrace{\langle E_{11}, E_{12}\rangle \vdash F_1(\overline{\tilde{S}_i}) <: F_2(\overline{\tilde{S}_j})}$$

$$\overbrace{\langle E_{21}, E_{22}\rangle \vdash F_2(\overline{\tilde{S}_j}) <: F_3(\overline{\tilde{S}_k})}$$

We deduce evidence for consistent transitivity for label ordering:

$$\overbrace{\langle E_{11}, E_{12}\rangle \circ^{\preccurlyeq} \langle E_{21}, E_{22}\rangle \vdash F_1(\overline{\tilde{S}_i}) <: F_3(\overline{\tilde{S}_k})}$$

where $\circ^{\preccurlyeq} : \textsc{Label}^2 \times \textsc{Label}^2 \to \textsc{Label}^2$ *is defined as:*

$$\begin{aligned}\langle E_{11}, E_{12}\rangle \circ^{\preccurlyeq} \langle E_{21}, E_{22}\rangle = \\ \alpha_\varepsilon(\{\langle S_{11}, S_{22}\rangle \in \gamma_\varepsilon(\langle E_{11}, E_{22}\rangle) \mid \\ \exists S \in \gamma_E(E_{12}) \cap \gamma_E(E_{21}).S_{11} <: S \wedge S <: S_{22}\})\end{aligned}$$

Proposition 27. $\gamma_E(E_1 \sqcap E_2) = \gamma_E(E_1) \cap \gamma_E(E_2)$.

Then following AGT,

Proposition 28.

$$\langle E_1, E_{21}\rangle \circ^{<:} \langle E_{22}, E_3\rangle = \triangle^{<:}(E_1, E_{21} \sqcap E_{22}, E_3)$$

where

$$\begin{aligned}\triangle^{<:}(E_1, E_2, E_3) = \alpha_\varepsilon(\{\langle S_1, S_3\rangle \in \gamma_\varepsilon(\langle E_1, E_3\rangle) \mid \\ \exists S_2 \in \gamma_\imath(E_2).S_1 <: S_2 \wedge S_2 <: S_3\})\end{aligned}$$

Definition 39 (Intervals join)**.**

$$[\ell_1, \ell_2] \mathbin{\widetilde{\curlyvee}} [\ell_3, \ell_4] = [\ell_1 \curlyvee \ell_3, \ell_2 \curlyvee \ell_4]$$

Definition 40 (Evidence label join)**.**

$$\langle \imath_1, \imath_2\rangle \mathbin{\widetilde{\curlyvee}} \langle \imath_3, \imath_4\rangle = \langle \imath_1 \mathbin{\widetilde{\curlyvee}} \imath_3, \imath_2 \mathbin{\widetilde{\curlyvee}} \imath_4\rangle$$

Definition 41.

$$\begin{aligned}\mathsf{Bool}_{\imath_1} \mathbin{\widetilde{\curlyvee}} \imath_2 &= \mathsf{Bool}_{(\imath_1 \mathbin{\widetilde{\curlyvee}} \imath_2)} \\ E_1 \xrightarrow{\imath_2}_{\imath_1} E_2 \mathbin{\widetilde{\curlyvee}} \imath_3 &= E_1 \xrightarrow[(\imath_1 \mathbin{\widetilde{\curlyvee}} \imath_3)]{\imath_2} E_2 \\ \mathsf{Ref}_{\imath_1}\ E \mathbin{\widetilde{\curlyvee}} \imath_2 &= \mathsf{Ref}_{(\imath_1 \mathbin{\widetilde{\curlyvee}} \imath_2)}\ E\end{aligned}$$

Definition 42.

$$\langle E_1, E_2\rangle \mathbin{\widetilde{\curlyvee}} \langle \imath_1, \imath_2\rangle = \langle E_1 \mathbin{\widetilde{\curlyvee}} \imath_1, E_2 \mathbin{\widetilde{\curlyvee}} \imath_2\rangle$$

Proposition 29. *If* $\varepsilon_S \vdash \widetilde{S}_1 \lesssim \widetilde{S}_2$ *and* $\varepsilon_l \vdash \tilde{\ell}_1 \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_2$ *then* $\varepsilon_S \mathbin{\widetilde{\curlyvee}} \varepsilon_l \vdash \widetilde{S}_1 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_1 <: \widetilde{S}_2 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_2$

C.4 Algorithmic definitions

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

C.4.1 Label Evidences

Definition 43 (Intervals join)**.** *2*

$$[\ell_1, \ell_2] \mathbin{\widetilde{\curlyvee}} [\ell_3, \ell_4] = [\ell_1 \curlyvee \ell_3, \ell_2 \curlyvee \ell_4]$$

Definition 44 (Intervals join)**.**

$$[\ell_1, \ell_2] \mathbin{\widetilde{\curlywedge}} [\ell_3, \ell_4] = [\ell_1 \curlywedge \ell_3, \ell_2 \curlywedge \ell_4]$$

Definition 45.

$$\begin{aligned}bounds(?) &= [\bot, \top] \\ bounds(\ell) &= [\ell, \ell] \\ bounds(\ell_1 \curlyvee \ell_2) &= bounds(\ell_1) \mathbin{\widetilde{\curlyvee}} bounds(\ell_2) \\ bounds(\ell_1 \curlywedge \ell_2) &= bounds(\ell_1) \mathbin{\widetilde{\curlyvee}} bounds(\ell_2) \\ bounds(\ell_1 \sqcap \ell_2) &= bounds(\ell_1) \sqcap bounds(\ell_2) \\ bounds(F_1(\overline{\ell_i}) \curlyvee F_2(\overline{\ell_i})) &= bounds(F_1(\overline{\ell_i})) \mathbin{\widetilde{\curlyvee}} bounds(F_2(\overline{\ell_i})) \\ bounds(F_1(\overline{\ell_i}) \curlywedge F_2(\overline{\ell_i})) &= bounds(F_1(\overline{\ell_i})) \mathbin{\widetilde{\curlywedge}} bounds(F_2(\overline{\ell_i})) \\ bounds(F_1(\overline{\ell_i}) \sqcap F_2(\overline{\ell_i})) &= bounds(F_1(\overline{\ell_i})) \sqcap bounds(F_2(\overline{\ell_i}))\end{aligned}$$

$$\frac{bounds(F_1(\overline{\tilde{\ell}_i})) = [\ell_1, \ell_2] \qquad bounds(F_2(\overline{\tilde{\ell}_j})) = [\ell_3, \ell_4]}{\mathcal{I}_{\preccurlyeq}^{F_1,F_2}(\tilde{\ell}_1, ...\tilde{\ell}_n, \tilde{\ell}_{n+1}, ...\tilde{\ell}_{n+m}) = \langle [\ell_1, \ell_2 \curlywedge \ell_4, \ell_1 \curlyvee \ell_3, \ell_4]\rangle}$$

The algorithmic definition of meet:

$$\begin{aligned}[\ell_1, \ell_2] \sqcap [\ell_3, \ell_4] = [\ell_1 \curlyvee \ell_3, \ell_2 \curlywedge \ell_4] \quad \text{if } valid([\ell_1 \curlyvee \ell_3, \ell_2 \curlywedge \ell_4]) \\ \imath \sqcap \imath' \text{ undefined otherwise}\end{aligned}$$

We calculate the algorithmic definition of $\triangle^{\preccurlyeq}$:

$$\frac{\ell_1 \preccurlyeq \ell_4 \qquad \ell_3 \preccurlyeq \ell_6 \qquad \ell_1 \preccurlyeq \ell_6}{\triangle^{\preccurlyeq}([\ell_1, \ell_2], [\ell_3, \ell_4], [\ell_5, \ell_6]) = \langle [\ell_1, \ell_2 \curlywedge \ell_4 \curlywedge \ell_6], [\ell_1 \curlyvee \ell_3 \curlyvee \ell_5, \ell_6]\rangle}$$

C.4.2 Type Evidences

We define a function $\downarrow$ () to transform functions over types into functions over labels. Also we define function $()^{-1}$ to invert the operator on types, used in the domain and latent effect of function types. Finally we define function $()^{\sqcap}$ to transform type operators into meets, given the invariant property of references.

$$\begin{aligned}\downarrow (x_1 \mathbin{\ddot{\curlyvee}} x_2) &= y_1 \curlyvee y_2 \\ \downarrow (x_1 \mathbin{\ddot{\curlywedge}} x_2) &= y_1 \curlywedge y_2 \\ \downarrow (G_1(\overline{x_i}) \mathbin{\ddot{\curlyvee}} G_2(\overline{x_j})) &= \downarrow (G_1(\overline{x_i})) \curlyvee \downarrow (G_2(\overline{x_j})) \\ \downarrow (G_1(\overline{x_i}) \mathbin{\ddot{\curlywedge}} G_2(\overline{x_j})) &= \downarrow (G_1(\overline{x_i})) \curlywedge \downarrow (G_2(\overline{x_j})) \\ (x_1 \mathbin{\ddot{\curlyvee}} x_2)^{-1} &= y_1 \mathbin{\ddot{\curlywedge}} y_2 \\ (x_1 \mathbin{\ddot{\curlywedge}} x_2)^{-1} &= y_1 \mathbin{\ddot{\curlyvee}} y_2 \\ (G_1(\overline{x_i}) \mathbin{\ddot{\curlyvee}} G_2(\overline{x_j}))^{-1} &= (G_1(\overline{x_i}))^{-1} \mathbin{\ddot{\curlywedge}} (G_2(\overline{x_j}))^{-1} \\ (G_1(\overline{x_i}) \mathbin{\ddot{\curlywedge}} G_2(\overline{x_j}))^{-1} &= (G_1(\overline{x_i}))^{-1} \mathbin{\ddot{\curlyvee}} (G_2(\overline{x_j}))^{-1} \\ (x_1 \mathbin{\ddot{\curlyvee}} x_2)^{\sqcap} &= y_1 \sqcap y_2 \\ (x_1 \mathbin{\ddot{\curlywedge}} x_2)^{\sqcap} &= y_1 \sqcap y_2 \\ (G_1(\overline{x_i}) \mathbin{\ddot{\curlyvee}} G_2(\overline{x_j}))^{\sqcap} &= (G_1(\overline{x_i}))^{\sqcap} \sqcap (G_2(\overline{x_j}))^{\sqcap} \\ (G_1(\overline{x_i}) \mathbin{\ddot{\curlywedge}} G_2(\overline{x_j}))^{\sqcap} &= (G_1(\overline{x_i}))^{\sqcap} \sqcap (G_2(\overline{x_j}))^{\sqcap}\end{aligned}$$

We use case-based analysis to calculate the algorithmic rules for the interior of consistent subtyping on gradual security types:

$$\frac{\mathcal{I}_{\preccurlyeq}^{\downarrow(G_1),\downarrow(G_2)}(\overline{\ell_i}) = \langle \imath_1, \imath_2 \rangle}{\mathcal{I}_{<:}^{G_1,G_2}(\overline{\mathsf{Bool}_{\tilde{\ell}_i}}) = \langle \mathsf{Bool}_{\imath_1}, \mathsf{Bool}_{\imath_2} \rangle}$$

$$\frac{\begin{array}{c}\mathcal{I}_{<:}^{G_1^{-1},G_2^{-1}}(\overline{\widetilde{S}_{i1}}) = \langle E'_{21}, E'_{11} \rangle \quad \mathcal{I}_{<:}^{G_1,G_2}(\overline{\widetilde{S}_{i2}}) = \langle E_{12}, E_{22} \rangle \\ \mathcal{I}_{\preccurlyeq}^{\downarrow(G_1),\downarrow(G_2)}(\overline{\ell_{i1}}) = \langle \imath_{11}, \imath_{12} \rangle \\ \mathcal{I}_{\preccurlyeq}^{\downarrow(G_1^{-1}),\downarrow(G_2^{-1})}(\overline{\ell_{i2}}) = \langle \imath_{21}, \imath_{22} \rangle\end{array}}{\mathcal{I}_{<:}^{G_1,G_2}(\overline{\widetilde{S}_{i1} \xrightarrow{\tilde{\ell}_{i2}}_{\tilde{\ell}_{i1}} \widetilde{S}_{i2}}) = \langle E_{11} \xrightarrow{\imath_{12}}_{\imath_{11}} E_{12}, E_{21} \xrightarrow{\imath_{22}}_{\imath_{21}} E_{22} \rangle}$$

$$\frac{\begin{array}{c}\mathcal{I}_{\preccurlyeq}^{\downarrow(G_1),\downarrow(G_2)}(\overline{\ell_i}, \overline{\ell_j}) = \langle \imath_1, \imath_2 \rangle \\ \mathcal{I}_{\preccurlyeq}^{G_1^{\sqcap},G_2^{\sqcap}}(\overline{\widetilde{S}_i}, \overline{\widetilde{S}_j}) = \langle E_1, E_2 \rangle \\ \mathcal{I}_{\preccurlyeq}^{G_2^{\sqcap},G_1^{\sqcap}}(\overline{\widetilde{S}_j}, \overline{\widetilde{S}_i}) = \langle E'_2, E'_1 \rangle\end{array}}{\mathcal{I}_{<:}^{G_1,G_2}(\overline{\mathsf{Ref}_{\tilde{\ell}_i}\ \widetilde{S}_i}, \overline{\mathsf{Ref}_{\tilde{\ell}_j}\ \widetilde{S}_j}) = \langle \mathsf{Ref}_{\imath_1}\ E_1 \sqcap E'_1, \mathsf{Ref}_{\imath_2}\ E_2 \sqcap E'_2 \rangle}$$

We calculate a recursive meet operator for gradual types:

$$\mathsf{Bool}_\imath \sqcap \mathsf{Bool}_{\imath'} = \mathsf{Bool}_{\imath \sqcap \imath'}$$

$$(\widetilde{S}_{11} \xrightarrow{\imath_2}_{\imath_1} \widetilde{S}_{12}) \sqcap (\widetilde{S}_{21} \xrightarrow{\imath'_2}_{\imath'_1} \widetilde{S}_{22}) = (\widetilde{S}_{11} \sqcap \widetilde{S}_{21}) \xrightarrow{\imath_2 \sqcap \imath'_2}_{\imath_1 \sqcap \imath'_1} (\widetilde{S}_{12} \sqcap \widetilde{S}_{22})$$

$$\mathsf{Ref}_\imath\ \widetilde{S}_1 \sqcap \mathsf{Ref}_{\imath'}\ \widetilde{S}_2 = \mathsf{Ref}_{\imath \sqcap \imath'}\ \widetilde{S}_1 \sqcap \widetilde{S}_2$$

$$\widetilde{S} \sqcap \widetilde{S}' \text{ undefined otherwise}$$

We calculate a recursive definition for $\triangle^{<:}$ by case analysis on the structure of the second argument,

$$\frac{\triangle^{\preccurlyeq}(\imath_1, \imath_2, \imath_3) = \langle \imath'_1, \imath'_3 \rangle}{\triangle^{<:}(\mathsf{Bool}_{\imath_1}, \mathsf{Bool}_{\imath_2}, \mathsf{Bool}_{\imath_3}) = \langle \mathsf{Bool}_{\imath'_1}, \mathsf{Bool}_{\imath'_3} \rangle}$$

$$\frac{\begin{array}{c}\triangle^{<:}(E_{31}, E_{21}, E_{11}) = \langle \widetilde{S}'_{31}, \widetilde{S}'_{11} \rangle \\ \triangle^{<:}(E_{12}, E_{22}, E_{32}) = \langle \widetilde{S}'_{12}, \widetilde{S}'_{32} \rangle \\ \triangle^{\preccurlyeq}(\imath_1, \imath_2, \imath_3) = \langle \imath'_1, \imath'_3 \rangle \\ \triangle^{\preccurlyeq}(\imath_{13}, \imath_{12}, \imath_{11}) = \langle \imath'_{13}, \imath'_{11} \rangle\end{array}}{\begin{array}{l}\triangle^{<:}(E_{11} \xrightarrow{\imath_{11}}_{\imath_1} E_{12}, E_{21} \xrightarrow{\imath_{12}}_{\imath_2} E_{22}, E_{31} \xrightarrow{\imath_{13}}_{\imath_3} E_{32}) \\ = \langle \widetilde{S}'_{11} \xrightarrow{\imath'_{11}}_{\imath'_1} \widetilde{S}'_{12}, \widetilde{S}'_{31} \xrightarrow{\imath'_{13}}_{\imath'_3} \widetilde{S}'_{32} \rangle\end{array}}$$

$$\frac{\begin{array}{c}\triangle^{\preccurlyeq}(\imath_1, \imath_2, \imath_3) = \langle \imath'_1, \imath'_3 \rangle \\ \widetilde{S}'_1 = E_1 \sqcap E_2 \qquad \widetilde{S}'_3 = E_2 \sqcap E_3\end{array}}{\triangle^{<:}(\mathsf{Ref}_{\imath_1}\ E_1, \mathsf{Ref}_{\imath_2}\ E_2, \mathsf{Ref}_{\imath_3}\ E_3) = \langle \mathsf{Ref}_{\imath'_1}\ \widetilde{S}'_1, \mathsf{Ref}_{\imath'_3}\ \widetilde{S}'_3 \rangle}$$

C.5 Proofs

Proposition 21 ($\alpha_\imath$ is Sound)**.** *If $\widehat{\ell}$ is not empty, then $\widehat{\ell} \subseteq \gamma_\imath(\alpha_\imath(\widehat{\ell}))$.*

Proof. Suppose $\widehat{\ell} = \{\overline{\ell_i}\}$. By definition of $\alpha_{\varepsilon_\ell}$, $\alpha_\imath(\{\overline{\ell_i}\}) = [\lambda\overline{\ell_i}, \gamma\overline{\ell_i}]$. Therefore

$$\gamma_\imath(\alpha_\imath(\{\overline{\ell_i}\})) = \{\ell \mid \ell \in \textsc{Label}, \lambda\overline{\ell_i} \preccurlyeq \ell \preccurlyeq \gamma\overline{\ell_i}\}$$

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

Proposition 22 ($\alpha_\imath$ is Optimal)**.** *If $\widehat{\ell} \subseteq \gamma_\imath(\imath)$ then $\alpha_\imath(\widehat{\ell}) \sqsubseteq \imath$.*

Proof. By case analysis on the structure of $\imath$. If $\imath = [\ell_1, \ell_2]$, $\gamma_{\varepsilon_\ell}(\imath) = \{\ell \mid \ell \in \textsc{Label}, \ell_1 \preccurlyeq \ell \preccurlyeq \ell_2\}$; $\widehat{\ell} \subseteq \{\ell \mid \ell \in \textsc{Label}, \ell_1 \preccurlyeq \ell \preccurlyeq \ell_2\}, \widehat{\ell} \neq \emptyset$ implies $\alpha_{\varepsilon_\ell}(\widehat{\ell}) = [\ell_3, \ell_4]$, where $\ell_1 \preccurlyeq \ell_3$ and $\ell_4 \preccurlyeq \ell_2$, therefore $[\ell_3, \ell_4] \sqsubseteq \imath$ (if $\widehat{\ell} = \emptyset$, $\alpha_{\varepsilon_\ell}(\widehat{\ell})$ is undefined). □

Proposition 23 (α_E is Sound)**.** *If $valid(\widehat{S})$ then $\widehat{S} \subseteq \gamma_E(\alpha_E(\widehat{S}))$.*

Proof. By well-founded induction on $\widehat{S}$. Similar to Prop 3. □

Proposition 24 (α_E is Optimal)**.** *If $valid(\widehat{S})$ and $\widehat{S} \subseteq \gamma_E(E)$ then $\alpha_E(\widehat{S}) \sqsubseteq E$.*

Proof. By induction on the structure of $\widetilde{S}$. Similar to Prop 4. □

Proposition 25. $\gamma_\imath(\imath_1 \sqcap \imath_2) = \gamma_\imath(\imath_1) \cap \gamma_\imath(\imath_2)$.

Proof.

$$\begin{array}{ll}\gamma_\imath(\imath_1 \sqcap \imath_2) & = \gamma_\imath(\alpha_\imath(\gamma_\imath(\imath_1) \cap \gamma_\imath(\imath_1))) \\ & \subseteq \gamma_\imath(\imath_1) \cap \gamma_\imath(\imath_1) \quad \text{(soundness of } \alpha_\imath\text{)}\end{array}$$

Let $\ell \in \gamma_\imath(\imath_1) \cap \gamma_\imath(\imath_1)$. We now that $\gamma_\imath(\imath_1 \sqcap \imath_2)$ is defined. Suppose $\imath_1 = [\ell_1, \ell_2]$ and $\imath_2 = [\ell_3, \ell_4]$. Therefore $\imath_1 \sqcap \imath_2 = [\ell_1 \curlyvee \ell_3, \ell_2 \curlywedge \ell_4]$.

But $\gamma_\imath(\imath_1) \cap \gamma_\imath(\imath_1) = \{\ell \mid \ell \in \textsc{Label}, \ell_1 \preccurlyeq \ell \preccurlyeq \ell_2\} \cap \{\ell \mid \ell \in \textsc{Label}, \ell_3 \preccurlyeq \ell \preccurlyeq \ell_4\}$. Which is equivalent to $\{\ell \mid \ell \in \textsc{Label}, \ell_1 \preccurlyeq \ell \preccurlyeq \ell_2 \wedge \ell_3 \preccurlyeq \ell \preccurlyeq \ell_4\}$, equivalent to $\{\ell \mid \ell \in \textsc{Label}, \ell_1 \curlyvee \ell_3 \preccurlyeq \ell \preccurlyeq \ell_2 \curlywedge \ell_4\}$. Which is by definition $\gamma_\imath([\ell_1 \curlyvee \ell_3, \ell_2 \curlywedge \ell_4])$, and the result holds.

□

Proposition 26.

$$\langle \imath_1, \imath_{21} \rangle \circ^{\preccurlyeq} \langle \imath_{22}, \imath_3 \rangle = \triangle^{\preccurlyeq}(\imath_1, \imath_{21} \sqcap \imath_{22}, \imath_3)$$

where

$$\begin{array}{l}\triangle^{\preccurlyeq}(\imath_1, \imath_2, \imath_3) = \alpha_\varepsilon(\{\langle \ell_1, \ell_3 \rangle \in \gamma_\varepsilon(\langle \imath_1, \imath_3 \rangle) \mid \\ \qquad\qquad \exists \ell_2 \in \gamma_\imath(\imath_2). \ell_1 \preccurlyeq \ell_2 \wedge \ell_2 \preccurlyeq \ell_3\})\end{array}$$

Proof. Follows directly from the definition of consistent transitivity and Prop 25. □

Proposition 27. $\gamma_E(E_1 \sqcap E_2) = \gamma_E(E_1) \cap \gamma_E(E_2)$.

Proof. By induction on evidence types ε_1 and ε_2 and Prop 25. □

Proposition 28.

$$\langle E_1, E_{21} \rangle \circ^{<:} \langle E_{22}, E_3 \rangle = \triangle^{<:}(E_1, E_{21} \sqcap E_{22}, E_3)$$

where

$$\begin{array}{l}\triangle^{<:}(E_1, E_2, E_3) = \alpha_\varepsilon(\{\langle S_1, S_3 \rangle \in \gamma_\varepsilon(\langle E_1, E_3 \rangle) \mid \\ \qquad\qquad \exists S_2 \in \gamma_\imath(E_2). S_1 <: S_2 \wedge S_2 <: S_3\})\end{array}$$

Proof. Follows directly from the definition of consistent transitivity and Prop 27. □

Proposition 29. *If $\varepsilon_S \vdash \widetilde{S}_1 \lesssim \widetilde{S}_2$ and $\varepsilon_l \vdash \tilde{\ell}_1 \precsim \tilde{\ell}_2$ then $\varepsilon_S \mathbin{\widetilde{\curlyvee}} \varepsilon_l \vdash \widetilde{S}_1 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_1 <: \widetilde{S}_2 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_2$*

Proof. By induction on types $\widetilde{S}_1$ and $\widetilde{S}_2$, using the definition of $\mathcal{I}_{<:}$ and Proposition 7. □

Proposition 30.

$$\frac{\ell_1 \preccurlyeq \ell_4 \qquad \ell_3 \preccurlyeq \ell_6 \qquad \ell_1 \preccurlyeq \ell_6}{\triangle^{\preccurlyeq}([\ell_1, \ell_2], [\ell_3, \ell_4], [\ell_5, \ell_6]) = \langle [\ell_1, \ell_2 \curlywedge \ell_4 \curlywedge \ell_6], [\ell_1 \curlyvee \ell_3 \curlyvee \ell_5, \ell_6] \rangle}$$

Proof. By definition:

$$\begin{aligned}\triangle^{\preccurlyeq}([\ell_1,\ell_2],[\ell_3,\ell_4],[\ell_5,\ell_6]) = \\ \alpha_\varepsilon(\{\langle \ell_1',\ell_3'\rangle \in \gamma_\varepsilon(\langle[\ell_1,\ell_2],[\ell_5,\ell_6]\rangle) \mid \\ \exists \ell_2' \in \gamma_\iota([\ell_3,\ell_4]).\ell_1' \preccurlyeq \ell_2' \preccurlyeq \ell_3'\})\end{aligned}$$

It is easy to see that $\alpha_\iota(\{\ \ell_{1i}'\ \}) = [\ell_1,\ell_{12}']$, for some ℓ_{12}'. We know that $\ell_{12}' \preccurlyeq \ell_2$, $\ell_{12}' \preccurlyeq \ell_4$ and $\ell_{12}' \preccurlyeq \ell_6$, *i.e.* $\ell_{12}' \preccurlyeq \ell_2 \curlywedge \ell_4 \curlywedge \ell_6$. But $\ell_2 \curlywedge \ell_4 \curlywedge \ell_6 \preccurlyeq \ell_4 \preccurlyeq \ell_6$ therefore

$$\begin{aligned}\langle \ell_2 \curlywedge \ell_4 \curlywedge \ell_6, \ell_6\rangle \in \{\langle \ell_1',\ell_3'\rangle \in \gamma_\varepsilon(\langle[\ell_1,\ell_2],[\ell_5,\ell_6]\rangle) \mid \\ \exists \ell_2' \in \gamma_\iota([\ell_3,\ell_4]).\ell_1' \preccurlyeq \ell_2' \preccurlyeq \ell_3'\}\end{aligned}$$

and by definition of α_ι, $\ell_2 \curlywedge \ell_4 \curlywedge \ell_6 \preccurlyeq \ell_{12}'$, then $\alpha_\iota(\{\ \ell_{1i}'\ \}) = [\ell_1, \ell_2 \curlywedge \ell_4 \curlywedge \ell_6]$. Similar argument is used to prove that $\alpha_\iota(\{\ \ell_{3i}'\ \}) = [\ell_1 \curlyvee \ell_3 \curlyvee \ell_5, \ell_6]$. □

Lemma 31. *Let* $\ell_i \in$ LABEL, *then* $(\ell_1 \curlywedge \ell_2) \curlyvee (\ell_3 \curlywedge \ell_4) \preccurlyeq (\ell_1 \curlyvee \ell_3) \curlywedge (\ell_2 \curlyvee \ell_4)$.

Proof.

$$\begin{array}{ll} & (\ell_1 \curlywedge \ell_2) \curlyvee (\ell_3 \curlywedge \ell_4) \\ \preccurlyeq & (\ell_1 \curlyvee (\ell_3 \curlywedge \ell_4)) \curlywedge (\ell_2 \curlyvee (\ell_3 \curlywedge \ell_4)) \\ \preccurlyeq & ((\ell_1 \curlyvee \ell_3) \curlywedge (\ell_1 \curlyvee \ell_4)) \curlywedge ((\ell_2 \curlyvee \ell_3) \curlywedge (\ell_2 \curlyvee \ell_4)) \\ \preccurlyeq & (\ell_1 \curlyvee \ell_3) \curlywedge (\ell_2 \curlyvee \ell_4) \end{array}$$

□

Proposition 7.

If $\varepsilon_1 \vdash \overbrace{F_{11}(\overline{\tilde{\ell}_i}) \preccurlyeq F_{12}(\overline{\tilde{\ell}_j})}$ *and* $\varepsilon_2 \vdash \overbrace{F_{21}(\overline{\tilde{\ell}_i}) \preccurlyeq F_{22}(\overline{\tilde{\ell}_j})}$ *then* $\varepsilon_1 \mathbin{\widetilde{\curlyvee}} \varepsilon_2 \vdash \overbrace{F_{11}(\overline{\tilde{\ell}_i}) \mathbin{\widetilde{\curlyvee}} F_{21}(\overline{\tilde{\ell}_i}) \preccurlyeq F_{12}(\overline{\tilde{\ell}_j}) \mathbin{\widetilde{\curlyvee}} F_{22}(\overline{\tilde{\ell}_j})}$.

Proof. By definition of interior noticing that $\varepsilon_1 \mathbin{\widetilde{\curlyvee}} \varepsilon_2$ can be more precise than the interior 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 $\varepsilon_1 \mathbin{\widetilde{\curlyvee}} \varepsilon_2 = \langle[\ell_1 \curlyvee \ell_5, \ell_2 \curlyvee \ell_6],[\ell_3 \curlyvee \ell_6, \ell_4 \curlyvee \ell_8]\rangle$.

If $bounds(\overline{F_{11}(\tilde{\ell}_i)}) = [\ell_{111}',\ell_{112}']$, $bounds(\overline{F_{12}(\tilde{\ell}_i)}) = [\ell_{121}',\ell_{122}']$, $bounds(\overline{F_{21}(\tilde{\ell}_i)}) = [\ell_{211}',\ell_{212}']$ and $bounds(\overline{F_{22}(\tilde{\ell}_i)}) = [\ell_{221}',\ell_{222}']$ We know that $\mathcal{I}_{\preccurlyeq}^{F_{11},F_{12}}(\overline{\tilde{\ell}_i}) = \langle[\ell_{111}',\ell_{112}' \curlywedge \ell_{122}', \ell_{111}' \curlyvee \ell_{121}', \ell_{122}']\rangle$. Therefore $\ell_{111}' \preccurlyeq \ell_1$, $\ell_2 \preccurlyeq \ell_{112}' \curlywedge \ell_{122}'$, $\ell_{111}' \curlyvee \ell_{121}' \preccurlyeq \ell_3$ and $\ell_4 \preccurlyeq \ell_{122}'$. Using the same argument, $\mathcal{I}_{\preccurlyeq}^{F_{21},F_{22}}(\overline{\tilde{\ell}_i}) = \langle[\ell_{211}',\ell_{212}' \curlywedge \ell_{222}', \ell_{211}' \curlyvee \ell_{221}', \ell_{222}']\rangle$. Therefore $\ell_{211}' \preccurlyeq \ell_5$, $\ell_6 \preccurlyeq \ell_{212}' \curlywedge \ell_{222}'$, $\ell_{211}' \curlyvee \ell_{221}' \preccurlyeq \ell_7$ and $\ell_8 \preccurlyeq \ell_{222}'$.

But the $\mathcal{I}_{\preccurlyeq}^{F_1',F_2'}(\overline{\tilde{\ell}_i}) = \langle[\ell_1', \ell_2' \curlywedge \ell_4'], [\ell_1' \curlyvee \ell_3', \ell_4']\rangle$ where $bounds(\overline{F_1'(\tilde{\ell}_i)}) = bounds(\overline{F_{11}(\tilde{\ell}_i)}) \mathbin{\widetilde{\curlyvee}} bounds(\overline{F_{21}(\tilde{\ell}_i)}) = [\ell_{111}',\ell_{112}'] \mathbin{\widetilde{\curlyvee}} [\ell_{211}',\ell_{212}'] = [\ell_{111}' \curlyvee \ell_{211}', \ell_{112}' \curlyvee \ell_{212}']$, and $bounds(\overline{F_2'(\tilde{\ell}_i)}) = bounds(\overline{F_{12}(\tilde{\ell}_i)}) \mathbin{\widetilde{\curlyvee}} bounds(\overline{F_{22}(\tilde{\ell}_i)}) = [\ell_{121}',\ell_{122}'] \mathbin{\widetilde{\curlyvee}} [\ell_{221}',\ell_{222}'] = [\ell_{121}' \curlyvee \ell_{221}', \ell_{122}' \curlyvee \ell_{222}']$.

We need to prove that $[\ell_1 \curlyvee \ell_5, \ell_2 \curlyvee \ell_6] \sqsubseteq [\ell_{111}' \curlyvee \ell_{211}', \ell_{112}' \curlyvee \ell_{212}']$, *i.e.* $\ell_{111}' \curlyvee \ell_{211}' \preccurlyeq \ell_1 \curlyvee \ell_5$ and $\ell_2 \curlyvee \ell_6 \preccurlyeq \ell_{112}' \curlyvee \ell_{212}'$. But $\ell_{111}' \preccurlyeq \ell_1$ and $\ell_{211}' \preccurlyeq \ell_5$, therefore $\ell_{111}' \curlyvee \ell_{211}' \preccurlyeq \ell_1 \curlyvee \ell_5$. Similarly, as $\ell_2 \preccurlyeq \ell_{112}' \curlywedge \ell_{122}'$ and $\ell_6 \preccurlyeq \ell_{212}' \curlywedge \ell_{222}'$, then $\ell_2 \curlyvee \ell_6 \preccurlyeq \ell_{112}' \curlyvee \ell_{212}'$. Therefore $[\ell_1 \curlyvee \ell_5, \ell_2 \curlyvee \ell_6] \sqsubseteq [\ell_{111}' \curlyvee \ell_{211}', \ell_{112}' \curlyvee \ell_{212}']$.

Using analogous argument, we also know that $[\ell_3 \curlyvee \ell_6, \ell_4 \curlyvee \ell_8] \sqsubseteq [\ell_{121}' \curlyvee \ell_{221}', \ell_{122}' \curlyvee \ell_{222}']$. Therefore $\varepsilon_1 \mathbin{\widetilde{\curlyvee}} \varepsilon_2 \sqsubseteq \mathcal{I}_{\preccurlyeq}^{F_1',F_2'}(\overline{\tilde{\ell}_i})$, and the result holds. □

Lemma 32. *Let* $S_1, S_2 \in$ TYPE. *Then*

1. *If* $(S_1 \mathbin{\ddot{\curlyvee}} S_2)$ *is defined then* $S_1 <: (S_1 \mathbin{\ddot{\curlyvee}} S_2)$.
2. *If* $(S_1 \mathbin{\ddot{\curlywedge}} S_2)$ *is defined then* $(S_1 \mathbin{\ddot{\curlywedge}} S_2) <: S_1$.

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

Case (Bool_ℓ). If $S_1 = \mathsf{Bool}_{\ell_1}$ then as $(S_1 \mathbin{\ddot{\curlyvee}} S_2)$ is defined then S_2 must have the form Bool_{ℓ_2} for some ℓ_2. Therefore $(S_1 \mathbin{\ddot{\curlyvee}} S_2) = \mathsf{Bool}_{(\ell_1 \curlyvee \ell_2)}$. But by definition of $\preccurlyeq$, $\ell_1 \preccurlyeq (\ell_1 \curlyvee \ell_2)$ and therefore we use ($<:_{\mathsf{Bool}}$) to conclude that $\mathsf{Bool}_{\ell_1} <: \mathsf{Bool}_{(\ell_1 \curlyvee \ell_2)}$, i.e. $S_1 <: (S_1 \mathbin{\ddot{\curlyvee}} S_2)$.

Case ($S \to_\ell S$). If $S_1 = S_{11} \to_{\ell_1} S_{12}$ then as $(S_1 \mathbin{\ddot{\curlyvee}} S_2)$ is defined then S_2 must have the form $S_{21} \to_{\ell_2} S_{22}$ for some S_{21}, S_{22} and ℓ_2.
We also know that $(S_1 \mathbin{\ddot{\curlyvee}} S_2) = (S_{11} \mathbin{\ddot{\curlywedge}} S_{21}) \to_{(\ell_1 \curlyvee \ell_2)} (S_{12} \mathbin{\ddot{\curlywedge}} S_{22})$. By definition of $\preccurlyeq$, $\ell_1 \preccurlyeq (\ell_1 \curlyvee \ell_2)$. Also, as $(S_1 \mathbin{\ddot{\curlyvee}} S_2)$ is defined then $(S_{11} \mathbin{\ddot{\curlywedge}} S_{21})$ is defined. Using the induction hypothesis of (2) on S_{11}, $(S_{11} \mathbin{\ddot{\curlywedge}} S_{21}) <: S_{11}$. Also, using the induction hypothesis of (1) on S_{12} we also know that $S_{12} <: (S_{12} \mathbin{\ddot{\curlywedge}} S_{22})$. Then by ($<:_\to$) we can conclude that $S_{11} \to_{\ell_1} S_{12} <: (S_{11} \mathbin{\ddot{\curlywedge}} S_{21}) \to_{(\ell_1 \curlyvee \ell_2)} (S_{12} \mathbin{\ddot{\curlywedge}} S_{22})$, i.e. $S_1 <: (S_1 \mathbin{\ddot{\curlyvee}} S_2)$.

The proof of (2) is similar to (1) but using the argument that $(\ell_1 \curlywedge \ell_2) \preccurlyeq \ell_1$. □

Lemma 33. *Let* $S \in$ TYPE *and* $\ell \in$ LABEL. *Then* $S <: S \curlyvee \ell$.

Proof. Straigthforward case analysis on type S using the fact that $\ell \preccurlyeq (\ell' \curlyvee \ell)$ for any ℓ'. □

Lemma 34. *Let* $S_1, S_2 \in$ TYPE *such that* $S_1 <: S_2$, *and let* $\ell_1, \ell_2 \in$ LABEL *such that* $\ell_1 \preccurlyeq \ell_2$. *Then* $S_1 \curlyvee \ell_1 <: S_2 \curlyvee \ell_2$.

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

C.6 Reduction

This section presents the full definition of the intrinsic reduction rules in Figure 12.

C.7 Properties of the Gradual Security Language

This section present the proof of type safety in section C.7.1, and the proof of dynamic gradual guarantee in section C.7.2.

C.7.1 Type Safety

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

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

Definition 46 (μ is well typed)**.** *A store* μ *is said to be* well typed *with respect to an intrinsic term* $t^{\widetilde{S}}$, *written* $t^{\widetilde{S}} \vdash \mu$, *if*

1. $freeLocs(t^{\widetilde{S}}) \subseteq dom(\mu)$, *and*
2. $\forall\, v \in cod(\mu), v \vdash \mu$ *and*
3. $\forall\, l^{\widetilde{S}} \in dom(\mu), \forall \tilde{\ell}_c, \tilde{\ell}_r, \varepsilon_r$, *such that* $\varepsilon_r \vdash \tilde{\ell}_r \mathbin{\widetilde{\preccurlyeq}} \tilde{\ell}_c$, *then* $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mu(l^{\widetilde{S}}) \in \text{TERM}_{\widetilde{S}}$.

Lemma 35. *Suppose* $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}} \in \text{TERM}_{\widetilde{S}}$, *then* $\forall \tilde{\ell}_r'$ *such that* $\tilde{\ell}_r' \preccurlyeq \tilde{\ell}_c$ *and* $\varepsilon_r' \vdash \tilde{\ell}_r' \preccurlyeq \tilde{\ell}_c$, *then* $\varepsilon_r' \tilde{\ell}_r' \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}} \in \text{TERM}_{\widetilde{S}}$.

Proof. By induction on the derivation of $\varepsilon_r' \tilde{\ell}_r' \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}} \in \text{TERM}_{\widetilde{S}}$. Noticing that no typing derivation depends on $\varepsilon_r' \tilde{\ell}_r'$, save for the judgement $\varepsilon_r' \vdash \tilde{\ell}_r' \preccurlyeq \tilde{\ell}_c$ which is premise of this lemma. □

$\boxed{\overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} : \text{CONF}_{\widetilde{S}} \times (\text{CONF}_{\widetilde{S}} \cup \{\textbf{error}\})}$

$$(\text{R}\longrightarrow)\ \frac{t^{\widetilde{S}} \mid \mu \ \overset{e\ell}{\longrightarrow}_{\tilde{\ell}_c}\ r \qquad r \in (\text{CONF}_{\widetilde{S}} \cup \{\textbf{error}\})}{t^{\widetilde{S}} \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ r}$$

$$(\text{R}f)\ \frac{t_1^{\widetilde{S}} \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ t_2^{\widetilde{S}} \mid \mu'}{f[t_1^{\widetilde{S}}] \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ f[t_2^{\widetilde{S}}] \mid \mu'}$$

$$(\text{R}g)\ \frac{et \longrightarrow_c et'}{g[et] \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ g[et'] \mid \mu'}$$

$$(\text{Rprot})\ \frac{t_1^{\widetilde{S}'} \mid \mu \ \overset{\varepsilon'_r(\tilde{\ell}_r \widetilde{\Upsilon} \tilde{\ell}')}{\longmapsto}_{\tilde{\ell}'_c}\ t_2^{\widetilde{S}'} \mid \mu'}{\text{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell\tilde{\ell}'} \varepsilon t_1^{\widetilde{S}'} \mid \mu \ \overset{\varepsilon_r\tilde{\ell}_r}{\longmapsto}_{\tilde{\ell}_c}\ \text{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell\tilde{\ell}'} \varepsilon t_2^{\widetilde{S}'} \mid \mu'}$$

$$(\text{Rprotg})\ \frac{et \longrightarrow_c et'}{\text{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell\tilde{\ell}'} et \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ \text{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell\tilde{\ell}'} et' \mid \mu'}$$

$$(\text{R}f\text{err})\ \frac{t_1^{\widetilde{S}} \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ \textbf{error}}{f[t_1^{\widetilde{S}}] \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ \textbf{error}}$$

$$(\text{R}g\text{err})\ \frac{et \longrightarrow_c \textbf{error}}{g[et] \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ \textbf{error}}$$

$$(\text{Rproterr})\ \frac{t^{\widetilde{S}'} \mid \mu \ \overset{\varepsilon'_r(\tilde{\ell}_r \widetilde{\Upsilon} \tilde{\ell}')}{\longmapsto}_{\tilde{\ell}'_c}\ \textbf{error}}{\text{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell\tilde{\ell}'} \varepsilon t^{\widetilde{S}'} \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ \textbf{error}}$$

$$(\text{Rprotgerr})\ \frac{et \longrightarrow_c \textbf{error}}{\text{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell\tilde{\ell}'} et \mid \mu \ \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}\ \textbf{error}}$$

Figure 12. GSL$_{\text{Ref}}$: Intrinsic Reduction

Lemma 36. *Suppose* $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v \in \text{TERM}_{\widetilde{S}}$, *then* $\forall \tilde{\ell}_c', \tilde{\ell}_r'$ *such that* $\varepsilon_r' \vdash \tilde{\ell}_r' \preccurlyeq \tilde{\ell}_c'$, *then* $\varepsilon_r' \tilde{\ell}_r' \overset{\tilde{\ell}_c'}{\triangleright} v \in \text{TERM}_{\widetilde{S}}$.

Proof. By induction on the derivation of $\varepsilon_r' \tilde{\ell}_r' \overset{\tilde{\ell}_c'}{\triangleright} v$ observing that for values, there is no premise that depends on the PC. □

Lemma 37 (Canonical forms)**.** *Consider a value* $v \in \text{TERM}_{\widetilde{S}}$. *Then either* $v = u$, *or* $v = \varepsilon u :: \widetilde{S}$ *with* $u \in \text{TERM}_{\widetilde{S}'}$ *and* $\varepsilon \vdash \widetilde{S}' \lesssim \widetilde{S}$. *Furthermore:*

1. *If* $\widetilde{S} = \mathsf{Bool}_{\tilde{\ell}}$ *then either* $v = b_{\tilde{\ell}}$ *or* $v = \varepsilon b_{\tilde{\ell}'} :: \mathsf{Bool}_{\tilde{\ell}}$ *with* $b_{\tilde{\ell}'} \in \text{TERM}_{\mathsf{Bool}_{\tilde{\ell}'}}$ *and* $\varepsilon \vdash \mathsf{Bool}_{\tilde{\ell}'} \lesssim \mathsf{Bool}_{\tilde{\ell}}$.
2. *If* $\widetilde{S} = \widetilde{S}_1 \overset{\tilde{\ell}_c}{\longrightarrow}_{\tilde{\ell}} \widetilde{S}_2$ *then either* $v = (\lambda^{\tilde{\ell}_c} x^{\widetilde{S}_1}.t^{\widetilde{S}_2})_{\tilde{\ell}}$ *with* $t^{\widetilde{S}_2} \in \text{TERM}_{\widetilde{S}_2}$ *or* $v = \varepsilon(\lambda^{\tilde{\ell}'_c} x^{\widetilde{S}'_1}.t^{\widetilde{S}'_2})_{\tilde{\ell}'} :: \widetilde{S}_1 \overset{\tilde{\ell}_c}{\longrightarrow}_{\tilde{\ell}} \widetilde{S}_2$ *with* $t^{\widetilde{S}'_2} \in \text{TERM}_{\widetilde{S}'_2}$ *and* $\varepsilon \vdash \widetilde{S}'_1 \overset{\tilde{\ell}'_c}{\longrightarrow}_{\tilde{\ell}'} \widetilde{S}'_2 \lesssim \widetilde{S}_1 \overset{\tilde{\ell}_c}{\longrightarrow}_{\tilde{\ell}} \widetilde{S}_2$.
3. *If* $\widetilde{S} = \mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1$ *then either* $v = l_{\tilde{\ell}}^{\widetilde{S}_1}$ *or* $v = \varepsilon l_{\tilde{\ell}'}^{\widetilde{S}'_1} :: \mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1$ *with* $l_{\tilde{\ell}'}^{\widetilde{S}'_1} \in \mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S}'_1$ *and* $\varepsilon \vdash \mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S}'_1 \lesssim \mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1$.

Proof. By direct inspection of the formation rules of gradual intrinsic terms (Figure 4). □

Lemma 38 (Substitution)**.** *If* $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}} \in \text{TERM}_{\widetilde{S}}$ *and* $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v \in \text{TERM}_{\widetilde{S}_1}$, *then* $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} [v/x^{\widetilde{S}_1}] t^{\widetilde{S}} \in \text{TERM}_{\widetilde{S}}$.

Proof. By induction on the derivation of $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}}$. □

Proposition 39 ($\longrightarrow$ is well defined)**.** *If* $t^{\widetilde{S}} \mid \mu \longrightarrow r$ *and* $t^{\widetilde{S}} \vdash \mu$, *then* $r \in \text{CONFIG}_{\widetilde{S}} \cup \{\textbf{error}\}$ *and if* $r = t'^{\widetilde{S}} \mid \mu' \in \text{CONFIG}_{\widetilde{S}}$ *then also* $t'^{\widetilde{S}} \mid \mu'$ *and* $dom(\mu) \subseteq dom(\mu')$.

Proof. By induction on the structure of a derivation of $t^{\widetilde{T}} \mid \mu \longrightarrow r$, considering the last rule used in the derivation. Suppose $e\ell = \varepsilon_r \tilde{\ell}_r$

Case (I$\oplus$). Then $t^{\widetilde{S}} = b_{1\ell_1} \oplus^{\tilde{\ell}} b_{2\ell_2}$. By construction we can suppose that $\tilde{\ell} = \tilde{\ell}_1' \widetilde{\Upsilon} \tilde{\ell}_2'$, then

$$(\text{I}\oplus)\ \frac{\begin{array}{c}\varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c \\ \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} b_{1\ell_1} \in \mathsf{Bool}_{\tilde{\ell}_1} \qquad \varepsilon_1 \vdash \mathsf{Bool}_{\tilde{\ell}_1} \lesssim \mathsf{Bool}_{\tilde{\ell}_1'} \\ \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} b_{2\ell_2} \in \mathsf{Bool}_{\tilde{\ell}_2} \qquad \varepsilon_2 \vdash \mathsf{Bool}_{\tilde{\ell}_2} \lesssim \mathsf{Bool}_{\tilde{\ell}_2'}\end{array}}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 b_{1\ell_1} \oplus^{\tilde{\ell}} \varepsilon_2 b_{2\ell_2} \in \text{TERM}_{\mathsf{Bool}_{\tilde{\ell}}}}$$

Therefore

$$\begin{array}{cc} & \varepsilon_1(b_1)_{\tilde{\ell}_1} \oplus^{\tilde{\ell}} \varepsilon_2(b_2)_{\tilde{\ell}_2} \mid \mu \\ \overset{e\ell}{\longrightarrow}_{\tilde{\ell}_c} & (\varepsilon_1 \widetilde{\Upsilon} \varepsilon_2)(b_1 \ [\![\oplus]\!]\ b_2)_{(\tilde{\ell}_1 \widetilde{\Upsilon} \tilde{\ell}_2)} :: \mathsf{Bool}_{\tilde{\ell}} \mid \mu \end{array}$$

Then

$$(\text{I}\oplus)\ \frac{\varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} (\varepsilon_1 \widetilde{\Upsilon} \varepsilon_2)(b_1 \ [\![\oplus]\!]\ b_2)_{(\tilde{\ell}_1 \widetilde{\Upsilon} \tilde{\ell}_2)} \in \text{TERM}_{\mathsf{Bool}_{\tilde{\ell}}}}$$

and the result holds.

Case (Iprot). Then $t^{\widetilde{S}} = e\ell \overset{\tilde{\ell}_c}{\triangleright} \text{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell\tilde{\ell}'} \varepsilon u$ and

$$(\text{Iprot})\ \frac{\begin{array}{c}\varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c \qquad \varepsilon'_r \vdash \widetilde{\tilde{\ell}_r \Upsilon \tilde{\ell}' \preccurlyeq \tilde{\ell}'_c} \\ \varepsilon'_r(\tilde{\ell}_r \widetilde{\Upsilon} \tilde{\ell}') \overset{\tilde{\ell}'_c}{\triangleright} u \in \text{TERM}_{\widetilde{S}'} \\ \varepsilon \vdash \widetilde{S}' \lesssim \widetilde{S} \qquad \varepsilon_\ell \vdash \tilde{\ell}' \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}\end{array}}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \text{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell\tilde{\ell}'} \varepsilon u \in \text{TERM}_{\widetilde{S}\widetilde{\Upsilon}\tilde{\ell}}}$$

Therefore

$$\text{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell\tilde{\ell}'} \varepsilon u \mid \mu \ \overset{e\ell}{\longrightarrow}_{\tilde{\ell}_c}\ (\varepsilon \widetilde{\Upsilon} \varepsilon_\ell)(u \widetilde{\Upsilon} \tilde{\ell}') :: \widetilde{S} \widetilde{\Upsilon} \tilde{\ell} \mid \mu$$

But by Lemma 36, $e\ell \overset{\tilde{\ell}_c}{\triangleright} u \in \text{TERM}_{\widetilde{S}'}$. Therefore by definition of join $e\ell \overset{\tilde{\ell}_c}{\triangleright} (u \widetilde{\Upsilon} \tilde{\ell}') \in \text{TERM}_{\widetilde{S}'\widetilde{\Upsilon}\tilde{\ell}'}$. Then using Lemma 7

$$\text{I::}\ \frac{\begin{array}{c}e\ell \overset{\tilde{\ell}_c}{\triangleright} (u \widetilde{\Upsilon} \tilde{\ell}') \in \text{TERM}_{\widetilde{S}'\widetilde{\Upsilon}\tilde{\ell}'} \\ (\varepsilon \widetilde{\Upsilon} \varepsilon_\ell) \vdash \widetilde{\widetilde{S}' \Upsilon \tilde{\ell}'} \lesssim \widetilde{S} \widetilde{\Upsilon} \tilde{\ell}\end{array}}{e\ell \overset{\tilde{\ell}_c}{\triangleright} (\varepsilon \widetilde{\Upsilon} \varepsilon_\ell)(u \widetilde{\Upsilon} \tilde{\ell}') :: \widetilde{S} \widetilde{\Upsilon} \tilde{\ell} \in \text{TERM}_{S\Upsilon\ell}}$$

and the result holds.

Case (Iapp). Then $t^{\widetilde{S}} = \varepsilon_1(\lambda^{\tilde{\ell}''_c} x^{\widetilde{S}_{11}}.t^{\widetilde{S}_{12}})_{\tilde{\ell}_1} \mathbin{@}^{\widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c} \tilde{\ell}\widetilde{S}_2}_{\varepsilon_\ell} \varepsilon_2 u$ and $\widetilde{S} = \widetilde{S}_2 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}$. Then

$$\text{(Iapp)}\ \frac{\dfrac{\mathcal{D}_1}{\varepsilon'_r \tilde{\ell}''_c \overset{\tilde{\ell}''_c}{\triangleright} t^{\widetilde{S}12} \in \text{TERM}_{\widetilde{S}_{12}}} \qquad \varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} (\lambda^{\tilde{\ell}''_c} x^{\widetilde{S}_{11}}.t^{\widetilde{S}_{12}})_{\tilde{\ell}_1} \in \text{TERM}_{\widetilde{S}_{11} \xrightarrow{\tilde{\ell}''_c}_{\tilde{\ell}_1} \widetilde{S}_{12}}} \qquad \frac{\mathcal{D}_2}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} u \in \text{TERM}_{\widetilde{S}'_2}} \qquad \varepsilon_2 \vdash \widetilde{S}'_2 \lesssim \widetilde{S}_1 \qquad \varepsilon_1 \vdash \widetilde{S}_{11} \xrightarrow{\tilde{\ell}''_c}_{\tilde{\ell}_1} \widetilde{S}_{12} \lesssim \widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_2 \qquad \widetilde{\varepsilon_\ell \vdash \tilde{\ell}_c \mathbin{\Upsilon} \tilde{\ell} \preccurlyeq \tilde{\ell}'_c} \qquad \varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1(\lambda^{\tilde{\ell}''_c} x^{\widetilde{S}_{11}}.t^{\widetilde{S}_{12}})_{\tilde{\ell}_1} \mathbin{@}^{\widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c} \tilde{\ell}\widetilde{S}_2}_{\varepsilon_\ell} \varepsilon_2 u \in \text{TERM}_{\widetilde{S}_2 \widetilde{\Upsilon} \tilde{\ell}}}$$

If $\varepsilon' = (\varepsilon_2 \circ^{<:} idom(\varepsilon_1))$ or $\varepsilon'_r = (\varepsilon_r \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon_1)) \circ^{\preccurlyeq} \varepsilon_\ell \circ^{\preccurlyeq} ilat(\varepsilon_1)$ are not defined, then $t^{\widetilde{S}} \mid \mu \xrightarrow[\tilde{\ell}_c]{e\ell}$ **error**, and then the result hold immediately. Suppose that consistent transitivity does hold, then

$$\varepsilon_1(\lambda^{\tilde{\ell}''_c} x^{\widetilde{S}_{11}}.t^{\widetilde{S}_{12}})_{\tilde{\ell}_1} \mathbin{@}^{\widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c} \tilde{\ell}\widetilde{S}_2}_{\varepsilon_\ell} \varepsilon_2 u \mid \mu \xrightarrow[\tilde{\ell}_c]{\varepsilon_r \tilde{\ell}_r} \text{prot}^{\tilde{\ell}''_c, \tilde{\ell}, \widetilde{S}_2}_{\varepsilon'_r, ilbl(\varepsilon_1)\tilde{\ell}_1} icod(\varepsilon_1)([(\varepsilon' u :: \widetilde{S}_{11})/x^{\widetilde{S}_{11}}]t^{\widetilde{S}_{12}}) \mid \mu$$

As $\varepsilon_2 \vdash \widetilde{S}'_2 \lesssim \widetilde{S}_1$ and by inversion lemma $idom(\varepsilon_1) \vdash \widetilde{S}_1 \lesssim \widetilde{S}_{11}$, then $\varepsilon' \vdash \widetilde{S}'_2 \lesssim \widetilde{S}_{11}$ and . Therefore
$\varepsilon'_r \tilde{\ell}''_c \overset{\tilde{\ell}''_c}{\triangleright} \varepsilon' u :: \widetilde{S}_{11} \in \text{TERM}_{\widetilde{S}_{11}}$, and by Lemma 38,
$\varepsilon'_r \tilde{\ell}''_c \overset{\tilde{\ell}''_c}{\triangleright} [(\varepsilon' u :: \widetilde{S}_{11})/x^{\widetilde{S}_{11}}]t^{\widetilde{S}_{12}} \in \text{TERM}_{\widetilde{S}_{12}}$.

We know that $\varepsilon_\ell \vdash \widetilde{\tilde{\ell}_c \mathbin{\Upsilon} \tilde{\ell} \preccurlyeq \tilde{\ell}'_c}$. By inversion on the label of types, $ilbl(\varepsilon_1) \vdash \tilde{\ell}_1 \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}$. Also by monotonicity of the join, $\varepsilon_r \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon_1) \vdash \tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_1 \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c \mathbin{\widetilde{\Upsilon}} \tilde{\ell}$. Then, by inversion on the latent effect of function types, $ilat(\varepsilon_1) \vdash \tilde{\ell}'_c \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}''_c$. Therefore combining evidences, as $\varepsilon'_r = (\varepsilon_r \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon_1)) \circ^{\preccurlyeq} \varepsilon_\ell \circ^{\preccurlyeq} ilat(\varepsilon_1)$, we may justify the runtime judgment $\varepsilon'_r \vdash \widetilde{\tilde{\ell}_r \mathbin{\Upsilon} \tilde{\ell}_1 \preccurlyeq \tilde{\ell}''_c}$.

Let us call $t'^{\widetilde{S}_{12}} = [(\varepsilon' u :: \widetilde{S}_{11})/x^{\widetilde{S}_{11}}]t^{\widetilde{S}_{12}}$. By Lemma 35, $\varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_1) \overset{\tilde{\ell}''_c}{\triangleright} t'^{\widetilde{S}_{12}} \in \text{TERM}_{\widetilde{S}_{12}}$. Then

$$\text{(Iprot)}\ \frac{\varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c \qquad \varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_1) \overset{\tilde{\ell}''_c}{\triangleright} t'^{\widetilde{S}_{12}} \in \text{TERM}_{\widetilde{S}_{12}} \qquad icod(\varepsilon_1) \vdash \widetilde{S}_{12} \lesssim \widetilde{S}_2 \qquad ilbl(\varepsilon_1) \vdash \tilde{\ell}_1 \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}}{e\ell \overset{\tilde{\ell}_c}{\triangleright} \text{prot}^{\tilde{\ell}''_c, \tilde{\ell}, \widetilde{S}_2}_{\varepsilon'_r, ilbl(\varepsilon_1)\tilde{\ell}_1} icod(\varepsilon_1)(t'^{\widetilde{S}_{12}}) \in \text{TERM}_{\widetilde{S}_2 \widetilde{\Upsilon} \tilde{\ell}}}$$

and the result holds.

Case (Iif-true). Then $t^{\widetilde{S}} = \text{if}\ ^{\tilde{\ell}} \varepsilon_1 b_{\tilde{\ell}_1}\ \text{then}\ \varepsilon_2 t^{\widetilde{S}_2}\ \text{else}\ \varepsilon_3 t^{\widetilde{S}_3}$, $\widetilde{S} = (\widetilde{S}_2 \mathbin{\widetilde{\vee}} \widetilde{S}_3) \mathbin{\widetilde{\Upsilon}} \tilde{\ell}$ and

$$\text{(If)}\ \frac{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} b_{\tilde{\ell}_1} \in \text{TERM}_{\text{Bool}_{\tilde{\ell}_1}} \qquad \varepsilon_1 \vdash \text{Bool}_{\tilde{\ell}_1} \lesssim \text{Bool}_{\tilde{\ell}} \qquad e\ell' = (\varepsilon_r \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon_1))(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_1) \qquad \varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c \qquad e\ell' \overset{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}}{\triangleright} t^{\widetilde{S}_2} \in \text{TERM}_{\widetilde{S}_2} \qquad \varepsilon_2 \vdash \widetilde{S}_2 \lesssim (\widetilde{S}_2 \mathbin{\widetilde{\vee}} \widetilde{S}_3) \qquad e\ell' \overset{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}}{\triangleright} t^{\widetilde{S}_3} \in \text{TERM}_{\widetilde{S}_3} \qquad \varepsilon_3 \vdash \widetilde{S}_3 \lesssim (\widetilde{S}_2 \mathbin{\widetilde{\vee}} \widetilde{S}_3)}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \text{if}\ ^{\tilde{\ell}} \varepsilon_1 b_{\tilde{\ell}_1}\ \text{then}\ \varepsilon_2 t^{\widetilde{S}_2}\ \text{else}\ \varepsilon_3 t^{\widetilde{S}_3} \in \text{TERM}_{(\widetilde{S}_2 \mathbin{\widetilde{\vee}} \widetilde{S}_3) \widetilde{\Upsilon} \tilde{\ell}}}$$

Therefore

$$\text{if}\ ^{\tilde{\ell}} \varepsilon_1 b_{\tilde{\ell}_1}\ \text{then}\ \varepsilon_2 t^{\widetilde{S}_2}\ \text{else}\ \varepsilon_3 t^{\widetilde{S}_3} \mid \mu \xrightarrow[\tilde{\ell}_c]{\varepsilon_r \tilde{\ell}_r} \text{prot}^{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}, \tilde{\ell}, (\widetilde{S}_2 \mathbin{\widetilde{\vee}} \widetilde{S}_3)}_{(\varepsilon_r \widetilde{\Upsilon} ilbl(\varepsilon_1)), ilbl(\varepsilon_1)\tilde{\ell}_1} \varepsilon_2 t^{\widetilde{S}_2} \mid \mu$$

But

$$\text{(Iprot)}\ \frac{\varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c \qquad (\varepsilon_r \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon_1))(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_1) \overset{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}}{\triangleright} t^{\widetilde{S}_2} \in \text{TERM}_{\widetilde{S}_2} \qquad \varepsilon_2 \vdash \widetilde{S}_2 \lesssim \widetilde{S}_2 \mathbin{\widetilde{\vee}} \widetilde{S}_3 \qquad ilbl(\varepsilon_1) \vdash \tilde{\ell}_1 \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \text{prot}^{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}, \tilde{\ell}, (\widetilde{S}_2 \mathbin{\widetilde{\vee}} \widetilde{S}_3)}_{(\varepsilon_r \widetilde{\Upsilon} ilbl(\varepsilon_1)), ilbl(\varepsilon_1)\tilde{\ell}_1} \varepsilon_2 t^{\widetilde{S}_2} \in \text{TERM}_{(\widetilde{S}_2 \mathbin{\widetilde{\vee}} \widetilde{S}_3) \widetilde{\Upsilon} \tilde{\ell}}}$$

and the result holds.

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

Case (Iref). Then $t^{\widetilde{S}} = \text{ref}^{\widetilde{S}'}_{\varepsilon_\ell}\ \varepsilon u$ and

$$\text{(Iref)}\ \frac{\varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c \qquad \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} u \in \text{TERM}_{\widetilde{S}''} \qquad \varepsilon \vdash \widetilde{S}'' \lesssim \widetilde{S}' \qquad \varepsilon_\ell \vdash \tilde{\ell}_c \mathrel{\widetilde{\preccurlyeq}} label(\widetilde{S}')}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \text{ref}^{\widetilde{S}'}_{\varepsilon_\ell}\ \varepsilon u \in \text{TERM}_{\text{Ref}_\perp\ \widetilde{S}'}}$$

If $\varepsilon' = \varepsilon \mathbin{\widetilde{\Upsilon}} (\varepsilon_r \circ^{\preccurlyeq} \varepsilon_\ell)$ is not defined, then $t^{\widetilde{S}'} \mid \mu \xrightarrow[\tilde{\ell}_c]{e\ell}$ **error**, and then the result hold immediately. Suppose that consistent transitivity does hold, then

$$\text{ref}^{\widetilde{S}'}_{\varepsilon_\ell}\ \varepsilon u \mid \mu \xrightarrow[\tilde{\ell}_c]{\varepsilon_r \tilde{\ell}_r} l^{\widetilde{S}'}_\perp \mid \mu[l^{\widetilde{S}'} \mapsto \varepsilon'(u \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_r) :: \widetilde{S}']$$

where $l^{\widetilde{S}'} \notin dom(\mu)$.

We know that $\varepsilon_\ell \vdash \tilde{\ell}_c \mathrel{\widetilde{\preccurlyeq}} label(\widetilde{S}')$, therefore $\varepsilon_r \circ^{\preccurlyeq} \varepsilon_\ell \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} label(\widetilde{S}')$. We also know that $\varepsilon \vdash \widetilde{S}'' \mathrel{\widetilde{\lesssim}} \widetilde{S}'$. Therefore combining both evidences we can justify that $\varepsilon \mathbin{\widetilde{\Upsilon}} (\varepsilon_r \circ^{<:} \varepsilon_\ell) \vdash \widetilde{\widetilde{S}'_2 \mathbin{\Upsilon} \tilde{\ell}_r <: \widetilde{S}'}$. But

$$\text{(I::)}\ \frac{\text{(Il)}\ \dfrac{\varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c}{l^{\widetilde{S}'}_\perp \in \text{TERM}_{\text{Ref}_\perp\ \widetilde{S}'}} \qquad \varepsilon'' \vdash \text{Ref}_{\tilde{\ell}_r}\ \widetilde{S}' \lesssim \text{Ref}_{\tilde{\ell}_c}\ \widetilde{S}' \qquad \varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon'' l^{\widetilde{S}'}_{\tilde{\ell}_r} :: \text{Ref}_{\tilde{\ell}_c}\ \widetilde{S}' \in \text{TERM}_{\text{Ref}_\perp\ \widetilde{S}'}}$$

Let us call $\mu' = \mu[l^{\widetilde{S}'} \mapsto \varepsilon'(u \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_r) :: \widetilde{S}']$. It is easy to see that $freeLocs(l^{\widetilde{S}'}) = l^{\widetilde{S}'}$ and $dom(\mu') = dom(\mu) \cup l^{\widetilde{S}'}$, then $freeLocs(l^{\widetilde{S}'}) \subseteq dom(\mu')$. Given that $t^{\widetilde{S}'} \vdash \mu$ then $freeLocs(u) \subseteq dom(\mu)$, and therefore $\forall\, v \in cod(\mu') = cod(\mu) \cup (\varepsilon'(u \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_r) :: \widetilde{S}'), freeLocs(v') \subseteq dom(\mu')$. Finally as $t^{\widetilde{S}'} \vdash \mu$

and $\mu'(l^{\widetilde{S'}}) = \varepsilon'(u \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_r) :: \widetilde{S'} \in \text{TERM}_{\widetilde{S'}}$ then we can conclude that $l^{\widetilde{S'}} \vdash \mu'$ and $dom(\mu) \subseteq dom(\mu')$, and the result holds.

Case (Ideref). Then $t^{\widetilde{S}} = !^{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S'}} \varepsilon l^{\widetilde{S''}}_{\tilde{\ell}'}$, $\widetilde{S} = \widetilde{S'} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}$ and

$$\text{(Ideref)}\ \frac{\begin{array}{c}\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} l^{\widetilde{S''}}_{\tilde{\ell}'} \in \text{TERM}_{\mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S''}} \\ \varepsilon \vdash \mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S''} \lesssim \mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S'} \\ \varepsilon_r \vdash \tilde{\ell}_r \preccurlyeq \tilde{\ell}_c\end{array}}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} !^{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S'}} \varepsilon l^{\widetilde{S''}}_{\tilde{\ell}'} \in \text{TERM}_{\widetilde{S'} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}}}$$

Then

$$!^{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S'}} \varepsilon l^{\widetilde{S''}}_{\tilde{\ell}'} \mid \mu \xrightarrow{\varepsilon_r \tilde{\ell}_r} \tilde{\ell}_c\ \mathsf{prot}^{\tilde{\ell}_c \mathbin{\widetilde{\Upsilon}} \tilde{\ell}, \tilde{\ell}, \widetilde{S'}}_{(\varepsilon_r \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon)), ilbl(\varepsilon) \tilde{\ell}'} iref(\varepsilon) v$$

where $\mu(l^{\widetilde{S''}}) = v$. As the store is well typed, we choose $\tilde{\ell}_r, \tilde{\ell}_c$ and ε_r, and therefore $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v \in \text{TERM}_{\widetilde{S''}}$. By Lemma 36, $(\varepsilon_r \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon))(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}') \overset{\tilde{\ell}_c \mathbin{\widetilde{\Upsilon}} \tilde{\ell}}{\triangleright} v \in \text{TERM}_{\widetilde{S''}}$. By inversion lemma on references, $ilbl(\varepsilon) \vdash \tilde{\ell}' \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}$ and $iref(\varepsilon) \vdash \widetilde{S''} \mathrel{\widetilde{\lesssim}} \widetilde{S'}$

$$\text{(Iprot)}\ \frac{\begin{array}{c}\varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c \qquad (\varepsilon_r \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon))(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}') \overset{\tilde{\ell}_c \mathbin{\widetilde{\Upsilon}} \tilde{\ell}}{\triangleright} v \in \text{TERM}_{\widetilde{S''}} \\ iref(\varepsilon) \vdash \widetilde{S''} \mathrel{\widetilde{\lesssim}} \widetilde{S'} \qquad ilbl(\varepsilon) \vdash \tilde{\ell}' \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}\end{array}}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}_c \mathbin{\widetilde{\Upsilon}} \tilde{\ell}, \tilde{\ell}, \widetilde{S'}}_{(\varepsilon_r \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon)), ilbl(\varepsilon) \tilde{\ell}'} iref(\varepsilon) v \in \text{TERM}_{\widetilde{S'} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}}}$$

and the result holds.

Case (Iassgn). Then $t^{\widetilde{S}} = l_\ell := v$ and

$$\text{(Iassgn)}\ \frac{\begin{array}{cc}\varepsilon_1 \vdash \mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S}'_1 \lesssim \mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1 & \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} l^{\widetilde{S}'_1}_{\tilde{\ell}'} \in \text{TERM}_{\mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S}'_1} \\ \varepsilon_2 \vdash \widetilde{S}_2 \lesssim \widetilde{S}_1 & \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} u \in \text{TERM}_{\widetilde{S}_2} \\ \varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c & \widetilde{\varepsilon_\ell \vdash \tilde{\ell}_c \mathbin{\widetilde{\Upsilon}} \tilde{\ell} \preccurlyeq label(\widetilde{S}_1)}\end{array}}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 l^{\widetilde{S}'_1}_{\tilde{\ell}'} \overset{\varepsilon_\ell}{:=} \varepsilon_2 u \in \text{TERM}_{\mathsf{Unit}_\perp}}$$

If $\varepsilon' = (iref(\varepsilon_1) \circ^{<:} \varepsilon_2) \mathbin{\widetilde{\Upsilon}} ((\varepsilon_r \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon_1)) \circ^{\preccurlyeq} \varepsilon_\ell \circ^{\preccurlyeq} ilbl(iref(\varepsilon_1))$ is not defined, then $t^{\widetilde{S'}} \mid \mu \xrightarrow{e\ell}_{\tilde{\ell}_c}$ **error**, and then the result hold immediately. Suppose that consistent transitivity does hold, then

$$\varepsilon_1 l^{\widetilde{S}}_{\tilde{\ell}} \overset{\varepsilon_\ell}{:=} \varepsilon_2 u \mid \mu \xrightarrow{\varepsilon_r \tilde{\ell}_r}_{\tilde{\ell}_c} \mathsf{unit}_\perp \mid \mu[l^{\widetilde{S}} \mapsto \varepsilon'(u \mathbin{\widetilde{\Upsilon}} (\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}))] :: \widetilde{S}]$$

We know that $\varepsilon_\ell \vdash \widetilde{\tilde{\ell}_c \mathbin{\widetilde{\Upsilon}} \tilde{\ell} \preccurlyeq label(\widetilde{S}_1)}$. Then by inversion on reference evidence types and inversion in the label of types, $ilbl(iref(\varepsilon_1)) \vdash label(\widetilde{S}_1) \mathrel{\widetilde{\preccurlyeq}} label(\widetilde{S}'_1)$. But $ilbl(\varepsilon_1) \vdash \tilde{\ell}' \preccurlyeq \tilde{\ell}$, using monotonicity of the join, $\varepsilon_r \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon_1) \vdash \widetilde{\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}' \preccurlyeq \tilde{\ell}_c \mathbin{\widetilde{\Upsilon}} \tilde{\ell}}$. Therefore $((\varepsilon_r \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon_1)) \circ^{\preccurlyeq} \varepsilon_\ell) \circ^{\preccurlyeq} ilbl(iref(\varepsilon_1)) \vdash \widetilde{\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}' \preccurlyeq label(\widetilde{S}'_1)}$. We also know that if $u \in \text{TERM}_{\widetilde{S}_2}$, then $(iref(\varepsilon_1) \circ^{<:} \varepsilon_2) \vdash \widetilde{S}_2 \lesssim \widetilde{S}'_1$. Combining both evidences, $\varepsilon' = (iref(\varepsilon_1) \circ^{<:} \varepsilon_2) \mathbin{\widetilde{\Upsilon}} (((\varepsilon_r \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon_1)) \circ^{\preccurlyeq} \varepsilon_\ell) \circ^{\preccurlyeq} ilbl(iref(\varepsilon_1)))$, and by Proposition 7 we can then justify that $\varepsilon' \vdash \widetilde{S}_2 \mathbin{\widetilde{\Upsilon}} \widetilde{(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}) <: \widetilde{S}}$ and therefore justify the ascription in the heap.

Let us call $\mu' = \mu[l^{\widetilde{S}} \mapsto \varepsilon'(u \mathbin{\widetilde{\Upsilon}} (\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell})) :: \widetilde{S}]$. As $freeLocs(\mathsf{unit}_\perp) = \emptyset$ then $freeLocs(\mathsf{unit}_\perp) \subseteq \mu'$.

As $t^{\widetilde{S}} \vdash \mu$ then $freeLocs(u) \in dom(\mu)$, and as $dom(\mu) = dom(\mu')$ then it is trivial to see that $\forall\, v' \in cod(\mu'), freeLocs(v') \subseteq dom(\mu')$, and the result holds.

□

Proposition 40 ($\longmapsto$ is well defined). *If* $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ *and* $t^{\widetilde{S}} \vdash \mu$, *then* $r \in \text{CONFIG}_{\widetilde{S}} \cup \{\,\textbf{error}\,\}$ *and if* $r = t'^{\widetilde{S}} \mid \mu' \in \text{CONFIG}_{\widetilde{S}}$ *then also* $t'^{\widetilde{S}} \mid \mu'$ *and* $dom(\mu) \subseteq dom(\mu')$.

Proof. By induction on the structure of a derivation of $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$.

Case (R$\longrightarrow$). $t^{\widetilde{S}} \mid \mu \xrightarrow{e\ell}_{\tilde{\ell}_c} r$. By well-definedness of $\longrightarrow$ (Prop 39), $r \in \text{CONFIG}_{\widetilde{T}} \cup \{\,\textbf{error}\,\}$ and if $r = t'^{\widetilde{S}} \mid \mu' \in \text{CONFIG}_{\widetilde{S}}$ then also $t'^{\widetilde{S}} \vdash \mu'$ and $dom(\mu) \subseteq dom(\mu')$.

Case (Rprot). $t^{\widetilde{S}} = \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}, \widetilde{S'}}_{\varepsilon'_r, \varepsilon_\ell \tilde{\ell}'} \varepsilon t^{\widetilde{S''}}_1$ and

$$\text{(Iprot)}\ \frac{\begin{array}{c}\varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c \qquad \varepsilon'_r \vdash \widetilde{\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}' \preccurlyeq \tilde{\ell}'_c} \\ \varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}') \overset{\tilde{\ell}'_c}{\triangleright} t^{\widetilde{S''}}_1 \in \text{TERM}_{\widetilde{S''}} \\ \varepsilon \vdash \widetilde{S''} \lesssim \widetilde{S'} \qquad \varepsilon_\ell \vdash \tilde{\ell}' \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}\end{array}}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}, \widetilde{S'}}_{\varepsilon'_r, \varepsilon_\ell \tilde{\ell}'} \varepsilon t^{\widetilde{S''}}_1 \in \text{TERM}_{\widetilde{S'} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}}}$$

and

$$\text{(Rprot)}\ \frac{t^{\widetilde{S''}}_1 \mid \mu \overset{\varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}')}{\longmapsto}_{\tilde{\ell}'_c} t^{\widetilde{S''}}_2 \mid \mu'}{\mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}, \widetilde{S'}}_{\varepsilon'_r, \varepsilon_\ell \tilde{\ell}'} \varepsilon t^{\widetilde{S''}}_1 \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}, \widetilde{S'}}_{\varepsilon'_r, \varepsilon_\ell \tilde{\ell}'} \varepsilon t^{\widetilde{S''}}_2 \mid \mu'}$$

Using induction hypothesis on the premise of (Rprot), $\varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}') \overset{\tilde{\ell}'_c}{\triangleright} t^{\widetilde{S''}}_2 \in \text{TERM}_{\widetilde{S''}}$, $t^{\widetilde{S''}}_2 \vdash \mu'$ and $dom(\mu) \subseteq dom(\mu')$. Therefore

$$\text{(Iprot)}\ \frac{\begin{array}{c}\varepsilon_r \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_c \qquad \varepsilon'_r \vdash \widetilde{\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}' \preccurlyeq \tilde{\ell}'_c} \\ \varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}') \overset{\tilde{\ell}'_c}{\triangleright} t^{\widetilde{S''}}_2 \in \text{TERM}_{\widetilde{S''}} \\ \varepsilon \vdash \widetilde{S''} \lesssim \widetilde{S'} \qquad \varepsilon_\ell \vdash \tilde{\ell}' \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}\end{array}}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}, \widetilde{S'}}_{\varepsilon'_r, \varepsilon_\ell \tilde{\ell}'} \varepsilon t^{\widetilde{S''}}_2 \in \text{TERM}_{\widetilde{S'} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}}}$$

and the result holds.

Case (Rf). $t^{\widetilde{S}} = f[t^{\widetilde{S'}}_1]$, $e\ell \overset{\tilde{\ell}_c}{\triangleright} f[t^{\widetilde{S'}}] \in \text{TERM}_{\widetilde{S}}$, $t^{\widetilde{S'}}_1 \mid \mu \xrightarrow{e\ell}_{\tilde{\ell}_c} t^{\widetilde{S'}}_2 \mid \mu'$, and consider $F : \text{TERM}_{\widetilde{S'}} \to \text{TERM}_{\widetilde{S}}$, where $F(e\ell \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S'}}) = e\ell \overset{\tilde{\ell}_c}{\triangleright} f[t^{\widetilde{S'}}]$. By induction hypothesis, $e\ell \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S'}}_2 \in \text{TERM}_{\widetilde{S'}}$, so $F(e\ell \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S'}}_2) = e\ell \overset{\tilde{\ell}_c}{\triangleright} f[t^{\widetilde{S'}}_2] \in \text{TERM}_{\widetilde{S}}$.

By induction hypothesis we also know that $t^{\widetilde{S'}}_2 \vdash \mu'$.

If $freeLocs(t^{\widetilde{S'}}_2) \subseteq \mu'$, $freeLocs(f[t^{\widetilde{S}}_1]) \subseteq \mu$, and $dom(\mu) \subseteq dom(\mu')$, then it is easy to see that $freeLocs(f[t^{\widetilde{S'}}_2]) \subseteq \mu'$, and therefore conclude that $f[t^{\widetilde{S}_2}] \vdash \mu'$.

Case (R$f$err, R$g$err, Rprot$f$err, Rprot$g$err). $r = \textbf{error}$.

Case (Rg). $t^{\widetilde{S}} = g[et]$, $e\ell \overset{\tilde{\ell}_c}{\triangleright} g[t^{\widetilde{S'}}] \in \text{TERM}_{\widetilde{S}}$, and consider $G : \text{EVLABEL} \times \text{GLABEL} \times \text{EVTERM} \to \text{TERM}_{\widetilde{S}}, G(e\ell, \tilde{\ell}_c, et) = e\ell \overset{\tilde{\ell}_c}{\triangleright} g[et]$ and $et \longrightarrow_c et'$. Then there exists $\widetilde{S}_e, \widetilde{S}_x$ such that $et = \varepsilon_e t^{\widetilde{S}_e}_e$ and $\varepsilon_e \vdash \widetilde{S}_e \lesssim \widetilde{S}_x$. Also, $t_e = \varepsilon_v v :: \widetilde{S}_e$, with $v \in \text{TERM}_{\widetilde{S}_v}$ and $\varepsilon_v \vdash \widetilde{S}_v \sim \widetilde{S}_e$.

We know that $\varepsilon_c = \varepsilon_v \circ^{<:} \varepsilon_e$ is defined, and $et = \varepsilon_e t_e \longrightarrow_c$

$\varepsilon_c v = et'$. By definition of $\circ^{<:}$ we have $\varepsilon_c \vdash \widetilde{S}_v \lesssim \widetilde{S}_x$, so $G(e\ell, \tilde{\ell}_c, et') = e\ell \overset{\tilde{\ell}_c}{\triangleright} g[et'] \in \text{TERM}_{\widetilde{S}}$.
As $freeLocs(et) = freeLocs(et')$ and $\mu' = \mu$ then it is easy to conclude that $g[et'] \vdash \mu$.

Case (R$prot g$). Similar case to (Rg) case, using $P : \text{EVTERM} \to \text{TERM}_{\widetilde{S}}$, $P(et) = e\ell \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell\tilde{\ell}'} et$.

□

Now we can stablish type safety: programs do not get stuck, though they may terminate with cast errors. Also the store of a program is well typed.

Proposition 8 (Type Safety). *If $e\ell \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}} \in \text{TERM}_{\widetilde{S}}$ then one of the following is true:*

1. *$t^{\widetilde{S}}$ is a value v;*
2. *if $t^{\widetilde{S}} \vdash \mu$ then $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} t'^{\widetilde{S}} \mid \mu'$ for some term $e\ell \overset{\tilde{\ell}_c}{\triangleright} t'^{\widetilde{S}} \in \text{TERM}_{\widetilde{S}}$ and some μ' such that $t'^{\widetilde{S}} \vdash \mu'$ and $dom(\mu) \subseteq dom(\mu')$;*
3. *$t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c}$ **error**.*

Proof. By induction on the structure of $e\ell \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}}$. For all cases consider $e\ell = \varepsilon_r\tilde{\ell}_r$.

Case (Iu,Il, Ib, Ix, Iλ). $t^{\widetilde{S}}$ is a value.

Case (Iprot). $t^{\widetilde{S}} = \mathsf{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell\tilde{\ell}'} \varepsilon t^{\widetilde{S}'}$, and

$$\text{(Iprot)}\frac{\begin{array}{c}\varepsilon_r \vdash \tilde{\ell}_r \precsim \tilde{\ell}_c \qquad \varepsilon'_r \vdash \widetilde{\tilde{\ell}_r \curlyvee \tilde{\ell}' \preccurlyeq \tilde{\ell}'_c} \\ \varepsilon'_r(\tilde{\ell}_r \curlyvee \tilde{\ell}') \overset{\tilde{\ell}'_c}{\triangleright} t^{\widetilde{S}'} \in \text{TERM}_{\widetilde{S}'} \\ \varepsilon \vdash \widetilde{S}' \lesssim \widetilde{S} \qquad \varepsilon_\ell \vdash \tilde{\ell}' \precsim \tilde{\ell}\end{array}}{\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}'_c,\tilde{\ell},\widetilde{S}}_{\varepsilon'_r,\varepsilon_\ell\tilde{\ell}'} \varepsilon t^{\widetilde{S}'} \in \text{TERM}_{\widetilde{S}\curlyvee\tilde{\ell}}}$$

By induction hypothesis on $t^{\widetilde{S}'}$, one of the following holds:

1. $t^{\widetilde{S}'}$ is a simple value , then by (R$\longrightarrow$), $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} v \mid \mu$, and by Prop 40, $e\ell \overset{\tilde{\ell}_c}{\triangleright} v \in \text{TERM}_{\widetilde{S}}$ and the result holds.
2. $t^{\widetilde{S}'}$ is an ascribed value v, then, $\varepsilon t^{\widetilde{S}'} \longrightarrow_c et'$ for some $et' \in \text{EVTERM} \cup \{\,\textbf{error}\,\}$. Hence $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{CONFIG}_{\widetilde{S}} \cup \{\,\textbf{error}\,\}$ by Prop 40 and either (R$prot$g), or (R$prot$gerr).
3. $t^{\widetilde{S}'} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r_1$ for some $r_1 \in \text{TERM}_{\widetilde{S}_1} \cup \{\,\textbf{error}\,\}$. Hence $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{CONFIG}_{\widetilde{S}} \cup \{\,\textbf{error}\,\}$ by Prop 40 and either (R$prot$), or (R$prot$ferr).

Case (I::). $t^{\widetilde{S}} = \varepsilon_1 t^{\widetilde{S}_1} :: \widetilde{S}_2$, and

$$\text{(I::)}\frac{\begin{array}{c}\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}_1} \in \text{TERM}_{\widetilde{S}_1} \\ \varepsilon_1 \vdash \widetilde{S}_1 \lesssim \widetilde{S}_2 \qquad \varepsilon_r \vdash \tilde{\ell}_r \precsim \tilde{\ell}_c\end{array}}{\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 t^{\widetilde{S}_1} :: \widetilde{S}_2 \in \text{TERM}_{\widetilde{S}_2}}$$

By induction hypothesis on $t^{\widetilde{S}_1}$, one of the following holds:

1. $t^{\widetilde{S}_1}$ is a value, in which case $t^{\widetilde{S}}$ is also a value.
2. $t^{\widetilde{S}_1} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r_1$ for some $r_1 \in \text{TERM}_{\widetilde{S}_1} \cup \{\,\textbf{error}\,\}$. Hence $t^{\widetilde{S}} \mid \mu \longmapsto r$ for some $r \in \text{CONFIG}_{\widetilde{S}} \cup \{\,\textbf{error}\,\}$ by Prop 40 and either (Rf), or (R$f$err).

Case (I$\widetilde{S}$if). $t^{\widetilde{S}} = \mathsf{if}\ ^{\tilde{\ell}}\varepsilon_1 t^{\widetilde{S}_1}\ \mathsf{then}\ \varepsilon_2 t^{\widetilde{S}_2}\ \mathsf{else}\ \varepsilon_3 t^{\widetilde{S}_3}$ and

$$\text{(If)}\frac{\begin{array}{c}\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}_1} \in \text{TERM}_{\widetilde{S}_1} \qquad \varepsilon_1 \vdash \widetilde{S}_1 \lesssim \mathsf{Bool}_{\tilde{\ell}} \qquad \varepsilon_r \vdash \tilde{\ell}_r \precsim \tilde{\ell}_c \\ e\ell' = (\varepsilon_r \widetilde{\curlyvee}\, ilbl(\varepsilon_1))(\tilde{\ell}_r \widetilde{\curlyvee}\, label(\widetilde{S}_1)) \\ e\ell' \overset{\tilde{\ell}_c}{\triangleright} \widetilde{\curlyvee\tilde{\ell}}\, t^{\widetilde{S}_2} \in \text{TERM}_{\widetilde{S}_2} \qquad \varepsilon_2 \vdash \widetilde{S}_2 \lesssim (\widetilde{S}_2 \,\widetilde{\curlyvee}\, \widetilde{S}_3) \\ e\ell' \overset{\tilde{\ell}_c}{\triangleright} \widetilde{\curlyvee\tilde{\ell}}\, t^{\widetilde{S}_3} \in \text{TERM}_{\widetilde{S}_3} \qquad \varepsilon_3 \vdash \widetilde{S}_3 \lesssim (\widetilde{S}_2 \,\widetilde{\curlyvee}\, \widetilde{S}_3)\end{array}}{\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{if}\ ^{\tilde{\ell}}\varepsilon_1 t^{\widetilde{S}_1}\ \mathsf{then}\ \varepsilon_2 t^{\widetilde{S}_2}\ \mathsf{else}\ \varepsilon_3 t^{\widetilde{S}_3} \in \text{TERM}_{\widetilde{(S_2 \,\widetilde{\curlyvee}\, S_3)}\widetilde{\curlyvee\tilde{\ell}}}}$$

By induction hypothesis on $t^{\widetilde{S}_1}$, one of the following holds:

1. $t^{\widetilde{S}_1}$ is a value u, then by (R$\longrightarrow$), $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ and $r \in \text{CONFIG}_{\widetilde{S}} \cup \{\,\textbf{error}\,\}$ by Prop 40.
2. $t^{\widetilde{S}_1}$ is an ascribed value v, then, $\varepsilon_1 t^{\widetilde{S}_1} \longrightarrow_c et'$ for some $et' \in \text{EVTERM} \cup \{\,\textbf{error}\,\}$. Hence $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{CONFIG}_{\widetilde{S}} \cup \{\,\textbf{error}\,\}$ by Prop 40 and either (R$prot$g), or (R$prot$gerr).
3. $t^{\widetilde{S}_1} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r_1$ for some $r_1 \in \text{TERM}_{\widetilde{S}_1} \cup \{\,\textbf{error}\,\}$. Hence $t^{\widetilde{S}} \mid \mu \longmapsto r$ for some $r \in \text{CONFIG}_{\widetilde{S}} \cup \{\,\textbf{error}\,\}$ by Prop 40 and either (Rf), or (R$f$err).

Case (Iapp). $t^{\widetilde{S}} = \varepsilon_1 t^{\widetilde{S}_1}\ @^{\widetilde{S}_{11} \overset{\tilde{\ell}'_c}{\longrightarrow} \tilde{\ell} \widetilde{S}_{12}}_{\varepsilon_\ell}\ \varepsilon_2 t^{\widetilde{S}_2}$

$$\text{(Iapp)}\frac{\begin{array}{c}\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}_1} \in \text{TERM}_{\widetilde{S}_1} \qquad \varepsilon_1 \vdash \widetilde{S}_1 \lesssim \widetilde{S}_{11} \overset{\tilde{\ell}'_c}{\longrightarrow} \tilde{\ell} \widetilde{S}_{12} \\ \varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}_2} \in \text{TERM}_{\widetilde{S}_2} \qquad \varepsilon_2 \vdash \widetilde{S}_2 \lesssim \widetilde{S}_{11} \\ \varepsilon_\ell \vdash \widetilde{\tilde{\ell}_c \curlyvee \tilde{\ell} \preccurlyeq \tilde{\ell}'_c} \qquad \varepsilon_r \vdash \tilde{\ell}_r \precsim \tilde{\ell}_c\end{array}}{\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 t^{\widetilde{S}_1}\ @^{\widetilde{S}_{11} \overset{\tilde{\ell}'_c}{\longrightarrow} \tilde{\ell} \widetilde{S}_{12}}_{\varepsilon_\ell}\ \varepsilon_2 t^{\widetilde{S}_2} \in \text{TERM}_{\widetilde{S}_{12}\widetilde{\curlyvee\tilde{\ell}}}}$$

By induction hypothesis on $t^{\widetilde{S}_1}$, one of the following holds:

1. $t^{\widetilde{S}_1}$ is a value $(\lambda x^{\widetilde{S}'_{11}}.t^{\widetilde{S}'_{12}})_{\tilde{\ell}'}$ (by canonical forms Lemma 37), posing $\widetilde{S}_1 = \widetilde{S}'_{11} \overset{\tilde{\ell}''}{\longrightarrow}_{\tilde{\ell}'} \widetilde{S}'_{12}$.
   Then by induction hypothesis on $t^{\widetilde{S}_2}$, one of the following holds:
   (a) $t^{\widetilde{S}_2}$ is a value u, then by (R$\longrightarrow$), $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ and $r \in \text{CONFIG}_{\widetilde{S}} \cup \{\,\textbf{error}\,\}$ by Prop 40.
   (b) $t^{\widetilde{S}_2}$ is an ascribed value v, then, $\varepsilon_2 t^{\widetilde{S}_2} \longrightarrow_c et'$ for some $et' \in \text{EVTERM} \cup \{\,\textbf{error}\,\}$. Hence $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{CONFIG}_{\widetilde{S}} \cup \{\,\textbf{error}\,\}$ by Prop 40 and either (R$prot$g), or (R$prot$gerr).
   (c) $t^{\widetilde{S}_2} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r_2$ for some $r_2 \in \text{CONFIG}_{\widetilde{S}_2} \cup \{\,\textbf{error}\,\}$. Hence $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{CONFIG}_{\widetilde{S}} \cup \{\,\textbf{error}\,\}$ by Prop 40 and either (Rf), or (R$f$err). Also by Prop 40, if $r = t'^{\widetilde{S}} \mid \mu' \in \text{TERM}_{\widetilde{S}}$ then also $t'^{\widetilde{S}} \mid \mu'$ and $dom(\mu) \subseteq dom(\mu')$.

2. $t^{\widetilde{S_1}}$ is an ascribed value v, then, $\varepsilon_1 t^{\widetilde{S_1}} \longrightarrow_c et'$ for some $et' \in \text{EvTerm} \cup \{\textbf{error}\}$. Hence $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{Config}_{\widetilde{S}} \cup \{\textbf{error}\}$ by Prop 40 and either (Rprotg), or (Rprotgerr).
3. $t^{\widetilde{S_1}} \mid \mu \longmapsto r_1$ for some $r_1 \in \text{Config}_{\widetilde{S_1}} \cup \{\textbf{error}\}$. Hence $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{Config}_{\widetilde{S}} \cup \{\textbf{error}\}$ by Prop 40 and either (Rf), or (R$f$err). Also by Prop 40, if $r = t'^{\widetilde{S}} \mid \mu' \in \text{Term}_{\widetilde{S}}$ then also $t'^{\widetilde{S}} \mid \mu'$ and $dom(\mu) \subseteq dom(\mu')$.

Case (I$\oplus$). Similar case to (Iapp)

Case (Iref). $t^{\widetilde{S}} = \mathsf{ref}^{\widetilde{S'}}_{\varepsilon_\ell}\, \varepsilon t^{\widetilde{S''}}$ and

$$\text{(Iref)}\ \frac{\begin{array}{cc}\varepsilon_r \vdash \tilde{\ell}_r \precsim \tilde{\ell}_c & \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S''}} \in \text{Term}_{\widetilde{S''}} \\ \varepsilon \vdash \widetilde{S''} \lesssim \widetilde{S'} & \varepsilon_\ell \vdash \tilde{\ell}_c \precsim label(\widetilde{S'})\end{array}}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{ref}^{\widetilde{S'}}_{\varepsilon_\ell}\, \varepsilon t^{\widetilde{S''}} \in \text{Term}_{\mathsf{Ref}_\perp\, \widetilde{S'}}}$$

By induction hypothesis on $t^{\widetilde{S''}}$, one of the following holds:

1. $t^{\widetilde{S''}}$ is a value v, then by (R$\longrightarrow$), $t^{\widetilde{S'}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ and $r \in \text{Config}_{\widetilde{S'}}$ by Prop 40. Also by Prop 40, if $r = t'^{\widetilde{S}} \mid \mu' \in \text{Term}_{\widetilde{S}}$ then also $t'^{\widetilde{S}} \mid \mu'$ and $dom(\mu) \subseteq dom(\mu')$.
2. $t^{\widetilde{S''}}$ is an ascribed value v, then, $\varepsilon t^{\widetilde{S'_1}} \longrightarrow_c et'$ for some $et' \in \text{EvTerm} \cup \{\textbf{error}\}$. Hence $t^{\widetilde{S'}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{Config}_{\widetilde{S'}} \cup \{\textbf{error}\}$ by Prop 40 and either (Rprotg), or (Rprotgerr).
3. $t^{\widetilde{S''}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r_1$ for some $r_1 \in \text{Config}_{\widetilde{S''}} \cup \{\textbf{error}\}$. Hence $t^{\widetilde{S'}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{Config}_{\widetilde{S'}} \cup \{\textbf{error}\}$ by Prop 40 and either (Rf), or (R$f$err). Also by Prop 40, if $r = t'^{\widetilde{S}} \mid \mu' \in \text{Term}_{\widetilde{S}}$ then also $t'^{\widetilde{S}} \mid \mu'$ and $dom(\mu) \subseteq dom(\mu')$.

Case (Ideref). $t^{\widetilde{S}} = !^{\mathsf{Ref}_{\tilde{\ell}}\, \widetilde{S'}}\, \varepsilon t^{\widetilde{S''}}$

$$\text{(Ideref)}\ \frac{\begin{array}{cc}\varepsilon_r \vdash \tilde{\ell}_r \precsim \tilde{\ell}_c & \\ \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S''}} \in \text{Term}_{\widetilde{S''}} & \varepsilon \vdash \widetilde{S''} \lesssim \mathsf{Ref}_{\tilde{\ell}}\, \widetilde{S'}\end{array}}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} !^{\mathsf{Ref}_{\tilde{\ell}}\, \widetilde{S'}}\, \varepsilon t^{\widetilde{S''}} \in \text{Term}_{\widetilde{S'} \widetilde{\gamma} \tilde{\ell}}}$$

By induction hypothesis on $t^{\widetilde{S''}}$, one of the following holds:

1. $t^{\widetilde{S''}}$ is a value $l^{\widetilde{S''_1}}$ (by canonical forms Lemma 37), where $\widetilde{S''} = \mathsf{Ref}_{\tilde{\ell'}}\, \widetilde{S'''}$, then by (R$\longrightarrow$), $t^{\widetilde{S'}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ and $r \in \text{Config}_{\widetilde{S'}}$ by Prop 40.
2. $t^{\widetilde{S''}}$ is an ascribed value v, then, $\varepsilon t^{\widetilde{S''}} \longrightarrow_c et'$ for some $et' \in \text{EvTerm} \cup \{\textbf{error}\}$. Hence $t^{\widetilde{S'}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{Config}_{\widetilde{S'}} \cup \{\textbf{error}\}$ by Prop 40 and either (Rprotg), or (Rprotgerr).
3. $t^{\widetilde{S''}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r_1$ for some $r_1 \in \text{Config}_{\widetilde{S''}} \cup \{\textbf{error}\}$. Hence $t^{\widetilde{S'}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{Config}_{\widetilde{S'}} \cup \{\textbf{error}\}$ by Prop 40 and either (Rf), or (R$f$err). Also by Prop 40, if $r = t'^{\widetilde{S}} \mid \mu' \in \text{Term}_{\widetilde{S}}$ then also $t'^{\widetilde{S}} \mid \mu'$ and $dom(\mu) \subseteq dom(\mu')$.

Case (I$\widetilde{S}$assign). $t^{\widetilde{S}} = \varepsilon_1 t^{\widetilde{S''_1}} \overset{\varepsilon_\ell}{:=} \varepsilon_2 t^{\widetilde{S}_2}$ and

$$\text{(Iassgn)}\ \frac{\begin{array}{cc}\varepsilon_1 \vdash \mathsf{Ref}_{\tilde{\ell'}}\, \widetilde{S'_1} \lesssim \mathsf{Ref}_{\tilde{\ell}}\, \widetilde{S}_1 & \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S''_1}} \in \text{Term}_{\mathsf{Ref}_{\tilde{\ell'}}\, \widetilde{S'_1}} \\ \varepsilon_2 \vdash \widetilde{S}_2 \lesssim \widetilde{S}_1 & \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}_2} \in \text{Term}_{\widetilde{S}_2} \\ \varepsilon_r \vdash \tilde{\ell}_r \precsim \tilde{\ell}_c & \varepsilon_\ell \vdash \tilde{\ell}_c \gamma \tilde{\ell} \precsim label(\widetilde{S}_1)\end{array}}{\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 t^{\widetilde{S''_1}} \overset{\varepsilon_\ell}{:=} \varepsilon_2 t^{\widetilde{S}_2} \in \text{Term}_{\mathsf{Unit}_\perp}}$$

By induction hypothesis on $t^{\widetilde{S''_1}}$, one of the following holds:

1. $t^{\widetilde{S''_1}}$ is a value $l^{\widetilde{S'''_1}}$ (by canonical forms Lemma 37), where $\widetilde{S''_1} = \mathsf{Ref}_{\tilde{\ell'}}\, \widetilde{S'''_1}$. Then by induction hypothesis on $t^{\widetilde{S}_2}$, one of the following holds:
   (a) $t^{\widetilde{S}_2}$ is a value u, then by (R$\longrightarrow$), $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ and $r \in \text{Config}_{\widetilde{S}} \cup \{\textbf{error}\}$ by Prop 40. Also by Prop 40, if $r = t'^{\widetilde{S}} \mid \mu' \in \text{Term}_{\widetilde{S}}$ then also $t'^{\widetilde{S}} \mid \mu'$ and $dom(\mu) \subseteq dom(\mu')$.
   (b) $t^{\widetilde{S}_2}$ is an ascribed value v, then, $\varepsilon_2 t^{\widetilde{S}_2} \longrightarrow_c et'$ for some $et' \in \text{EvTerm} \cup \{\textbf{error}\}$. Hence $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{Config}_{\widetilde{S}} \cup \{\textbf{error}\}$ by Prop 40 and either (Rprotg), or (Rprotgerr).
   (c) $t^{\widetilde{S}_2} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r_2$ for some $r_2 \in \text{Config}_{\widetilde{S}_2} \cup \{\textbf{error}\}$. Hence $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{Config}_{\widetilde{S}} \cup \{\textbf{error}\}$ by Prop 40 and either (Rf), or (R$f$err). Also by Prop 40, if $r = t'^{\widetilde{S}} \mid \mu' \in \text{Term}_{\widetilde{S}}$ then also $t'^{\widetilde{S}} \mid \mu'$ and $dom(\mu) \subseteq dom(\mu')$.
2. $t^{\widetilde{S''_1}}$ is an ascribed value v, then, $\varepsilon_1 t^{\widetilde{S_1}} \longrightarrow_c et'$ for some $et' \in \text{EvTerm} \cup \{\textbf{error}\}$. Hence $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{Config}_{\widetilde{S}} \cup \{\textbf{error}\}$ by Prop 40 and either (Rprotg), or (Rprotgerr).
3. $t^{\widetilde{S''_1}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r_1$ for some $r_1 \in \text{Config}_{\widetilde{S''_1}} \cup \{\textbf{error}\}$. Hence $t^{\widetilde{S}} \mid \mu \overset{e\ell}{\longmapsto}_{\tilde{\ell}_c} r$ for some $r \in \text{Config}_{\widetilde{S}} \cup \{\textbf{error}\}$ by Prop 40 and either (Rf), or (R$f$err). Also by Prop 40, if $r = t'^{\widetilde{S}} \mid \mu' \in \text{Term}_{\widetilde{S}}$ then also $t'^{\widetilde{S}} \mid \mu'$ and $dom(\mu) \subseteq dom(\mu')$.

□

C.7.2 Dynamic Gradual Guarantee

In this section we present the proof the Dynamic Gradual Guarantee for GSL$_{\mathsf{Ref}}$.

Definition 47 (Intrinsic term precision)**.** *Let* $\Omega \in \mathcal{P}(\text{Var}_* \times \text{Var}_*) \cup \mathcal{P}(\text{Loc}_* \times \text{Loc}_*)$ *be defined as* $\Omega ::= \{ x^{\widetilde{S}_{i1}} \sqsubseteq x^{\widetilde{S}_{i2}}, l^{\widetilde{S}_{i1}} \sqsubseteq l^{\widetilde{S}_{i2}} \}$ *We define an ordering relation* $(\cdot \vdash \cdot \sqsubseteq \cdot) \in (\mathcal{P}(\text{Var}_* \times \text{Var}_*) \cup \mathcal{P}(\text{Loc}_* \times \text{Loc}_*)) \times \text{Term}_* \times \text{Term}_*$ *shown in Figure 13.*

Definition 48 (Well Formedness of Ω)**.** *We say that* Ω *is well formed iff* $\forall \{ l^{\widetilde{S}_{i1}} \sqsubseteq l^{\widetilde{S}_{i2}} \} \in \Omega. \widetilde{S}_{i1} \sqsubseteq \widetilde{S}_{i2}$

Before proving the gradual guarantee, we first establish some auxiliary properties of precision. For the following propositions, we assume Well Formedness of Ω (Prop 48).

Proposition 41. *If* $\Omega \vdash t^{\widetilde{S}_1} \sqsubseteq t^{\widetilde{S}_2}$ *for some* $\Omega \in \mathcal{P}(\text{Var}_* \times \text{Var}_*) \cup \mathcal{P}(\text{Loc}_* \times \text{Loc}_*)$*, then* $\widetilde{S}_1 \sqsubseteq \widetilde{S}_2$.

$$\frac{}{\Omega \cup \{\, x^{\widetilde{S}_1} \sqsubseteq x^{\widetilde{S}_2} \,\} \vdash x^{\widetilde{S}_1} \sqsubseteq x^{\widetilde{S}_2}} \qquad \frac{\tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2}{\Omega \vdash b_{\tilde{\ell}_1} \sqsubseteq b_{\tilde{\ell}_2}} \qquad \frac{\tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2}{\Omega \vdash \mathsf{unit}_{\tilde{\ell}_1} \sqsubseteq \mathsf{unit}_{\tilde{\ell}_2}} \qquad \frac{\tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2}{\Omega \cup \{\, l^{\widetilde{S}_1} \sqsubseteq l^{\widetilde{S}_2} \,\} \vdash l^{\widetilde{S}_1}_{\tilde{\ell}_1} \sqsubseteq l^{\widetilde{S}_2}_{\tilde{\ell}_2}}$$

$$\frac{\widetilde{S}_{11} \sqsubseteq \widetilde{S}_{12} \quad \tilde{\ell}_{c1}{}' \sqsubseteq \tilde{\ell}_{c2}{}' \quad \tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2 \quad \Omega \cup \{\, x^{\widetilde{T}_{11}} \sqsubseteq x^{\widetilde{T}_{12}} \,\} \vdash t^{\widetilde{T}_{12}} \sqsubseteq t^{\widetilde{T}_{22}}}{\Omega \vdash (\lambda^{\tilde{\ell}_{c1}{}'} x^{\widetilde{S}_{11}}.t^{\widetilde{S}_{12}})_{\tilde{\ell}_1} \sqsubseteq (\lambda^{\tilde{\ell}_{c2}{}'} x^{\widetilde{S}_{21}}.t^{\widetilde{S}_{22}})_{\tilde{\ell}_2}}$$

$$\frac{\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2} \quad \tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2 \quad \tilde{\ell}'_1 \sqsubseteq \tilde{\ell}'_1 \quad \varepsilon_{r1} \sqsubseteq \varepsilon_{r2} \quad \varepsilon_1 \sqsubseteq \varepsilon_2 \quad \widetilde{S}_1 \sqsubseteq \widetilde{S}_2 \quad \Omega \vdash t^{\widetilde{S}'_1} \sqsubseteq t^{\widetilde{S}'_2} \quad \varepsilon_{\ell 1} \sqsubseteq \varepsilon_{\ell 2}}{\Omega \vdash \mathsf{prot}^{\tilde{\ell}_{c1},\tilde{\ell}_1,\widetilde{S}_1}_{\varepsilon_{r1},\varepsilon_{\ell 1}\tilde{\ell}'_1} \varepsilon_1 t^{\widetilde{S}'_1} \sqsubseteq \mathsf{prot}^{\tilde{\ell}_{c2},\tilde{\ell}_2,\widetilde{S}_2}_{\varepsilon'_{r2},\varepsilon_{\ell 2}\tilde{\ell}'_2} \varepsilon_2 t^{\widetilde{S}'_2}}$$

$$\frac{\Omega \vdash t^{\widetilde{S}_{11}} \sqsubseteq t^{\widetilde{S}_{21}} \quad \widetilde{S}_{12} \sqsubseteq \widetilde{S}_{22} \quad \varepsilon_1 \sqsubseteq \varepsilon_2}{(\varepsilon_1 t^{\widetilde{S}_{11}} :: \widetilde{S}_{12}) \sqsubseteq (\varepsilon_2 t^{\widetilde{S}_{21}} :: \widetilde{S}_{22})}$$

$$\frac{\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2} \quad \Omega \vdash t^{\widetilde{S}_{11}} \sqsubseteq t^{\widetilde{S}_{21}} \quad \Omega \vdash t^{\widetilde{S}_{12}} \sqsubseteq t^{\widetilde{S}_{22}} \quad \varepsilon_{11} \sqsubseteq \varepsilon_{21} \quad \varepsilon_{12} \sqsubseteq \varepsilon_{22} \quad \varepsilon_{\ell 1} \sqsubseteq \varepsilon_{\ell 2} \quad \widetilde{S}_1 \sqsubseteq \widetilde{S}_3 \quad \widetilde{S}_2 \sqsubseteq \widetilde{S}_4 \quad \tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2}{\Omega \vdash \varepsilon_{11} t^{\widetilde{S}_{11}} \; @^{\widetilde{S}_1 \xrightarrow{\tilde{\ell}_{c1}} \tilde{\ell}_1 \widetilde{S}_2}_{\varepsilon_{\ell 1}} \; {}_{12}\, t^{\widetilde{S}_{12}} \sqsubseteq \varepsilon_{21} t^{\widetilde{S}_{21}} \; @^{\widetilde{S}_3 \xrightarrow{\tilde{\ell}_{c2}} \tilde{\ell}_2 \widetilde{S}_4}_{\varepsilon_{\ell 2}} \; {}_{22}\, t^{\widetilde{S}_{22}}}$$

$$\frac{\tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2 \quad \Omega \vdash t^{\widetilde{S}_{11}} \sqsubseteq t^{\widetilde{S}_{21}} \quad \varepsilon_{11} \sqsubseteq \varepsilon_{21} \quad \Omega \vdash t^{\widetilde{S}_{12}} \sqsubseteq t^{\widetilde{S}_{23}} \quad \varepsilon_{12} \sqsubseteq \varepsilon_{22} \quad \Omega \vdash t^{\widetilde{S}_{13}} \sqsubseteq t^{\widetilde{S}_{23}} \quad \varepsilon_{13} \sqsubseteq \varepsilon_{23}}{\Omega \vdash \mathsf{if}\ ^{\tilde{\ell}_1}\varepsilon_{11} t^{\widetilde{S}_{11}}\ \mathsf{then}\ \varepsilon_{12} t^{\widetilde{S}_{12}}\ \mathsf{else}\ \varepsilon_{13} t^{\widetilde{S}_{13}} \sqsubseteq \mathsf{if}\ ^{\tilde{\ell}_2}\varepsilon_{21} t^{\widetilde{S}_{21}}\ \mathsf{then}\ \varepsilon_{22} t^{\widetilde{S}_{22}}\ \mathsf{else}\ \varepsilon_{23} t^{\widetilde{S}_{23}}}$$

$$\frac{\Omega \vdash t^{\widetilde{S}_{11}} \sqsubseteq t^{\widetilde{S}_{21}} \quad \Omega \vdash t^{\widetilde{S}_{12}} \sqsubseteq t^{\widetilde{S}_{22}} \quad \varepsilon_{11} \sqsubseteq \varepsilon_{21} \quad \varepsilon_{12} \sqsubseteq \varepsilon_{22} \quad \tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2}{\Omega \vdash (\varepsilon_{11} t^{\widetilde{S}_{11}} \oplus^{\tilde{\ell}_1} \varepsilon_{12} t^{\widetilde{S}_{12}}) \sqsubseteq (\varepsilon_{21} t^{\widetilde{S}_{21}} \oplus^{\tilde{\ell}_2} \varepsilon_{22} t^{\widetilde{S}_{22}})}$$

$$\frac{\widetilde{S}_1 \sqsubseteq \widetilde{S}_2 \quad \varepsilon_{\ell 1} \sqsubseteq \varepsilon_{\ell 2} \quad \tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2} \quad \varepsilon_1 \sqsubseteq \varepsilon_2 \quad \Omega \vdash t^{\widetilde{S}'_1} \sqsubseteq t^{\widetilde{S}'_2}}{\Omega \vdash \mathsf{ref}^{\widetilde{S}_1}_{\varepsilon_{\ell 1}} \varepsilon_1 t^{\widetilde{S}'_1} \sqsubseteq \mathsf{ref}^{\widetilde{S}_2}_{\varepsilon_{\ell 2}} \varepsilon_2 t^{\widetilde{S}'_2}}$$

$$\frac{\Omega \vdash t^{\widetilde{S}_{11}} \sqsubseteq t^{\widetilde{S}_{21}} \quad \widetilde{S}_1 \sqsubseteq \widetilde{S}_2 \quad \varepsilon_1 \sqsubseteq \varepsilon_2}{\Omega \vdash !^{\widetilde{S}_1}\varepsilon_1 t^{\widetilde{S}_{11}} \sqsubseteq\ !^{\widetilde{S}_2}\varepsilon_2 t^{\widetilde{S}_{21}}}$$

$$\frac{\Omega \vdash t^{\widetilde{S}_{11}} \sqsubseteq t^{\widetilde{S}_{21}} \quad \Omega \vdash t^{\widetilde{S}_{12}} \sqsubseteq t^{\widetilde{S}_{22}} \quad \varepsilon_{11} \sqsubseteq \varepsilon_{21} \quad \varepsilon_{12} \sqsubseteq \varepsilon_{22} \quad \varepsilon_1 \sqsubseteq \varepsilon_2}{\Omega \vdash \varepsilon_{11} t^{\widetilde{S}_{11}} :^{\varepsilon_1}= \varepsilon_{12} t^{\widetilde{S}_{12}} \sqsubseteq\ \varepsilon_{21} t^{\widetilde{S}_{21}} :^{\varepsilon_2}= \varepsilon_{22} t^{\widetilde{S}_{22}}}$$

$$\frac{\forall l^{\widetilde{S}_1} \in dom(\mu_1).\exists l^{\widetilde{S}_2} \in dom(\mu_2)\ s.t. \quad \Omega \vdash l^{\widetilde{S}_1} \sqsubseteq l^{\widetilde{S}_2} \quad \Omega \vdash \mu_1(l^{\widetilde{S}_1}) \sqsubseteq \mu_2(l^{\widetilde{S}_2})}{\Omega \vdash \mu_1 \sqsubseteq \mu_2}$$

Figure 13. Intrinsic term precision

Proof. Straightforward induction on $\Omega \vdash t^{\widetilde{S}_1} \sqsubseteq t^{\widetilde{S}_2}$, since the corresponding precision on types is systematically a premise (either directly or transitively). □

Proposition 42. *Let* $g_1, g_2 \in$ EvFrame *such that* $e\ell_1 \overset{\tilde{\ell}_{c1}}{\triangleright} g[\varepsilon_1 t^{\widetilde{S}_1}] \in \textsc{Term}_{\widetilde{S}'_1}$, $e\ell_2 \overset{\tilde{\ell}_{c2}}{\triangleright} g[\varepsilon_2 t^{\widetilde{S}_2}] \in \textsc{Term}_{\widetilde{S}'_2}$, *with* $\widetilde{S}'_1 \sqsubseteq \widetilde{S}'_2$. *Then if* $g_1[\varepsilon_{11} t_1^{\widetilde{S}_1}] \sqsubseteq g_2[\varepsilon_{21} t_1^{\widetilde{S}_2}]$, $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$ *and* $t_2^{\widetilde{S}_1} \sqsubseteq t_2^{\widetilde{S}_2}$, *then* $g_1[\varepsilon_{12} t_1^{\widetilde{S}_1}] \sqsubseteq g_2[\varepsilon_{22} t_1^{\widetilde{S}_2}]$

Proof. We proceed by case analysis on g_i.

Case $(\square\ @^{\widetilde{S}}_{\varepsilon}\ et)$. Then for $i \in \{1, 2\}$ g_i must have the form $\square @^{\widetilde{S}_i}_{\varepsilon'_i} \varepsilon'_i t^{\widetilde{S}'_i}$ for some $\varepsilon'_i, \widetilde{S}_i, \varepsilon'_i$ and $t^{\widetilde{S}'_i}$. As $g_1[\varepsilon_{11} t_1^{\widetilde{S}_1}] \sqsubseteq g_2[\varepsilon_{21} t_1^{\widetilde{S}_2}]$ then by $\sqsubseteq_{APP}$ $\varepsilon_1 \sqsubseteq \varepsilon_2, \varepsilon'_1 \sqsubseteq \varepsilon'_2$ and $t^{\widetilde{S}'_1} \sqsubseteq t^{\widetilde{S}'_2}$.

As $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$ and $t_2^{\widetilde{S}_1} \sqsubseteq t_2^{\widetilde{S}_2}$, then by $\sqsubseteq_{APP}$ $\varepsilon_{12} t_2^{\widetilde{S}_1}\ @^{\widetilde{S}_1}_{\varepsilon'_1} \varepsilon'_1 t^{\widetilde{S}'_1} \sqsubseteq \varepsilon_{22} t_2^{\widetilde{S}_2}\ @^{\widetilde{S}_2}_{\varepsilon'_2} \varepsilon'_2 t^{\widetilde{S}'_2}$, and the result holds.

Case $(\square \oplus^{\tilde{\ell}} et, ev \oplus^{\tilde{\ell}} \square, ev\ @^{\widetilde{S}}_{\varepsilon_\ell} \square, \square :: \widetilde{S}, !^{\widetilde{S}}\square, \square :^{\varepsilon_\ell}= et$, $ev :^{\varepsilon_\ell}= \square, \mathsf{if}\ ^{\tilde{\ell}}\square\ \mathsf{then}\ et\ \mathsf{else}\ et)$. Straightforward using similar argument to the previous case.

□

Proposition 43. *Let* $g_1, g_2 \in$ EvFrame *such that* $e\ell_1 \overset{\tilde{\ell}_{c1}}{\triangleright} g_1[\varepsilon_1 t^{\widetilde{S}_1}] \in \textsc{Term}_{\widetilde{S}'_1}$, $e\ell_2 \overset{\tilde{\ell}_{c2}}{\triangleright} g_2[\varepsilon_2 t^{\widetilde{S}_2}] \in \textsc{Term}_{\widetilde{S}'_2}$, *with* $\widetilde{S}'_1 \sqsubseteq \widetilde{S}'_2$. *Then if* $g_1[\varepsilon_1 t^{\widetilde{S}_1}] \sqsubseteq g_2[\varepsilon_2 t^{\widetilde{S}_2}]$ *then* $t^{\widetilde{S}_1} \sqsubseteq t^{\widetilde{S}_2}$ *and* $\varepsilon_1 \sqsubseteq \varepsilon_2$.

Proof. We proceed by case analysis on g_i.

Case $(\square\ @^{\widetilde{S}}_{\varepsilon}\ et)$. Then there must exist some $\varepsilon_{\ell i}, \widetilde{S}_i, \varepsilon'_i$ and $t^{\widetilde{S}'_i}$ such that $g[\varepsilon_1 t^{\widetilde{S}_1}] = \varepsilon_1 t^{\widetilde{S}_1}\ @^{\widetilde{S}_1}_{\varepsilon_1} \varepsilon_1' t^{\widetilde{S}'_1}$ and $g[\varepsilon_2 t^{\widetilde{S}_2}] = \varepsilon_2 t^{\widetilde{S}_2}\ @^{\widetilde{S}}_{\varepsilon_{\ell 2}} \varepsilon'_2 t^{\widetilde{S}'_2}$. Then by the hypothesis and the premises of $(\sqsubseteq_{APP})$, $t^{\widetilde{S}_1} \sqsubseteq t^{\widetilde{S}_2}$ and $\varepsilon_1 \sqsubseteq \varepsilon_2$, and the result holds immediately.

Case $(\square \oplus^{\tilde{\ell}} et, ev \oplus^{\tilde{\ell}} \square, ev\ @^{\widetilde{S}}_{\varepsilon_\ell} \square, \square :: \widetilde{S}, !^{\widetilde{S}}\square, \square :^{\varepsilon_\ell}= et$, $ev :^{\varepsilon_\ell}= \square, \mathsf{if}\ ^{\tilde{\ell}}\square\ \mathsf{then}\ et\ \mathsf{else}\ et)$. Straightforward using similar argument to the previous case.

□

Proposition 44. *Let* $f_1, f_2 \in \text{EvFrame}$ *such that* $e\ell_1 \overset{\tilde{\ell}_{c1}}{\triangleright} f_1[t^{\widetilde{S}_1}] \in \text{Term}_{\widetilde{S}_1'}$, $e\ell_2 \overset{\tilde{\ell}_{c2}}{\triangleright} f_2[t^{\widetilde{S}_2}] \in \text{Term}_{\widetilde{S}_2'}$, *with* $\widetilde{S}_1' \sqsubseteq \widetilde{S}_2'$. *Then if* $f_1[t_1^{\widetilde{S}_1}] \sqsubseteq f_2[t_1^{\widetilde{S}_2}]$ *and* $t_2^{\widetilde{S}_1} \sqsubseteq t_2^{\widetilde{S}_2}$, *then* $f_1[t_1^{\widetilde{S}_1}] \sqsubseteq f_2[t_1^{\widetilde{S}_2}]$

Proof. Suppose $f_i[t^{\widetilde{S}_1}] = g_i[\varepsilon_i t^{\widetilde{S}_i}]$. We know that $e\ell_1 \overset{\tilde{\ell}_{c1}}{\triangleright} g_1[\varepsilon_1 t_1^{\widetilde{S}_1}] \in \text{Term}_{\widetilde{S}_1'}$, $e\ell_1 \overset{\tilde{\ell}_{c1}}{\triangleright} g_2[\varepsilon_2 t_1^{\widetilde{S}_2}] \in \text{Term}_{\widetilde{S}_2'}$ and $\widetilde{S}_1' \sqsubseteq \widetilde{S}_2'$. Therefore if $g_1[\varepsilon_1 t_1^{\widetilde{S}_1}] \sqsubseteq g_1[\varepsilon_1 t_1^{\widetilde{S}_2}]$, by Prop 43, $\varepsilon_1 \sqsubseteq \varepsilon_2$. Finally by Prop 42 we conclude that $g_1[\varepsilon_1 t_2^{\widetilde{S}_1}] \sqsubseteq g_1[\varepsilon_1 t_2^{\widetilde{S}_2}]$. □

Proposition 45. *Let* $f_1, f_2 \in \text{EvFrame}$ *such that* $e\ell_1 \overset{\tilde{\ell}_{c1}}{\triangleright} f_1[t^{\widetilde{S}_1}] \in \text{Term}_{\widetilde{S}_1'}$, $e\ell_2 \overset{\tilde{\ell}_{c2}}{\triangleright} f_2[t^{\widetilde{S}_2}] \in \text{Term}_{\widetilde{S}_2'}$, *with* $\widetilde{S}_1' \sqsubseteq \widetilde{S}_2'$. *Then if* $f_1[t^{\widetilde{S}_1}] \sqsubseteq f_2[t^{\widetilde{S}_2}]$ *then* $t^{\widetilde{S}_1} \sqsubseteq t^{\widetilde{S}_2}$.

Proof. Suppose $f_i[t^{\widetilde{S}_1}] = g_i[\varepsilon_i t_1^{\widetilde{S}_i}]$. We know that $e\ell_1 \overset{\tilde{\ell}_{c1}}{\triangleright} g_1[\varepsilon_1 t_1^{\widetilde{S}_1}] \in \text{Term}_{\widetilde{S}_1'}$, $e\ell_1 \overset{\tilde{\ell}_{c1}}{\triangleright} g_2[\varepsilon_2 t_1^{\widetilde{S}_2}] \in \text{Term}_{\widetilde{S}_2'}$ and $\widetilde{S}_1' \sqsubseteq \widetilde{S}_2'$. Therefore if $g_1[\varepsilon_1 t_1^{\widetilde{S}_1}] \sqsubseteq g[\varepsilon_2 t_1^{\widetilde{S}_2}]$, then using Prop 43 we conclude that $t^{\widetilde{S}_1} \sqsubseteq t^{\widetilde{S}_2}$.

□

Proposition 46 (Substitution preserves precision)**.** *If* $\Omega \cup \{x^{\widetilde{S}_3} \sqsubseteq x^{\widetilde{S}_4}\} \vdash t^{\widetilde{S}_1} \sqsubseteq t^{\widetilde{S}_2}$ *and* $\Omega \vdash t^{\widetilde{S}_3} \sqsubseteq t^{\widetilde{S}_4}$, *then* $\Omega \vdash [t^{\widetilde{S}_3}/x^{\widetilde{S}_3}]t^{\widetilde{S}_1} \sqsubseteq [t^{\widetilde{S}_4}/x^{\widetilde{S}_4}]t^{\widetilde{S}_2}$.

Proof. By induction on the derivation of $t^{\widetilde{S}_1} \sqsubseteq t^{\widetilde{S}_2}$, and case analysis of the last rule used in the derivation. All cases follow either trivially (no premises) or by the induction hypotheses. □

Proposition 47 (Monotone precision for $\circ^{<:}$)**.** *If* $\varepsilon_1 \sqsubseteq \varepsilon_2$ *and* $\varepsilon_3 \sqsubseteq \varepsilon_4$ *then* $\varepsilon_1 \circ^{<:} \varepsilon_3 \sqsubseteq \varepsilon_2 \circ^{<:} \varepsilon_4$.

Proof. By definition of consistent transitivity for $<:$ and the definition of precision. □

Proposition 48 (Monotone precision for $\circ^{\preccurlyeq}$)**.** *If* $\varepsilon_1 \sqsubseteq \varepsilon_2$ *and* $\varepsilon_3 \sqsubseteq \varepsilon_4$ *then* $\varepsilon_1 \circ^{\preccurlyeq} \varepsilon_3 \sqsubseteq \varepsilon_2 \circ^{\preccurlyeq} \varepsilon_4$.

Proof. By definition of consistent transitivity for $\preccurlyeq$ and the definition of precision. □

Proposition 49 (Monotone precision for join)**.** *If* $\varepsilon_1 \sqsubseteq \varepsilon_2$ *and* $\varepsilon_3 \sqsubseteq \varepsilon_4$ *then* $\varepsilon_1 \mathbin{\widetilde{\Upsilon}} \varepsilon_3 \sqsubseteq \varepsilon_2 \mathbin{\widetilde{\Upsilon}} \varepsilon_4$.

Proof. By definition of join and the definition of precision. □

Proposition 50. *If* $\mathsf{Ref}\ \widetilde{S}_1 \sqsubseteq \mathsf{Ref}\ \widetilde{S}_2$ *then* $\widetilde{S}_1 \sqsubseteq \widetilde{S}_2$.

Proof. By definition of precision we know that $\{\mathsf{Ref}\ T \mid T \in \gamma(\widetilde{S}_1)\} \subseteq \{\mathsf{Ref}\ T \mid T \in \gamma(\widetilde{S}_2)\}$. This relation is true only if $\gamma(\widetilde{S}_1) \subseteq \gamma(\widetilde{S}_2)$ which is equivalent to $\widetilde{S}_1 \sqsubseteq \widetilde{S}_2$. □

Proposition 51 (Dynamic guarantee for $\longrightarrow$)**.** *Suppose* $\Omega \vdash t_1^{\widetilde{S}_1} \sqsubseteq t_1^{\widetilde{S}_2}$, $e\ell_1 \sqsubseteq e\ell_2$, $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$, *and* $\Omega \vdash \mu_1 \sqsubseteq \mu_2$. *If* $t_1^{\widetilde{S}_1} \mid \mu_1 \xrightarrow{e\ell_1}_{\tilde{\ell}_{c1}} t_2^{\widetilde{S}_1} \mid \mu_1'$ *then* $t_1^{\widetilde{S}_2} \mid \mu_2 \xrightarrow{e\ell_2}_{\tilde{\ell}_{c2}} t_2^{\widetilde{S}_2} \mid \mu_2'$ *where* $\Omega' \vdash t_2^{\widetilde{S}_1} \sqsubseteq t_2^{\widetilde{S}_2}$ *and* $\Omega' \vdash \mu_1' \sqsubseteq \mu_2'$, *for some* $\Omega' \supseteq \Omega$.

Proof. By induction on the structure of $t_1^{\widetilde{S}_1} \sqsubseteq t_1^{\widetilde{S}_2}$. For simplicity we omit the $\Omega \vdash$ notation on precision relations when it is not relevant for the argument.

Case ($\longrightarrow \oplus$). We know that $t_1^{\widetilde{S}_1} = (\varepsilon_{11}(b_1)_{\tilde{\ell}_{11}} \oplus^{\tilde{\ell}_1} \varepsilon_{12}(b_2)_{\tilde{\ell}_{12}})$ then by $(\sqsubseteq_\oplus)$ $t_1^{\widetilde{S}_2} = (\varepsilon_{21}(b_1)_{\tilde{\ell}_{21}} \oplus^{\tilde{\ell}_1} \varepsilon_{22}(b_2)_{\tilde{\ell}_{22}})$ for some $\varepsilon_{21}, \varepsilon_{22}$ such that $\varepsilon_{11} \sqsubseteq \varepsilon_{21}$ and $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$.
If $t_1^{\widetilde{S}_1} \mid \mu_1 \xrightarrow{e\ell_1}_{\tilde{\ell}_{c1}} b_3 \mid \mu_1$ where $b_3 = (\varepsilon_{11} \mathbin{\widetilde{\Upsilon}} \varepsilon_{12})(b_1 \ [\![\oplus]\!]\ b_2)_{(\tilde{\ell}_{11} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_{21})} :: \mathsf{Bool}_{\tilde{\ell}_1}$, then
$t_1^{\widetilde{S}_2} \mid \mu_2 \xrightarrow{e\ell_2}_{\tilde{\ell}_{c2}} b_3' \mid \mu_2$ where $b_3' = (\varepsilon_{21} \mathbin{\widetilde{\Upsilon}} \varepsilon_{22})(b_1 \ [\![\oplus]\!]\ b_2)_{(\tilde{\ell}_{21} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_{22})} :: \mathsf{Bool}_{\tilde{\ell}_2}$. By Lemma 49, $(\varepsilon_{11} \mathbin{\widetilde{\Upsilon}} \varepsilon_{12}) \sqsubseteq (\varepsilon_{21} \mathbin{\widetilde{\Upsilon}} \varepsilon_{22})$. Also $(\tilde{\ell}_{11} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_{21}) \sqsubseteq (\tilde{\ell}_{21} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_{22})$.

$$\dfrac{\dfrac{(\tilde{\ell}_{11} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_{21}) \sqsubseteq (\tilde{\ell}_{12} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_{22})}{\Omega \vdash (b_1 \ [\![\oplus]\!]\ b_2)_{(\tilde{\ell}_{11} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_{21})} \sqsubseteq (b_1 \ [\![\oplus]\!]\ b_2)_{(\tilde{\ell}_{21} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_{22})}} \quad \mathsf{Bool}_{\tilde{\ell}_1} \sqsubseteq \mathsf{Bool}_{\tilde{\ell}_2} \qquad (\varepsilon_{11} \mathbin{\widetilde{\Upsilon}} \varepsilon_{12}) \sqsubseteq (\varepsilon_{21} \mathbin{\widetilde{\Upsilon}} \varepsilon_{22})}{\begin{array}{c}(\varepsilon_{11} \mathbin{\widetilde{\Upsilon}} \varepsilon_{12})(b_1 \ [\![\oplus]\!]\ b_2)_{(\tilde{\ell}_{11} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_{21})} :: \mathsf{Bool}_{\tilde{\ell}_1} \sqsubseteq \\ (\varepsilon_{21} \mathbin{\widetilde{\Upsilon}} \varepsilon_{22})(b_1 \ [\![\oplus]\!]\ b_2)_{(\tilde{\ell}_{21} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_{22})} :: \mathsf{Bool}_{\tilde{\ell}_2}\end{array}}$$

Therefore $t_2^{\widetilde{S}_1} \sqsubseteq t_2^{\widetilde{S}_2}$. As $\Omega' = \Omega$, $\mu_1' = \mu_1$ and $\mu_2 = \mu_2'$ then $\Omega' \vdash \mu_1' \sqsubseteq \mu_2'$.

Case ($\longrightarrow$prot). We know that $t_1^{\widetilde{S}_1} = \mathsf{prot}^{\tilde{\ell}_{c1}', \tilde{\ell}_1, \widetilde{S}_1}_{\varepsilon_{r1}', \varepsilon_{\ell 1} \tilde{\ell}_1'} \varepsilon_1 u_1$, then by $(\sqsubseteq_{\mathsf{prot}})$ $t_1^{\widetilde{S}_2} = \mathsf{prot}^{\tilde{\ell}_{c2}', \tilde{\ell}_2, \widetilde{S}_2}_{\varepsilon_{r2}', \varepsilon_{\ell 2} \tilde{\ell}_2'} \varepsilon_2 u_2$, and therefore

$$\dfrac{\begin{array}{ccc}\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2} & & \tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2 \\ \tilde{\ell}_1' \sqsubseteq \tilde{\ell}_1' & \varepsilon_{r1} \sqsubseteq \varepsilon_{r2} & \varepsilon_1 \sqsubseteq \varepsilon_2 \\ \widetilde{S}_1 \sqsubseteq \widetilde{S}_2 & \Omega \vdash u_1 \sqsubseteq u_2 & \varepsilon_{\ell 1} \sqsubseteq \varepsilon_{\ell 2}\end{array}}{\Omega \vdash \mathsf{prot}^{\tilde{\ell}_{c1}, \tilde{\ell}_1, \widetilde{S}_1}_{\varepsilon_{r1}, \varepsilon_{\ell 1} \tilde{\ell}_1'} \varepsilon_1 u_1 \sqsubseteq \mathsf{prot}^{\tilde{\ell}_{c2}, \tilde{\ell}_2, \widetilde{S}_2}_{\varepsilon_{r2}', \varepsilon_{\ell 2} \tilde{\ell}_2'} \varepsilon_2 u_2}$$

for some $\varepsilon_2, u_2, \widetilde{S}_2$ and $\varepsilon_{\ell 2}$, where $u_1 \in \text{Term}_{\widetilde{S}_1'}$ and $u_2 \in \text{Term}_{\widetilde{S}_2'}$. If
$t_1^{\widetilde{S}_1} \mid \mu_1 \xrightarrow{\varepsilon_{r1} \tilde{\ell}_{r1}}_{\tilde{\ell}_{c1}} (\varepsilon_1 \mathbin{\widetilde{\Upsilon}} \varepsilon_{\ell 1})(u_1 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_1') :: \widetilde{S}_1 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_1 \mid \mu_1$. Therefore, $t_1^{\widetilde{S}_2} \mid \mu_2 \xrightarrow{\varepsilon_{r2} \tilde{\ell}_{r2}}_{\tilde{\ell}_{c2}} (\varepsilon_2 \mathbin{\widetilde{\Upsilon}} \varepsilon_{\ell 2})(u_2 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_2') :: \widetilde{S}_2 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_2 \mid \mu_2$. By Lemma 49, $(\varepsilon_1 \mathbin{\widetilde{\Upsilon}} \varepsilon_{\ell 1}) \sqsubseteq (\varepsilon_2 \mathbin{\widetilde{\Upsilon}} \varepsilon_{\ell 2})$, and as join is monotone $\widetilde{S}_1 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_1 \sqsubseteq \widetilde{S}_2 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_2$ and $(u_1 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_1') \sqsubseteq (u_2 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_2')$. Therefore by $\sqsubseteq_{::}$, $(\varepsilon_1 \mathbin{\widetilde{\Upsilon}} \varepsilon_{\ell 1})(u_1 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_1') :: \widetilde{S}_1 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_1 \sqsubseteq (\varepsilon_2 \mathbin{\widetilde{\Upsilon}} \varepsilon_{\ell 2})(u_2 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_2') :: \widetilde{S}_2 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_2$. As $\Omega' = \Omega$, $\mu_1' = \mu_1$ and $\mu_2 = \mu_2'$ then $\Omega' \vdash \mu_1' \sqsubseteq \mu_2'$.

Case ($\longrightarrow$app). We know that
$t_1^{\widetilde{S}_1} = \varepsilon_{11}(\lambda x^{\widetilde{S}_{11}}.t^{\widetilde{S}_{12}})_{\tilde{\ell}_1'} \ @^{\widetilde{S}_1 \xrightarrow{\tilde{\ell}_{c1}'} \tilde{\ell}_1 \widetilde{S}_2}_{\varepsilon_{\ell 1}} \ \varepsilon_{12} u$ then by $(\sqsubseteq_{app})$ $t_1^{\widetilde{S}_2}$ must have the form $t_1^{\widetilde{S}_2} = \varepsilon_{21}(\lambda x^{\widetilde{S}_{21}}.t^{\widetilde{S}_{22}})_{\tilde{\ell}_2'} \ @^{\widetilde{S}_3 \xrightarrow{\tilde{\ell}_{c2}'} \tilde{\ell}_2 \widetilde{S}_4}_{\varepsilon_{\ell 2}} \ \varepsilon_{22} u_2$ for some $\varepsilon_{21}, x^{\widetilde{S}_{21}}, t^{\widetilde{S}_{22}}, \widetilde{S}_3, \widetilde{S}_4, \varepsilon_{22}, \tilde{\ell}_{c2}', \tilde{\ell}_2$ and u_2.
Let us pose $\varepsilon_1 = \varepsilon_{12} \circ^{<:} idom(\varepsilon_{11})$ and $\varepsilon_{r1}' = (\varepsilon_{r1} \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon_{11})) \circ^{\preccurlyeq} \varepsilon_{\ell 1} \circ^{\preccurlyeq} ilat(\varepsilon_{11})$. Then
$t_1^{\widetilde{S}_1} \mid \mu_1 \xrightarrow{e\ell_1}_{\tilde{\ell}_{c1}} \mathsf{prot}^{\tilde{\ell}_{c1}', \tilde{\ell}_1, \widetilde{S}_2}_{\varepsilon_{r1}', ilbl(\varepsilon_{11}) \tilde{\ell}_1'} icod(\varepsilon_{11}) t_1' \mid \mu_1$ with $t_1' = [(\varepsilon_1 u_1 :: \widetilde{S}_{11})/x^{\widetilde{S}_{11}}] t^{\widetilde{S}_{12}}$.
Also, let us pose $\varepsilon_2 = \varepsilon_{22} \circ^{<:} idom(\varepsilon_{21})$ and $\varepsilon_{r1}' = (\varepsilon_{r1} \mathbin{\widetilde{\Upsilon}} ilbl(\varepsilon_{11})) \circ^{\preccurlyeq} \varepsilon_{\ell 1} \circ^{\preccurlyeq} ilat(\varepsilon_{11})$. Then
$t_1^{\widetilde{S}_2} \mid \mu_2 \xrightarrow{e\ell_2}_{\tilde{\ell}_{c2}} \mathsf{prot}^{\tilde{\ell}_{c2}', \tilde{\ell}_2, \widetilde{S}_4}_{\varepsilon_{r2}', ilbl(\varepsilon_{21}) \tilde{\ell}_2'} icod(\varepsilon_{21}) t_2' \mid \mu_2$ with $t_2' =$

$[(\varepsilon_2 u_2 :: \widetilde{S}_{21})/x^{\widetilde{S}_{21}}]t^{\widetilde{S}_{22}}$.

As $\Omega \vdash t_1^{\widetilde{S}_1} \sqsubseteq t_1^{\widetilde{S}_2}$, then $u_1 \sqsubseteq u_2$, $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$ and $idom(\varepsilon_{11}) \sqsubseteq idom(\varepsilon_{21})$ as well, then by Prop 47 $\varepsilon_1 \sqsubseteq \varepsilon_2$. Then $\varepsilon_1 u_1 :: \widetilde{S}_{11} \sqsubseteq \varepsilon_2 u_2 :: \widetilde{S}_{21}$ by $(\sqsubseteq_{::})$.

We also know by $(\sqsubseteq_{APP})$ and $(\sqsubseteq_\lambda)$ that $\Omega \cup \{x^{\widetilde{S}_{21}} \sqsubseteq x^{\widetilde{S}_{21}}\} \vdash t^{\widetilde{S}_{12}} \sqsubseteq t^{\widetilde{S}_{22}}$. By Substitution preserves precision (Prop 46) $t_1' \sqsubseteq t_2'$, therefore $icod(\varepsilon_{11})t_1' :: \widetilde{S}_2 \sqsubseteq icod(\varepsilon_{21})t_2' :: \widetilde{S}_4$ by $(\sqsubseteq_{::})$. Then as $\tilde{\ell}_{c1}' \sqsubseteq \tilde{\ell}_{c2}'$, $\tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2$, $ilbl(\varepsilon_{11}) \sqsubseteq ilbl_{21}$, $\tilde{\ell}_1' \sqsubseteq \tilde{\ell}_2'$ and by Lemma 47 and 49, $\varepsilon_{r1}' \sqsubseteq \varepsilon_{r2}'$. Then by $(\sqsubseteq_{\mathsf{prot}})$ $t_2^{\widetilde{S}_1} \sqsubseteq t_2^{\widetilde{S}_2}$. As $\Omega' = \Omega$, $\mu_1' = \mu_1$ and $\mu_2 = \mu_2'$ then $\Omega' \vdash \mu_1' \sqsubseteq \mu_2'$.

Case ($\longrightarrow$if-true). $t_1^{\widetilde{S}_1} = \mathsf{if}\ ^{\tilde{\ell}_1}\varepsilon_{11}\mathsf{true}_{\tilde{\ell}_1'}\ \mathsf{then}\ \ \mathsf{else}\ \varepsilon_{12}t^{\widetilde{S}_{12}}\varepsilon_{13}t^{\widetilde{S}_{13}}$ then by $(\sqsubseteq_{if})$ $t_1^{\widetilde{S}_2}$ has the form $t_1^{\widetilde{S}_2} = \mathsf{if}\ ^{\tilde{\ell}_2}\varepsilon_{21}\mathsf{true}_{\tilde{\ell}_2'}\ \mathsf{then}\ \ \mathsf{else}\ \varepsilon_{22}t^{\widetilde{S}_{22}}\varepsilon_{23}t^{\widetilde{S}_{23}}$ for some $\varepsilon_{21}, \varepsilon_{22}, t^{\widetilde{S}_{22}}, \varepsilon_{23}$, and $t^{\widetilde{S}_{32}}$. Then $t_1^{\widetilde{S}_1} \mid \mu_1 \xrightarrow{e\ell_1}_{\tilde{\ell}_{c1}} \mathsf{prot}^{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}_1, \tilde{\ell}_1, (\widetilde{S}_{12} \dot{\widetilde{\vee}} \widetilde{S}_{13})}_{(\varepsilon_{r1} \widetilde{\Upsilon} ilbl(\varepsilon_{12})), ilbl(\varepsilon_{11}) \tilde{\ell}_1'} \varepsilon_{12} t^{\widetilde{S}_{12}} \mid \mu_1$, and $t_1^{\widetilde{S}_2} \mid \mu 2 \xrightarrow{e\ell_2}_{\tilde{\ell}_{c2}} \mathsf{prot}^{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}_2, \tilde{\ell}_2, (\widetilde{S}_{12} \dot{\widetilde{\vee}} \widetilde{S}_{13})}_{(\varepsilon_{r2} \widetilde{\Upsilon} ilbl(\varepsilon_{22})), ilbl(\varepsilon_{21}) \tilde{\ell}_2'} \varepsilon_{22} t^{\widetilde{S}_{22}} \mid \mu_2$. Using the fact that $t_1^{\widetilde{S}_1} \sqsubseteq t_2^{\widetilde{S}_2}$ we know that $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$, $t^{\widetilde{S}_{12}} \sqsubseteq t^{\widetilde{S}_{22}}$, $\tilde{\ell}_1' \sqsubseteq \tilde{\ell}_2'$, as $\tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2$ and as join is monotone, $\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}_1 \sqsubseteq \tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}_2$. Also as $\varepsilon_{r1} \sqsubseteq \varepsilon_{r2}$ and $ilbl(\varepsilon_{12}) \sqsubseteq ilbl(\varepsilon_{22})$ then by Lemma 49 $(\varepsilon_{r1} \widetilde{\Upsilon} ilbl(\varepsilon_{12})) \sqsubseteq (\varepsilon_{r2} \widetilde{\Upsilon} ilbl(\varepsilon_{22}))$. Then using $(\sqsubseteq_{\mathsf{prot}})$, $t_1^{\widetilde{S}_2} \sqsubseteq t_2^{\widetilde{S}_2}$. As $\Omega' = \Omega$, $\mu_1' = \mu_1$ and $\mu_2 = \mu_2'$ then $\Omega' \vdash \mu_1' \sqsubseteq \mu_2'$.

Case ($\longrightarrow$if-false). Same as case $\longrightarrow$if-true, using the fact that $\varepsilon_{13} \sqsubseteq \varepsilon_{23}$ and $t^{\widetilde{S}_{13}} \sqsubseteq t^{\widetilde{S}_{23}}$.

Case ($\longrightarrow$ref). We know that $t_1^{\widetilde{S}_1} = \mathsf{ref}^{\widetilde{S}_1}_{\varepsilon_{\ell 1}}\ \varepsilon_1 u_1$, then by $(\sqsubseteq_{ref})$ $t_1^{\widetilde{S}_2} = \mathsf{ref}^{\widetilde{S}_2}_{\varepsilon_{\ell 2}}\ \varepsilon_2 u_2$, and therefore

$$\frac{\widetilde{S}_1 \sqsubseteq \widetilde{S}_2 \qquad \varepsilon_{\ell 1} \sqsubseteq \varepsilon_{\ell 2} \qquad \tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2} \qquad \varepsilon_1 \sqsubseteq \varepsilon_2 \qquad \Omega \vdash u_1 \sqsubseteq u_2}{\Omega \vdash \mathsf{ref}^{\widetilde{S}_1}_{\varepsilon_{\ell 1}}\ \varepsilon_1 u_1 \sqsubseteq \mathsf{ref}^{\widetilde{S}_2}_{\varepsilon_{\ell 2}}\ \varepsilon_2 u_2}$$

for some $\varepsilon_2, u_2, \widetilde{S}_2$ and $\varepsilon_{\ell 2}$, where $u_1 \in \text{TERM}_{\widetilde{S}_1'}$ and $u_2 \in \text{TERM}_{\widetilde{S}_2'}$. If $t_1^{\widetilde{S}_1} \mid \mu_1 \xrightarrow{e\ell_1}_{\tilde{\ell}_{c1}} l_\perp^{\widetilde{S}_1} \mid \mu_1[l^{\widetilde{S}_1} \mapsto v_1']$, for some $l^{\widetilde{S}_1} \notin \mu_1$ and where $v_1' = \varepsilon_1'(u_1 \widetilde{\Upsilon} \tilde{\ell}_{r1}) :: \widetilde{S}_1, \varepsilon_1' = \varepsilon_1 \widetilde{\Upsilon} (\varepsilon_{r1} \circ^{\preccurlyeq} \varepsilon_{\ell 1})$. Therefore, $t_1^{\widetilde{S}_2} \mid \mu_2 \xrightarrow{e\ell_2}_{\tilde{\ell}_{c2}} l_\perp^{\widetilde{S}_2} \mid \mu_2[l^{\widetilde{S}_2} \mapsto v_2']$, for some $l^{\widetilde{S}_2} \notin \mu_2$ and where $v_1' = \varepsilon_2'(u_2 \widetilde{\Upsilon} \tilde{\ell}_{r2}) :: \widetilde{S}_2, \varepsilon_2' = \varepsilon_2 \widetilde{\Upsilon} (\varepsilon_{r2} \circ^{\preccurlyeq} \varepsilon_{\ell 2})$. By Lemma 49 and 47, $\varepsilon_1' \sqsubseteq \varepsilon_2'$. Also as $\varepsilon_{r1} \sqsubseteq \varepsilon_{r2}$ and $\widetilde{S}_1 \sqsubseteq \widetilde{S}_2$, then by definition of rf, $\varepsilon_1'' \sqsubseteq \varepsilon_2''$. Then using $\Omega' = \Omega \cup \{ l^{\widetilde{S}_1} \sqsubseteq l^{\widetilde{S}_2} \}$ and that $\perp \sqsubseteq \perp$, by $(\sqsubseteq_l)$ we can see that $\Omega' \vdash l_\perp^{\widetilde{S}_1} \sqsubseteq l_\perp^{\widetilde{S}_2}$. As $\tilde{\ell}_{r1} \sqsubseteq \tilde{\ell}_{r2}$, by monotonicity of the join, $u_1 \widetilde{\Upsilon} \tilde{\ell}_{r1} \sqsubseteq u_2 \widetilde{\Upsilon} \tilde{\ell}_{r2}$. Therefore using $\sqsubseteq_{::}$, $\Omega' \vdash v_1' \sqsubseteq v_2'$. Also because $\Omega \subseteq \Omega'$, then by the fact that $\Omega \vdash \mu_1 \sqsubseteq \mu_2$, it is easy to see that $\Omega \cup \{ l^{\widetilde{S}_1} \sqsubseteq l^{\widetilde{S}_2} \} \vdash \mu_1[l^{\widetilde{S}_1} \mapsto v_1'] \sqsubseteq \mu_2[l^{\widetilde{S}_2} \mapsto v_2']$, i.e. $\Omega' \vdash \mu_1' \sqsubseteq \mu_2'$.

Case ($\longrightarrow$deref). We know that $t_1^{\widetilde{S}_1} = {!}^{\mathsf{Ref}\,\tilde{\ell}_1\ \widetilde{S}_1'}\varepsilon_1 l_{\tilde{\ell}_1'}^{\widetilde{S}_1''}$, $t_1^{\widetilde{S}_2} = {!}^{\mathsf{Ref}\,\tilde{\ell}_2\ \widetilde{S}_2'}\varepsilon_2 l_{\tilde{\ell}_2'}^{\widetilde{S}_2''}$ and so $\Omega \vdash {!}^{\mathsf{Ref}\,\tilde{\ell}_1\ \widetilde{S}_1'}\varepsilon_1 l_{\tilde{\ell}_1'}^{\widetilde{S}_1''} \sqsubseteq {!}^{\mathsf{Ref}\,\tilde{\ell}_2\ \widetilde{S}_2'}\varepsilon_2 l_{\tilde{\ell}_2'}^{\widetilde{S}_2''}$. As $\Omega \vdash \mu_1 \sqsubseteq \mu_2$, using $(\sqsubseteq_\mu)$ then $\Omega \vdash \mu_1(l^{\widetilde{S}_1''}) \sqsubseteq \mu_2(l^{\widetilde{S}_2''})$. Then ${!}^{\mathsf{Ref}\,\tilde{\ell}_1\ \widetilde{S}_1'}\varepsilon_1 l_{\tilde{\ell}_1'}^{\widetilde{S}_1''} \mid \mu \xrightarrow[\tilde{\ell}_{c1}]{\varepsilon_{r1}\tilde{\ell}_{r1}} \mathsf{prot}^{\tilde{\ell}_{c1}\widetilde{\Upsilon}\tilde{\ell}_1, \tilde{\ell}_1, \widetilde{S}_1'}_{(\varepsilon_{r1}\widetilde{\Upsilon}\varepsilon_1'), \varepsilon_1'\tilde{\ell}_1'} iref(\varepsilon_1)\mu_1(l^{\widetilde{S}_1''})$ where $\varepsilon_1' = ilbl(\varepsilon_1)$. Therefore ${!}^{\mathsf{Ref}\,\tilde{\ell}_2\ \widetilde{S}_2'}\varepsilon_2 l_{\tilde{\ell}_2'}^{\widetilde{S}_2''} \mid \mu \xrightarrow[\tilde{\ell}_{c2}]{\varepsilon_{r2}\tilde{\ell}_{r2}} \mathsf{prot}^{\tilde{\ell}_{c2}\widetilde{\Upsilon}\tilde{\ell}_2, \tilde{\ell}_2, \widetilde{S}_2'}_{(\varepsilon_{r2}\widetilde{\Upsilon}\varepsilon_2'), \varepsilon_2'\tilde{\ell}_2'} iref(\varepsilon_2)\mu_2(l^{\widetilde{S}_2''})$ where $\varepsilon_2' = ilbl(\varepsilon_2)$. By monotonicity of the join $\tilde{\ell}_{c1}\widetilde{\Upsilon}\tilde{\ell}_1 \sqsubseteq \tilde{\ell}_{c2}\widetilde{\Upsilon}\tilde{\ell}_2$ and $(\varepsilon_{r1} \widetilde{\Upsilon} \varepsilon_1') \sqsubseteq (\varepsilon_{r2} \widetilde{\Upsilon} \varepsilon_2')$.

Then using $(\sqsubseteq_{\mathsf{prot}})$ we can conclude that $\Omega \vdash t_2^{\widetilde{S}_1} \sqsubseteq t_2^{\widetilde{S}_2}$. As $\Omega' = \Omega$, $\mu_1 = \mu_1'$ and $\mu_2 = \mu_2'$ then also $\Omega' \vdash \mu_1' \sqsubseteq \mu_2'$.

Case ($\longrightarrow$assign).

$$\varepsilon_1 l_{\tilde{\ell}}^{\widetilde{S}} :=^{\varepsilon_\ell} \varepsilon_2 u \mid \mu \xrightarrow{\varepsilon_r \tilde{\ell}_r}_{\tilde{\ell}_c} \begin{cases} \mathsf{unit}_\perp \mid \mu[l^{\widetilde{S}} \mapsto \varepsilon'(u \widetilde{\Upsilon} (\tilde{\ell}_r \widetilde{\Upsilon} \tilde{\ell})) :: \widetilde{S}] \\ \mathbf{error} \qquad \text{if } \varepsilon' \text{ is not defined} \end{cases}$$

where $\varepsilon' = (iref(\varepsilon_1) \circ^{<:} \varepsilon_2) \widetilde{\Upsilon} ((\varepsilon_r \widetilde{\Upsilon} ilbl(\varepsilon_1)) \circ^{\preccurlyeq} \varepsilon_\ell \circ^{\preccurlyeq} ilbl(iref(\varepsilon_1))$

We know that $t_1^{\widetilde{S}_1} = \varepsilon_{11} l_{\tilde{\ell}_1}^{\widetilde{S}_{11}} :=^{\varepsilon_{\ell 1}} \varepsilon_{12} u_1$, $t_1^{\widetilde{S}_2} = \varepsilon_{21} l_{\tilde{\ell}_2}^{\widetilde{S}_{21}} :=^{\varepsilon_{\ell 2}} \varepsilon_{22} u_2$ and so $\Omega \vdash \varepsilon_{11} l_{\tilde{\ell}_1}^{\widetilde{S}_{11}} :=^{\varepsilon_{\ell 1}} \varepsilon_{12} u_1 \sqsubseteq \varepsilon_{21} l_{\tilde{\ell}_2}^{\widetilde{S}_{21}} :=^{\varepsilon_{\ell 2}} \varepsilon_{22} u_2$. Let $\varepsilon_1' = itref(\varepsilon_{11}) \circ^{=} \varepsilon_{12}$, then $t_1^{\widetilde{S}_1} \mid \mu_1 \xrightarrow[\tilde{\ell}_{c1}]{\varepsilon_{r1}\tilde{\ell}_{r1}} \mathsf{unit}_\perp \mid \mu_1[l^{\widetilde{S}_{11}} \mapsto v_1]$, where $v_1 = \varepsilon_1'(u_1 \widetilde{\Upsilon} (\tilde{\ell}_{r1} \widetilde{\Upsilon} \tilde{\ell}_1)) :: \widetilde{S}_{11}$, and $\varepsilon_1' = (iref(\varepsilon_{11}) \circ^{<:} \varepsilon_{12}) \widetilde{\Upsilon} ((\varepsilon_{r1} \widetilde{\Upsilon} ilbl(\varepsilon_{11})) \circ^{\preccurlyeq} \varepsilon_{\ell 1} \circ^{\preccurlyeq} ilbl(iref(\varepsilon_{11}))$. Similar, let $\varepsilon_2' = itref(\varepsilon_{21}) \circ^{=} \varepsilon_{22}$, then $t_1^{\widetilde{S}_2} \mid \mu_2 \xrightarrow[\tilde{\ell}_{c2}]{\varepsilon_{r2}\tilde{\ell}_{r2}} \mathsf{unit}_\perp \mid \mu_2[l^{\widetilde{S}_{21}} \mapsto v_2]$, where $v_2 = \varepsilon_2'(u_2 \widetilde{\Upsilon} (\tilde{\ell}_{r2} \widetilde{\Upsilon} \tilde{\ell}_2)) :: \widetilde{S}_{21}$, and $\varepsilon_2' = (iref(\varepsilon_{21}) \circ^{<:} \varepsilon_{22}) \widetilde{\Upsilon} ((\varepsilon_{r2} \widetilde{\Upsilon} ilbl(\varepsilon_{21})) \circ^{\preccurlyeq} \varepsilon_{\ell 2} \circ^{\preccurlyeq} ilbl(iref(\varepsilon_{21}))$. We need to prove that $\mu_1' = \mu_1[l^{\widetilde{S}_{11}} \mapsto v_1] \sqsubseteq \mu_2' = \mu_2[l^{\widetilde{S}_{21}} \mapsto v_2]$. Because $\Omega \vdash \mu_1 \sqsubseteq \mu_2$ then $\Omega \vdash l^{\widetilde{S}_{11}} \sqsubseteq l^{\widetilde{S}_{21}}$ by $(\sqsubseteq_\mu)$. By well formedness of Ω we also know that $\widetilde{S}_{11} \sqsubseteq \widetilde{S}_{21}$. Therefore, by Lemmas 47, 48 and 49 $\varepsilon_1' \sqsubseteq \varepsilon_2'$. Then using $\sqsubseteq_{::}$, $v_1 \sqsubseteq v_2$, following that $\Omega' = \Omega \vdash \mu_1' \sqsubseteq \mu_2'$.

□

Proposition 9 (Dynamic guarantee)**.** *Suppose* $t_1^{\widetilde{S}_1} \sqsubseteq t_1^{\widetilde{S}_2}$, $e\ell_1 \sqsubseteq e\ell_2$, $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$, *and* $\mu_1 \sqsubseteq \mu_2$. *If* $t_1^{\widetilde{S}_1} \mid \mu_1 \xrightarrow[\tilde{\ell}_{c1}]{e\ell_1} t_2^{\widetilde{S}_1} \mid \mu_1'$ *then* $t_1^{\widetilde{S}_2} \mid \mu_2 \xrightarrow[\tilde{\ell}_{c2}]{e\ell_2} t_2^{\widetilde{S}_2} \mid \mu_2'$ *where* $t_2^{\widetilde{S}_1} \sqsubseteq t_2^{\widetilde{S}_2}$ *and* $\mu_1' \sqsubseteq \mu_2'$.

Proof. We prove the following property instead: Suppose $\Omega \vdash t_1^{\widetilde{S}_1} \sqsubseteq t_1^{\widetilde{S}_2}$, $e\ell_1 \sqsubseteq e\ell_2$, $\tilde{\ell}_{c1} \sqsubseteq \tilde{\ell}_{c2}$, and $\Omega \vdash \mu_1 \sqsubseteq \mu_2$. If $t_1^{\widetilde{S}_1} \mid \mu_1 \xrightarrow[\tilde{\ell}_{c1}]{e\ell_1} t_2^{\widetilde{S}_1} \mid \mu_1'$ then $t_1^{\widetilde{S}_2} \mid \mu_2 \xrightarrow[\tilde{\ell}_{c2}]{e\ell_2} t_2^{\widetilde{S}_2} \mid \mu_2'$ where $\Omega' \vdash t_2^{\widetilde{S}_1} \sqsubseteq t_2^{\widetilde{S}_2}$ and $\Omega' \vdash \mu_1' \sqsubseteq \mu_2'$, for some $\Omega' \supseteq \Omega$.

By induction on the structure of a derivation of $t_1^{\widetilde{S}_1} \sqsubseteq t_1^{\widetilde{S}_2}$. For simplicity we omit the $\Omega \vdash$ notation on precision relations when it is not relevant for the argument.

Case (R$\longrightarrow$). $\Omega \vdash t_1^{\widetilde{S}_1} \sqsubseteq t_1^{\widetilde{S}_2}, \Omega \vdash \mu_1 \sqsubseteq \mu_2$ and $t_1^{\widetilde{S}_1} \mid \mu_1 \xrightarrow[\tilde{\ell}_{c1}]{e\ell_1} t_2^{\widetilde{S}_1} \mid \mu_1'$. By dynamic guarantee of $\longrightarrow$ (Prop 51), $t_1^{\widetilde{S}_2} \mid \mu_2 \xrightarrow[\tilde{\ell}_{c2}]{e\ell_2} t_1^{\widetilde{S}_2} \mid \mu_2'$ where $\Omega' \vdash t_2^{\widetilde{S}_1} \sqsubseteq t_2^{\widetilde{S}_2}, \Omega' \vdash \mu_1' \sqsubseteq \mu_2'$ for some $\Omega' \supseteq \Omega$. And the result holds immediately.

Case (Rf). $t_1^{\widetilde{S}_1} = f_1[t_1^{\widetilde{S}_1'}], t_1^{\widetilde{S}_2} = f_2[t_1^{\widetilde{S}_2'}]$. We know that $\Omega \vdash f_1[t_1^{\widetilde{S}_1'}] \sqsubseteq f_2[t_1^{\widetilde{S}_2'}]$. By using Prop 41, $\widetilde{S}_1' \sqsubseteq \widetilde{S}_2'$. By Prot 45, we also know that $\Omega \vdash t_1^{\widetilde{S}_1'} \sqsubseteq t_1^{\widetilde{S}_2'}$. By induction hypothesis, $t_1^{\widetilde{S}_1'} \mid \mu_1 \xrightarrow[\tilde{\ell}_{c1}]{e\ell_1} t_2^{\widetilde{S}_1'} \mid \mu_1', t_1^{\widetilde{S}_2'} \mid \mu_2 \xrightarrow[\tilde{\ell}_{c2}]{e\ell_2} t_2^{\widetilde{S}_2'} \mid \mu_2', \Omega' \vdash$

$t_2^{\widetilde{S}'_1} \sqsubseteq t_2^{\widetilde{S}'_2}$ and $\Omega' \vdash \mu'_1 \sqsubseteq \mu'_2$ for some $\Omega' \supseteq \Omega$.

Then by Prop 44 then $\Omega' \vdash f_1[t_2^{\widetilde{S}'_1}] \sqsubseteq f_2[t_2^{\widetilde{S}'_2}]$ and the result holds.

Case (Rprot). Then $t_1^{\widetilde{S}_1} = \mathsf{prot}^{\tilde{\ell}'_{c1},\tilde{\ell}_1,\widetilde{S}_1}_{\varepsilon'_{r1},\varepsilon_{\ell 1}\tilde{\ell}'_1} \varepsilon_1 t_1^{\widetilde{S}'_1}$ and $t_1^{\widetilde{S}_2} = \mathsf{prot}^{\tilde{\ell}'_{c2},\tilde{\ell}_2,\widetilde{S}_2}_{\varepsilon'_{r2},\varepsilon_{\ell 2}\tilde{\ell}'_2} \varepsilon_2 t_1^{\widetilde{S}'_2}$

As $t_1^{\widetilde{S}_1} \sqsubseteq t_1^{\widetilde{S}_2}$ then by $(\sqsubseteq_{\mathsf{prot}})$, $t_1^{\widetilde{S}'_1} \sqsubseteq t_1^{\widetilde{S}'_2}$, $\tilde{\ell}'_{c1} \sqsubseteq \tilde{\ell}'_{c2}, \varepsilon'_{r1} \sqsubseteq \varepsilon'_{r2}, \varepsilon_{\ell 1} \sqsubseteq \varepsilon_{\ell 2}, \tilde{\ell}_1 \sqsubseteq \tilde{\ell}_2, \tilde{\ell}'_1 \sqsubseteq \tilde{\ell}'_2$, and $\varepsilon_1 \sqsubseteq \varepsilon_2$. By (Rprot), $t_1^{\widetilde{S}'_i} \mid \mu \xrightarrow{\varepsilon'_{ri}(\tilde{\ell}_{ri} \curlyvee \tilde{\ell}'_i)}_{\tilde{\ell}'_{ci}} t_2^{\widetilde{S}'_i} \mid \mu'$ and by induction hypothesis, $t_2^{\widetilde{S}'_1} \sqsubseteq t_2^{\widetilde{S}'_2}$ and $\Omega' \vdash \mu'_1 \sqsubseteq \mu'_2$ for some $\Omega' \supseteq \Omega$.

But then by $(\sqsubseteq_{\mathsf{prot}})$, $\Omega' \vdash \mathsf{prot}^{\tilde{\ell}'_{c1},\tilde{\ell}_1,\widetilde{S}_1}_{\varepsilon'_{r1},\varepsilon_{\ell 1}\tilde{\ell}'_1} \varepsilon_1 t_2^{\widetilde{S}'_1} \sqsubseteq \mathsf{prot}^{\tilde{\ell}'_{c2},\tilde{\ell}_2,\widetilde{S}_2}_{\varepsilon'_{r2},\varepsilon_{\ell 2}\tilde{\ell}'_2} \varepsilon_2 t_2^{\widetilde{S}'_2}$ and the result holds.

Case (Rg). $t_1^{\widetilde{S}_1} = g_1[et_1]$, $t_1^{\widetilde{S}_2} = g_2[et_2]$, where $\Omega \vdash g_1[et_1] \sqsubseteq g_2[et_2]$. Also $et_1 \longrightarrow_c et'_1$ and $et_2 \longrightarrow_c et'_2$.
Then there exists $\widetilde{S}_1$, ε_{11}, ε_{12} and v_1 such that $et_1 = \varepsilon_{11}(\varepsilon_{12} v_1 :: \widetilde{S}_1)$. Also there exists $\widetilde{S}_2$, ε_{21}, ε_{22} and v_2 such that $et_2 = \varepsilon_{21}(\varepsilon_{22} v_2 :: \widetilde{S}_2)$. By Prop 43, $\varepsilon_{11} \sqsubseteq \varepsilon_{21}$, and by $(\sqsubseteq_{::})$ $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$, $v_1 \sqsubseteq v_2$ and $\widetilde{S}_1 \sqsubseteq \widetilde{S}_2$. Then as $et_1 \longrightarrow_c (\varepsilon_{12} \circ^{<:} \varepsilon_{11}) v_1$ and $et_2 \longrightarrow_c (\varepsilon_{22} \circ^{<:} \varepsilon_{21}) v_2$ then, by Prop 47 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 42, it follows that $\Omega \vdash g_1[et'_1] \sqsubseteq g_1[et'_2]$. As $\Omega' = \Omega$, $\mu'_1 = \mu_1$ and $\mu_2 = \mu'_2$ then $\Omega' \vdash \mu'_1 \sqsubseteq \mu'_2$.

Case (Rprotg). Analogous to (Rprot) case but using $\longrightarrow_c$ instead. □

D. Noninterference

In this section we present the proof of noninterference for $\mathrm{GSL}_{\mathsf{Ref}}$. Section D.1 present some auxiliary definitions and section D.2 present the proof of noninterference.

D.1 Definitions

We introduce a function $uval$, which strips away ascriptions from a simple value:

$$uval : \textsc{GType} \to \textsc{SimpleValue}$$
$$uval(u) = u$$
$$uval(\varepsilon u :: \widetilde{S}) = u.$$

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:

$$rval : \textsc{Value} \to \textsc{RawValue}$$
$$rval(b_{\tilde{\ell}}) = b$$
$$rval(\varepsilon b_{\tilde{\ell}} :: \widetilde{S}) = b$$
$$\vdots$$

Definition 49 (Gradual security logical relations)**.** *For an arbitrary element ℓ_o of the security lattice, the ℓ_o-level gradual security relations are step-indexed and type-indexed binary relations on pairs of closed terms and stores defined inductively as presented in Figure 14. The notation $\langle \varepsilon_r \tilde{\ell}_1 \overset{\tilde{\ell}_c}{\triangleright} v_1, \mu_1 \rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_2 \overset{\tilde{\ell}_c}{\triangleright} v_2, \mu_2 \rangle : \widetilde{S}$ indicates that the pair value v_1 and store μ_1 is related to the pair value v_2 and store μ_2 at type $\widetilde{S}$ for k steps when observed at the security level ℓ_o. Similarly, the notation $\langle \varepsilon_r \tilde{\ell}_1 \overset{\tilde{\ell}_c}{\triangleright} t_1, \mu_1 \rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_2 \overset{\tilde{\ell}_c}{\triangleright} t_2, \mu_2 \rangle : \mathcal{C}(\widetilde{S})$ indicates that the pair term t_1 and store μ_1, and the pair term t_2 and store μ_2 are related computations for k steps, that produce related values and related stores at type $\widetilde{S}$ when observed at the security level ℓ_o. Finally, notation $\langle \tilde{\ell}_1, \mu_1 \rangle \approx^k_{\ell_o} \langle \tilde{\ell}_1, \mu_1 \rangle$ relates label $\tilde{\ell}_1$ and store μ_1 with label $\tilde{\ell}_2$ and store μ_2 are related stores for k steps.*

We say that a value is *observable* at level ℓ_o if, given a PC $\tilde{\ell}_r$, the value is typeable, and the PC and the label of the value are sublabel of ℓ_o. Also, as value v can be a casted value, we need to analyze if its underlying evidence justify that the security level of the bare value is also subsumed by the observer security level. This is done by passing the value as argument of a function that expects a value with security level ℓ_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 ℓ_o, and therefore is *not* observable. For instance, consider value $v = \langle \mathsf{Bool}_{\mathsf{H}}, \mathsf{Bool}_?, \mathsf{Bool}_? \rangle^2_1 \, \mathsf{true}_? :: \mathsf{Bool}_?$. The level of the value and the bare value are sublabel of ℓ_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 $\mathsf{L} \preccurlyeq \mathsf{H}$), because the consistent transitivity operation $\langle \mathsf{Bool}_{\mathsf{L}}, \mathsf{Bool}_{\mathsf{L}} \rangle^1_1 \circ^{<:} \langle \mathsf{Bool}_{\mathsf{H}}, \mathsf{Bool}_?, \mathsf{Bool}_? \rangle^2_1$ 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. We say that store μ_2 is the evolution of store μ_1, annotated $\mu_1 \twoheadrightarrow \mu_2$ if the domain of μ_1 is a subset of μ_2.

Two pairs of values and stores are related for k steps at type $\mathsf{Bool}_{\tilde{\ell}}$ if the values can be typed as $\mathsf{Bool}_{\tilde{\ell}}$ using the labels as context, and if the labels and stores are related for k steps. Additionally, if the values are observable then the raw values are the same.

Similarly, two pairs are observables at type $\mathsf{Unit}_{\tilde{\ell}}$ if the values have type $\mathsf{Unit}_{\tilde{\ell}}$ and the stores are also related (the raw values are always the same).

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 at the return type.

Pairs are related at type $\mathsf{Ref}_{\tilde{\ell}} \, \widetilde{S}$ similarly to booleans: values have a type and the stores are related. If the values are observables then the locations must be the same. By definition of related stores if the locations are the same then the underlying values in the store are also related for $j < k$ steps.

Two pairs of terms and stores are related computations for k steps at type $\widetilde{S}$, first, if the terms can be typed as $\widetilde{S}$. Second if both labels and stores are related for k steps. Third, if for any $j < k$ both terms can be reduced for at least j steps and 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 $\widetilde{S}$. Notice that the logical relation also relates programs that do not terminate as long as after k steps the new stores are also related.

Definition 50 (Secure program)**.** *A well-typed program configuration $\langle \varepsilon_r \tilde{\ell} \overset{\tilde{\ell}_c}{\triangleright} t, \mu \rangle$ that produces a ℓ_o-observable output of type $\widetilde{S}$ (*i.e. *$label(\widetilde{S}) \preccurlyeq \ell_o$) is secure iff $\langle \varepsilon_r \tilde{\ell} \overset{\tilde{\ell}_c}{\triangleright} t, \mu \rangle \approx_{\ell_o} \langle \varepsilon_r \tilde{\ell} \overset{\tilde{\ell}_c}{\triangleright} t, \mu \rangle : \mathcal{C}(\widetilde{S})$.*

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

$$\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v_2, \mu_2\rangle : \mathsf{Bool}_{\tilde{\ell}} \iff \varepsilon_r\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i \in \textsc{Term}_{\mathsf{Bool}_{\tilde{\ell}}} \wedge \langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle \wedge \mathsf{obs}_{\ell_o}(\varepsilon_{ri}\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i) \implies rval(v_1) = rval(v_2)$$

$$\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v_2, \mu_2\rangle : \mathsf{Unit}_{\tilde{\ell}} \iff \varepsilon_r\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i \in \textsc{Term}_{\mathsf{Unit}_{\tilde{\ell}}} \wedge \langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle \wedge \mathsf{obs}_{\ell_o}(\varepsilon_{ri}\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i) \implies rval(v_1) = rval(v_2)$$

$$\begin{aligned}\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v_2, \mu_2\rangle : \widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_2 \iff\ & \\ \varepsilon_r\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i \in \textsc{Term}_{\widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_2} \wedge \langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle \wedge & \\ \mathsf{obs}_{\ell_o}(\varepsilon_{ri}\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i) \implies \forall j \le k. \forall \widetilde{S}' = \widetilde{S}''_1 \xrightarrow{\tilde{\ell}''_c}_{\tilde{\ell}''} \widetilde{S}''_2, \widetilde{S}'_1, & \\ \varepsilon_1 \vdash \widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_2 \lesssim \widetilde{S}', \text{ and } \varepsilon_2 \vdash \widetilde{S}'_1 \lesssim \widetilde{S}''_1, \varepsilon_\ell \vdash \tilde{\ell}_c \curlyvee \tilde{\ell} \preccurlyeq \tilde{\ell}'_c, \text{ we have:} & \\ \forall v'_i, \mu'_i, \langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v'_1, \mu'_1\rangle \approx^j_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v'_2, \mu_2\rangle : \widetilde{S}'_1, \mu_i \rightarrowtriangle \mu'_i, & \\ \langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} (\varepsilon_1 v_1 \mathbin{@}^{\widetilde{S}'}_{\varepsilon_\ell} \varepsilon_2 v'_1), \mu'_1\rangle \approx^j_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} (\varepsilon_1 v_2 \mathbin{@}^{\widetilde{S}'}_{\varepsilon_\ell} \varepsilon_2 v'_2), \mu'_2\rangle : \mathcal{C}(\widetilde{S}''_2 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}') &\end{aligned}$$

$$\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v_2, \mu_2\rangle : \mathsf{Ref}_{\tilde{\ell}}\, \widetilde{S} \iff \varepsilon_r\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i \in \textsc{Term}_{\mathsf{Ref}_{\tilde{\ell}}\, \widetilde{S}} \wedge \langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle \wedge \mathsf{obs}_{\ell_o}(\varepsilon_{ri}\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} v_i) \implies rval(v_1) = rval(v_2)$$

$$\begin{aligned}\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} t_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} t_2, \mu_2\rangle : \mathcal{C}(\widetilde{S}) \iff\ & \varepsilon_r\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} t_i \in \textsc{Term}_{\widetilde{S}} \wedge \langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle \wedge \\ & (t_i \mid \mu_i \overset{\varepsilon_{ri}\tilde{\ell}_{ri}}{\longmapsto}{}^{j}_{\tilde{\ell}_c} t'_i \mid \mu'_i \wedge \langle \tilde{\ell}_{r1}, \mu'_1\rangle \approx^{k-j}_{\ell_o} \langle \tilde{\ell}_{r2}, \mu'_2\rangle \wedge \mu_i \rightarrowtriangle \mu'_i \wedge irred(t'_i) \\ & \implies \langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} t'_1, \mu'_1\rangle \approx^{k-j}_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} t'_2, \mu'_2\rangle : \widetilde{S})\end{aligned}$$

$$\begin{aligned}\langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle \iff\ & \exists \tilde{\ell}_{ci}, \varepsilon_{ri} \vdash \tilde{\ell}_{ri} \mathbin{\widetilde{\preccurlyeq}} \tilde{\ell}_{ci}, \forall j < k, \forall l^{\widetilde{S}} \in dom(\mu_1) \cap dom(\mu_2). \\ & \langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} \mu_1(l^{\widetilde{S}}), \mu_1\rangle \approx^j_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} \mu_2(l^{\widetilde{S}}), \mu_2\rangle : \widetilde{S}\end{aligned}$$

$$\mu_1 \rightarrowtriangle \mu_2 \iff dom(\mu_1) \subseteq dom(\mu_2)$$

$$\mathsf{obs}_{\ell_o}(\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v) \iff \varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v \in \textsc{Term}_{\widetilde{S}} \wedge \tilde{\ell}_r \mathbin{\widetilde{\preccurlyeq}} \ell_o \wedge \left((v = \varepsilon u :: \widetilde{S}) \implies ilbl(\varepsilon) \circ^{\preccurlyeq} \varepsilon_2 \text{ is defined}\right)$$
where $\varepsilon_2 = \mathcal{I}_{\preccurlyeq}(label(\widetilde{S}), \ell_o)$

Figure 14. Gradual security logical relations

Definition 51. *Let σ be a substitution and Γ a type substitution. We say that substitution σ satisfy environment Γ, written $\sigma \models \Gamma$, if and only if $dom(\sigma) = \Gamma$.*

Definition 52 (Related substitutions)**.** *Tuples $\langle \varepsilon_{r1}\tilde{\ell}_{r1}, \tilde{\ell}_{c1}, \sigma_1, \mu_1\rangle$ and $\langle \varepsilon_{r2}\tilde{\ell}_{r2}, \tilde{\ell}_{c2}, \sigma_2, \mu_2\rangle$ are related on k steps, notation $\Gamma \vdash \langle \varepsilon_{r1}\tilde{\ell}_{r1}, \tilde{\ell}_{c1}, \sigma_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2}, \tilde{\ell}_{c2}, \sigma_2, \mu_2\rangle$, if $\sigma_i \models \Gamma$, $\langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle$ and $\forall x^{\widetilde{S}} \in \Gamma. \langle \varepsilon_{r1}\tilde{\ell}_1 \overset{\tilde{\ell}_{c1}}{\triangleright} \sigma_1(x^{\widetilde{S}}), \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_2 \overset{\tilde{\ell}_{c2}}{\triangleright} \sigma_2(x^{\widetilde{S}}), \mu_2\rangle : \widetilde{S}$.*

D.2 Proof of noninterference

Lemma 52 (Noninterference for booleans)**.** *Suppose $k > 0$, and*

- *$\tilde{\ell}_r \mathbin{\widetilde{\preccurlyeq}} \ell_o$, $\tilde{\ell}_c \mathbin{\widetilde{\preccurlyeq}} \ell_o$, $\varepsilon_r \vdash \tilde{\ell}_r \mathbin{\widetilde{\preccurlyeq}} \tilde{\ell}_c$*
- *an open term $\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}} \in \textsc{Term}_{\mathsf{Bool}_{\ell_o}}$ where $FV(t) = \{ x^{\widetilde{S}_1} \}$ with $label(\widetilde{S}_1) \mathbin{\widetilde{\not\preccurlyeq}} \ell_o$*
- *two compatible valid stores $t^{\widetilde{S}} \vdash \mu_i$, $\langle \tilde{\ell}_r, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_r, \mu_2\rangle$*

Then for any $j < k$, $v_1, v_2 \in \textsc{Term}_{\widetilde{S}_1}$, if both

- *$t^{\widetilde{S}}[v_1/x^{\widetilde{S}_1}] \mid \mu_1 \overset{\varepsilon_r\tilde{\ell}_r}{\longmapsto}{}^{j}_{\tilde{\ell}_c} v'_1 \mid \mu'_1$*
- *$t^{\widetilde{S}}[v_2/x^{\widetilde{S}_1}] \mid \mu_2 \overset{\varepsilon_r\tilde{\ell}_r}{\longmapsto}{}^{j}_{\tilde{\ell}_c} v'_2 \mid \mu'_2$*

we have that $rval(v'_1) = rval(v'_2)$ and $\langle \tilde{\ell}_r, \mu'_1\rangle \approx^{k-j}_{\ell_o} \langle \tilde{\ell}_r, \mu'_2\rangle$.

Proof. The result follows as a special case of Proposition 73 below. □

In this theorem, we treat $t^{\widetilde{S}}$ as a program that takes $x^{\widetilde{S}_1}$ as its input. Furthermore, the security level $\tilde{\ell}' = label(\widetilde{S}_1)$ of the input is not subsumed by the security level ℓ_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 ℓ_o.

Note that we compare 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).

The notion of stores being compatible at a given observation level is indexed by a number of reduction steps, k. We write $\langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle$. This relation essentially states that all the values at observable store locations are equal.

Finally, we use notation $t^S \mid \mu \xrightarrow{e\ell}{}^{k}_{\tilde{\ell}_c} t'^S \mid \mu'$ to describe that configuration $t^S \mid \mu$ reduces, in at most k steps, to configuration $t'^S \mid \mu'$.

Lemma 53. *Consider $\varepsilon_1 \vdash \tilde{\ell} \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}'$. If $\forall \varepsilon_2$ such that $\varepsilon_2 \vdash \tilde{\ell}' \mathrel{\widetilde{\preccurlyeq}} \ell_o$, $\varepsilon_1 \circ^{\preccurlyeq} \varepsilon_2 \vdash \tilde{\ell} \mathrel{\widetilde{\preccurlyeq}} \ell_o$ is not defined. Then if $\varepsilon_3 \vdash \tilde{\ell}' \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}''$, then $\forall \varepsilon_4$ such that $\varepsilon_4 \vdash \tilde{\ell}'' \mathrel{\widetilde{\preccurlyeq}} \ell_o$, then $(\varepsilon_1 \circ^{\preccurlyeq} \varepsilon_3) \circ^{\preccurlyeq} \varepsilon_4 \vdash \tilde{\ell} \mathrel{\widetilde{\preccurlyeq}} \ell_o$ is not defined*

Proof. Applying associativity: $(\varepsilon_1 \circ^{\preccurlyeq} \varepsilon_3) \circ^{\preccurlyeq} \varepsilon_4 = \varepsilon_1 \circ^{\preccurlyeq} (\varepsilon_3 \circ^{\preccurlyeq} \varepsilon_4)$, but $(\varepsilon_3 \circ^{\preccurlyeq} \varepsilon_4) \vdash \tilde{\ell}' \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_o$, and we know that $\varepsilon_1 \circ^{\preccurlyeq} \varepsilon_2$ is not defined $\forall \varepsilon_2$ such that $\varepsilon_2 \vdash \tilde{\ell}' \mathrel{\widetilde{\preccurlyeq}} \ell_o$. Therefore $(\varepsilon_1 \circ^{\preccurlyeq} \varepsilon_3) \circ^{\preccurlyeq} \varepsilon_4 \vdash \tilde{\ell} \mathrel{\widetilde{\preccurlyeq}} \ell_o$ is not defined and the result holds. □

Lemma 54. *Consider $\varepsilon_1 \vdash \tilde{\ell} \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}'$. If $\forall \varepsilon_2$ such that $\varepsilon_2 \vdash \tilde{\ell}' \mathrel{\widetilde{\preccurlyeq}} \ell_o$, $\varepsilon_1 \circ^{\preccurlyeq} \varepsilon_2 \vdash \tilde{\ell} \mathrel{\widetilde{\preccurlyeq}} \ell_o$ is not defined. Also $\varepsilon_0 \vdash \tilde{\ell}_1 \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_2$, if $\varepsilon_3 \vdash \widetilde{\tilde{\ell}_2 \curlyvee \tilde{\ell}'} \preccurlyeq \ell_o$, then $(\varepsilon_0 \mathrel{\widetilde{\curlyvee}} \varepsilon_1) \circ^{\preccurlyeq} \varepsilon_3 \vdash \widetilde{\tilde{\ell}_1 \curlyvee \tilde{\ell}} \preccurlyeq \ell_o$ is not defined*

Proof. Let us prove that if $(\varepsilon_0 \mathrel{\widetilde{\curlyvee}} \varepsilon_1) \circ^{\preccurlyeq} \varepsilon_3 \vdash \widetilde{\tilde{\ell}_1 \curlyvee \tilde{\ell}} \preccurlyeq \ell_o$ is defined, then $\varepsilon_1 \circ^{\preccurlyeq} \varepsilon_2$ is defined.

As join is monotone $\exists \varepsilon_0'$ such that $\varepsilon_0' \vdash \widetilde{\tilde{\ell}' \preccurlyeq \tilde{\ell}_2 \curlyvee \tilde{\ell}'}$.

Suppose $\varepsilon_1 = \langle [\ell_{11}, \ell_{12}], [\ell_{21}, \ell_{22}] \rangle$, $\varepsilon_0 = \langle [\ell_{31}, \ell_{32}], [\ell_{41}, \ell_{42}] \rangle$, $\varepsilon_0' = \langle [\ell_{51}, \ell_{52}], [\ell_{61}, \ell_{62}] \rangle$, and $\varepsilon_3 = \langle [\ell_{71}, \ell_{72}], [\ell_{81}, \ell_{82}] \rangle$.
As $\varepsilon_0 \mathrel{\widetilde{\curlyvee}} \varepsilon_1 = \langle [\ell_{11} \curlyvee \ell_{31}, \ell_{12} \curlyvee \ell_{32}], [\ell_{21} \curlyvee \ell_{41}, \ell_{22} \curlyvee \ell_{42}] \rangle$ is defined, then $\ell_{11} \curlyvee \ell_{31} \preccurlyeq \ell_{12} \curlyvee \ell_{32}$ and $\ell_{21} \curlyvee \ell_{41} \preccurlyeq \ell_{22} \curlyvee \ell_{42}$. Also as

$$(\varepsilon_0 \mathrel{\widetilde{\curlyvee}} \varepsilon_1) \circ^{\preccurlyeq} \varepsilon_3 = \langle [\ell_{11} \curlyvee \ell_{31}, (\ell_{12} \curlyvee \ell_{32}) \curlywedge ((\ell_{22} \curlyvee \ell_{42}) \curlywedge \ell_{72}) \curlywedge \ell_{82}], [\ell_{11} \curlyvee \ell_{31} \curlyvee \ell_{21} \curlyvee \ell_{41} \curlyvee \ell_{72} \curlyvee \ell_{81}, \ell_{82}] \rangle$$

is defined then $\ell_{21} \curlyvee \ell_{41} \curlyvee \ell_{71} \preccurlyeq (\ell_{22} \curlyvee \ell_{42}) \curlywedge \ell_{72}$, $\ell_{11} \curlyvee \ell_{31} \preccurlyeq (\ell_{22} \curlyvee \ell_{42}) \curlywedge \ell_{72}$, $\ell_{11} \curlyvee \ell_{31} \preccurlyeq \ell_{82}$, and $\ell_{21} \curlyvee \ell_{41} \curlyvee \ell_{71} \preccurlyeq \ell_{82}$.

If we choose ε_0' as the interior of the judgment, then we do not get new information, therefore $[\ell_{21}, \ell_{22}] \sqsubseteq [\ell_{51}, \ell_{52}]$, *i.e.* $\ell_{51} \preccurlyeq \ell_{21}$ and $\ell_{22} \preccurlyeq \ell_{52}$. Using the same argument $\ell_{61} \preccurlyeq \ell_{71}$ and $\ell_{72} \preccurlyeq \ell_{62}$.

Then

$$\begin{array}{ll} & \varepsilon_0' \circ^{\preccurlyeq} \varepsilon_3 \\ = & \triangle^{\preccurlyeq}([\ell_{51}, \ell_{52}], [\ell_{61}, \ell_{62}] \sqcap [\ell_{71}, \ell_{72}], [\ell_{81}, \ell_{82}]) \\ = & \triangle^{\preccurlyeq}([\ell_{51}, \ell_{52}], [\ell_{61} \curlyvee \ell_{71}, \ell_{62} \curlywedge \ell_{72}], [\ell_{81}, \ell_{82}]) \\ = & \triangle^{\preccurlyeq}([\ell_{51}, \ell_{52}], [\ell_{71}, \ell_{72}], [\ell_{81}, \ell_{82}]) \end{array}$$

which is defined if $\ell_{51} \preccurlyeq \ell_{72}$, $\ell_{71} \preccurlyeq \ell_{82}$ and $\ell_{51} \preccurlyeq \ell_{82}$. But $\ell_{51} \preccurlyeq \ell_{21} \preccurlyeq \ell_{21} \curlyvee \ell_{41} \curlyvee \ell_{71} \preccurlyeq (\ell_{22} \curlyvee \ell_{42}) \curlywedge \ell_{72} \preccurlyeq \ell_{72}$, $\ell_{51} \preccurlyeq \ell_{21} \preccurlyeq \ell_{21} \curlyvee \ell_{41} \curlyvee \ell_{71} \preccurlyeq \ell_{82}$ and $\ell_{71} \preccurlyeq \ell_{21} \curlyvee \ell_{41} \curlyvee \ell_{71} \preccurlyeq \ell_{82}$.

Therefore

$$\varepsilon_0' \circ^{\preccurlyeq} \varepsilon_3 = \langle [\ell_{51}, \ell_{52} \curlywedge \ell_{72} \curlywedge \ell_{82}], [\ell_{51} \curlyvee \ell_{71} \curlyvee \ell_{81}, \ell_{82}] \rangle$$

Using the same method, $\varepsilon_1 \circ^{\preccurlyeq} (\varepsilon_0' \circ^{\preccurlyeq} \varepsilon_3)$ is defined if $\ell_{21} \curlyvee \ell_{51} \preccurlyeq \ell_{22} \curlywedge (\ell_{52} \curlywedge \ell_{72} \curlywedge \ell_{82})$, $\ell_{11} \preccurlyeq \ell_{22} \curlywedge (\ell_{52} \curlywedge \ell_{72} \curlywedge \ell_{82})$, and $\ell_{11} \preccurlyeq \ell_{82}$.

But by definition of $\imath$ $\ell_{21} \preccurlyeq \ell_{22}$, also $\ell_{21} \preccurlyeq \ell_{22} \preccurlyeq \ell_{52}$, $\ell_{21} \preccurlyeq \ell_{21} \curlyvee \ell_{41} \curlyvee \ell_{71} \preccurlyeq (\ell_{22} \curlyvee \ell_{42}) \curlywedge \ell_{72} \preccurlyeq \ell_{72}$, $\ell_{21} \preccurlyeq \ell_{21} \curlyvee \ell_{41} \curlyvee \ell_{71} \preccurlyeq \ell_{82}$, and $\ell_{51} \preccurlyeq \ell_{71} \preccurlyeq \ell_{72}$, therefore $\ell_{21} \curlyvee \ell_{51} \preccurlyeq \ell_{22} \curlywedge (\ell_{52} \curlywedge \ell_{72} \curlywedge \ell_{82})$.

Also $\ell_{11} \preccurlyeq \ell_{22} \preccurlyeq \ell_{52}$, $\ell_{11} \preccurlyeq \ell_{11} \curlyvee \ell_{31} \preccurlyeq (\ell_{22} \curlyvee \ell_{42}) \curlywedge \ell_{72} \preccurlyeq \ell_{72}$, and $\ell_{11} \preccurlyeq \ell_{11} \curlyvee \ell_{31} \preccurlyeq \ell_{82}$, therefore $\ell_{11} \preccurlyeq \ell_{22} \curlywedge (\ell_{52} \curlywedge \ell_{72} \curlywedge \ell_{82})$, and $\ell_{11} \preccurlyeq \ell_{82}$.

Then as $\varepsilon_1 \circ^{\preccurlyeq} (\varepsilon_0' \circ^{\preccurlyeq} \varepsilon_3)$ is defined then if we choose $\varepsilon_2 = (\varepsilon_0' \circ^{\preccurlyeq} \varepsilon_3) \vdash \tilde{\ell}' \preccurlyeq \ell_o$, the result holds.

□

Lemma 55. *Consider $\varepsilon_1, \varepsilon_2$ and ε_3, such that $\varepsilon_1 \vdash \tilde{\ell}_1 \preccurlyeq \tilde{\ell}_2$, $\varepsilon_2 \vdash \tilde{\ell}_2 \preccurlyeq \tilde{\ell}_3$ and $\varepsilon_3 \vdash \tilde{\ell}_3 \preccurlyeq \tilde{\ell}_4$. $(\varepsilon_1 \circ^{\preccurlyeq} \varepsilon_2) \circ^{\preccurlyeq} \varepsilon_3$ is defined, if and only if, $\varepsilon_1 \circ^{\preccurlyeq} (\varepsilon_2 \circ^{\preccurlyeq} \varepsilon_3)$ is defined.*

Proof. Suppose $\varepsilon_1 = \langle [\ell_{11}, \ell_{12}], [\ell_{21}, \ell_{22}] \rangle$, $\varepsilon_2 = \langle [\ell_{31}, \ell_{32}], [\ell_{41}, \ell_{42}] \rangle$, and $\varepsilon_3 = \langle [\ell_{51}, \ell_{52}], [\ell_{61}, \ell_{62}] \rangle$ Then

$$\begin{array}{ll} & (\varepsilon_1 \circ^{\preccurlyeq} \varepsilon_2) \circ^{\preccurlyeq} \varepsilon_3 \\ = & \triangle^{\preccurlyeq}([\ell_{11}, \ell_{12}], [\ell_{21}, \ell_{22}] \sqcap [\ell_{31}, \ell_{32}], [\ell_{41}, \ell_{42}]) \circ^{\preccurlyeq} \varepsilon_3 \\ = & \triangle^{\preccurlyeq}([\ell_{11}, \ell_{12}], [\ell_{21} \curlyvee \ell_{31}, \ell_{22} \curlywedge \ell_{32}], [\ell_{41}, \ell_{42}]) \circ^{\preccurlyeq} \varepsilon_3 \\ = & \langle [\ell_{11}, \ell_{12} \curlywedge (\ell_{22} \curlywedge \ell_{32}) \curlywedge \ell_{42}], [\ell_{11} \curlyvee (\ell_{21} \curlyvee \ell_{31}) \curlyvee \ell_{41}, \ell_{42}] \rangle \\ & \circ^{\preccurlyeq} \langle [\ell_{51}, \ell_{52}], [\ell_{61}, \ell_{62}] \rangle \\ = & \triangle^{\preccurlyeq}([\ell_{11}, \ell_{12} \curlywedge (\ell_{22} \curlywedge \ell_{32}) \curlywedge \ell_{42}], [\ell_{11} \curlyvee (\ell_{21} \curlyvee \ell_{31}) \curlyvee \ell_{41}, \ell_{42}] \sqcap \\ & [\ell_{51}, \ell_{52}], [\ell_{61}, \ell_{62}]) \\ = & \triangle^{\preccurlyeq}([\ell_{11}, \ell_{12} \curlywedge (\ell_{22} \curlywedge \ell_{32}) \curlywedge \ell_{42}], \\ & [\ell_{11} \curlyvee (\ell_{21} \curlyvee \ell_{31}) \curlyvee \ell_{41} \curlyvee \ell_{51}, \ell_{42} \curlywedge \ell_{52}], \\ & [\ell_{61}, \ell_{62}]) \\ = & \langle [\ell_{11}, \ell_{21}'], [\ell_{61}', \ell_{62}] \rangle \end{array}$$

where $\ell_{21}' = \ell_{12} \curlywedge (\ell_{22} \curlywedge \ell_{32}) \curlywedge \ell_{42} \curlywedge \ell_{52} \curlywedge \ell_{62}$ and $\ell_{61}' = \ell_{11} \curlyvee (\ell_{21} \curlyvee \ell_{31}) \curlyvee \ell_{41} \curlyvee \ell_{51} \curlyvee \ell_{61}$. But

$$\begin{array}{ll} & \varepsilon_1 \circ^{\preccurlyeq} (\varepsilon_2 \circ^{\preccurlyeq} \varepsilon_3) \\ = & \varepsilon_1 \circ^{\preccurlyeq} \triangle^{\preccurlyeq}([\ell_{31}, \ell_{32}], [\ell_{41}, \ell_{42}] \sqcap [\ell_{51}, \ell_{52}], [\ell_{61}, \ell_{62}]) \\ = & \varepsilon_1 \circ^{\preccurlyeq} \triangle^{\preccurlyeq}([\ell_{31}, \ell_{32}], [\ell_{41} \curlyvee \ell_{51}, \ell_{42} \curlywedge \ell_{52}], [\ell_{61}, \ell_{62}]) \\ = & \langle [\ell_{11}, \ell_{12}], [\ell_{21}, \ell_{22}] \rangle \circ^{\preccurlyeq} \\ & \langle [\ell_{31}, \ell_{32} \curlywedge (\ell_{42} \curlywedge \ell_{52}) \curlywedge \ell_{62}], [\ell_{31} \curlyvee (\ell_{41} \curlyvee \ell_{51}) \curlyvee \ell_{61}, \ell_{62}] \rangle \\ = & \triangle^{\preccurlyeq}([\ell_{11}, \ell_{12}], [\ell_{21}, \ell_{22}] \sqcap \\ & [\ell_{31}, \ell_{32} \curlywedge (\ell_{42} \curlywedge \ell_{52}) \curlywedge \ell_{62}], [\ell_{31} \curlyvee (\ell_{41} \curlyvee \ell_{51}) \curlyvee \ell_{61}, \ell_{62}]) \\ = & \triangle^{\preccurlyeq}([\ell_{11}, \ell_{12}], \\ & [\ell_{21}, \ell_{22} \curlywedge (\ell_{32} \curlywedge \ell_{42}) \curlywedge \ell_{52} \curlywedge \ell_{62}], \\ & [\ell_{31} \curlyvee (\ell_{41} \curlyvee \ell_{51}) \curlyvee \ell_{61}, \ell_{62}] \\ = & \langle [\ell_{11}, \ell_{21}'], [\ell_{61}', \ell_{62}] \rangle \end{array}$$

where $\ell_{21}' = \ell_{12} \curlywedge (\ell_{22} \curlywedge \ell_{32}) \curlywedge \ell_{42} \curlywedge \ell_{52} \curlywedge \ell_{62}$ and $\ell_{61}' = \ell_{11} \curlyvee (\ell_{21} \curlyvee \ell_{31}) \curlyvee \ell_{41} \curlyvee \ell_{51} \curlyvee \ell_{61}$, and the result holds. □

Lemma 56. *Consider $\varepsilon_1, \varepsilon_2$ and ε_3 such that $\varepsilon_1 \vdash \tilde{\ell}_1 \preccurlyeq \tilde{\ell}_2$, $\varepsilon_2 \vdash \tilde{\ell}_2 \preccurlyeq \tilde{\ell}_3$ and $\varepsilon_3 \vdash \tilde{\ell}_3 \preccurlyeq \tilde{\ell}_4$. If $\varepsilon_1 \mathrel{\widetilde{\curlyvee}} (\varepsilon_2 \circ^{\preccurlyeq} \varepsilon_3)$ is defined, then $(\varepsilon_1 \mathrel{\widetilde{\curlyvee}} \varepsilon_2) \circ^{\preccurlyeq} (\varepsilon_1 \mathrel{\widetilde{\curlyvee}} \varepsilon_3)$ is defined*

Proof. By definition of join and consistent transitivity, using the property that the join operator is monotone. □

Lemma 57. *If $\nexists \varepsilon_1$, such that $\varepsilon_1 \vdash \tilde{\ell}_1 \lesssim \tilde{\ell}_2$, then $\nexists \varepsilon_2$, such that $\varepsilon_2 \vdash \widetilde{\tilde{\ell}_1 \curlyvee \tilde{\ell}_3 \preccurlyeq \tilde{\ell}_2}$.*

Proof. By definition of join and consistent transitivity, using the property that the join operator is monotone. □

Lemma 58. *Consider stores $\mu_1, \mu_2, \mu_1', \mu_2'$ such that $\mu_i \rightarrowtail \mu_i'$, and substitutions σ_1 and σ_2, such that $\Gamma \vdash \langle \varepsilon_{r1} \tilde{\ell}_{r1}, \tilde{\ell}_{c1}, \sigma_1, \mu_1 \rangle \approx^{k}_{\ell_o} \langle \varepsilon_{r2} \tilde{\ell}_{r2}, \tilde{\ell}_{c2}, \sigma_2, \mu_2 \rangle$, then if $\forall j \leq k$, if $\langle \tilde{\ell}_{r1}, \mu_1' \rangle \approx^{j}_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2' \rangle$ then $\Gamma \vdash \langle \varepsilon_{r1} \tilde{\ell}_{r1}, \tilde{\ell}_{c1}, \sigma_1, \mu_1' \rangle \approx^{j}_{\ell_o} \langle \varepsilon_{r2} \tilde{\ell}_{r2}, \tilde{\ell}_{c2}, \sigma_2, \mu_2' \rangle$*

Proof. By definition of related computations and related stores. The key argument is that given that $\mu_i \rightarrowtail \mu_i'$ then μ_i' have the same locations of μ_i and the values still are related as well given that they still have the same type. □

Lemma 59. *Consider stores μ_1, μ_2 and substitutions σ_1 and σ_2, then if $\Gamma \vdash \langle \varepsilon_{r1} \tilde{\ell}_{r1}, \tilde{\ell}_{c1}, \sigma_1, \mu_1 \rangle \approx^{k}_{\ell_o} \langle \varepsilon_{r2} \tilde{\ell}_{r2}, \tilde{\ell}_{c2}, \sigma_2, \mu_2 \rangle$, then $\forall \tilde{\ell}_{ri}', \tilde{\ell}_{ci}'$ such that $\tilde{\ell}_{ri} \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_{ri}'$, $\tilde{\ell}_{ci} \mathrel{\widetilde{\preccurlyeq}} \tilde{\ell}_{ci}'$ and $\varepsilon_{ri}' \vdash \tilde{\ell}_{ri}' \preccurlyeq \tilde{\ell}_{ci}'$, then $\Gamma \vdash \langle \varepsilon_{r1}' \tilde{\ell}_{r1}', \tilde{\ell}_{c1}', \sigma_1, \mu_1' \rangle \approx^{j}_{\ell_o} \langle \varepsilon_{r2}' \tilde{\ell}_{r2}', \tilde{\ell}_{c2}', \sigma_2, \mu_2' \rangle$.*

Proof. By definition of related values and related stores. A value is typed with any program counter and increasing the security level of the runtime program counters may make the value non-observable and the result holds by definition. □

Lemma 60 (Substitution preserves typing)**.** *If* $e\ell \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}} \in \text{TERM}_{\widetilde{S}}$ *and* $\sigma \models FV(t^{\widetilde{S}})$ *then* $e\ell \overset{\tilde{\ell}_c}{\triangleright} \sigma(t^{\widetilde{S}}) \in \text{TERM}_{\widetilde{S}}$.

Proof. By induction on the derivation of $e\ell \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}} \in \text{TERM}_{\widetilde{S}}$ □

Lemma 61 (Reduction preserves relations)**.** *Consider* $\varepsilon_{ri}\tilde{\ell}_{ri} \overset{\tilde{\ell}_c}{\triangleright} t_i \in \text{TERM}_{\widetilde{S}}$, $\mu_i \in \text{STORE}$, $t_i \vdash \mu_i$, *and* $\langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle$. *Consider* $j < k$, *posing*
$t_i \mid \mu_i \overset{\varepsilon_{ri}\tilde{\ell}_{ri}}{\underset{\tilde{\ell}_c}{\longmapsto}}{}^{j} t_i' \mid \mu_i'$, *we have*
$\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} t_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} t_2, \mu_2\rangle : \mathcal{C}(\widetilde{S})$ *if and only if*
$\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} t_1', \mu_1'\rangle \approx^{k-j}_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} t_2', \mu_2'\rangle : \mathcal{C}(\widetilde{S})$

Proof. Direct by definition of
$\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} t_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} t_2, \mu_2\rangle : \mathcal{C}(\widetilde{S})$ and transitivity of $\overset{e\ell}{\longmapsto}{}^{j}_{\tilde{\ell}_c}$. □

Lemma 62. *Suppose* $\varepsilon \vdash \widetilde{S}' \lesssim \widetilde{S}$ *and* $\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v \in \text{TERM}_{\widetilde{S}'}$. *If* $\mathsf{obs}_{\ell_o}(\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v)$ *does not hold, then* $\mathsf{obs}_{\ell_o}(\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon v :: \widetilde{S})$ *does not hold.*

Proof. If $\nexists \varepsilon_1, \varepsilon_1 \vdash \tilde{\ell}_r \precsim \ell_o$ then the result holds immediately. A value v may be a bare value or an ascribed value $\varepsilon' u :: \widetilde{S}'$.

If value v is a bare values then if $\tilde{\ell}' = label(\widetilde{S}')$ and $\tilde{\ell} = label(\widetilde{S})$, then suppose $\exists \varepsilon_2', \varepsilon_2' \vdash \tilde{\ell} \precsim \ell_o$ (otherwise the result holds immediately). Then $\varepsilon \circ^{\prec} \varepsilon_2' \vdash \tilde{\ell}' \precsim \ell_o$, but u is not observable meaning that $\nexists \varepsilon_2, \varepsilon_2 \precsim \ell_o$. Therefore $\varepsilon \circ^{\prec} \varepsilon_2'$ is not defined, and the result holds.

If v is an ascribed value $\varepsilon' u :: \widetilde{S}'$, and suppose $\exists \varepsilon_2', \varepsilon_2' \vdash \tilde{\ell} \precsim \ell_o$ (otherwise the result holds immediately). Then we need to prove that $(\varepsilon' \circ^{\prec} \varepsilon) \circ^{\prec} \varepsilon_2'$ is not defined. But as v is not observable if $\nexists \varepsilon_2 \vdash \tilde{\ell}' \precsim \ell_o$, then by Lemma 55, if $(\varepsilon' \circ^{\prec} \varepsilon) \circ^{\prec} \varepsilon_2'$ is defined if and only if $\varepsilon' \circ^{\prec} (\varepsilon \circ^{\prec} \varepsilon_2')$ is defined. But $(\varepsilon \circ^{\prec} \varepsilon_2') \vdash \tilde{\ell}' \precsim \ell_o$ which is a contradiction and therefore $(\varepsilon' \circ^{\prec} \varepsilon) \circ^{\prec} \varepsilon_2'$ is not defined. Similar argument is applied if $\forall \varepsilon_2$ such that $\varepsilon_2 \vdash \tilde{\ell}' \precsim \ell_o$ and $\varepsilon' \circ^{\prec} \varepsilon_2$ is not defined. If $\varepsilon' \circ^{\prec} (\varepsilon \circ^{\prec} \varepsilon_2')$ is defined then it means that we have found an evidence such that $\varepsilon_2 \vdash \tilde{\ell}' \precsim \ell_o$ and $\varepsilon' \circ^{\prec} \varepsilon_2$ is defined which is a contradiction and the result holds.

□

Lemma 63 (Ascription preserves relation)**.** *Suppose* $\varepsilon \vdash \widetilde{S}' \lesssim \widetilde{S}$.

1. *If* $\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v_2, \mu_2\rangle : \widetilde{S}'$ *then*
   $\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} \varepsilon v_1 :: \widetilde{S}, \mu_1\rangle \approx^{k+1}_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} \varepsilon v_2 :: \widetilde{S}, \mu_2\rangle : \mathcal{C}(\widetilde{S})$.
2. *If* $\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} t_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} t_2, \mu_2\rangle : \mathcal{C}(\widetilde{S}')$ *then*
   $\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} \varepsilon t_1 :: \widetilde{S}, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} \varepsilon t_2 :: \widetilde{S}, \mu_2\rangle : \mathcal{C}(\widetilde{S})$.

Proof. Following Zdancewic (2002), the proof proceeds by induction on the judgment $\varepsilon \vdash \widetilde{S}' \lesssim \widetilde{S}$. The difference here is that consistent subtyping is justified by evidence, and that the terms have to be ascribed to exploit subtyping. In particular, case 1 above establishes a computation-level relation because each ascribed term $(\varepsilon v_i :: \widetilde{S})$ may not be a value: each value v_i is either a bare value u_i or a casted value $\varepsilon_i u_i :: \widetilde{S}_i$, with $\varepsilon_i \vdash S_i \lesssim \widetilde{S}$. In the latter case, $(\varepsilon(\varepsilon_i u_i :: \widetilde{S}_i) :: \widetilde{S})$ either steps to **error** (in which case the relation is vacuously established), or steps to $\varepsilon' u_i :: \widetilde{S}$, which is a value. Next if both values were originally observables, then whatever the label of $\widetilde{S}$ both values are going to be related. If both values were originally not observables, then by Lemma 63 both values are going to be still non observables.

□

Lemma 64. *Suppose* $\tilde{\ell}_r \precsim \tilde{\ell}_c$, $\tilde{\ell} \precsim \tilde{\ell}_c$ *and* $\varepsilon_r' \vdash \widetilde{\tilde{\ell}_r \curlyvee \tilde{\ell}} \precsim \tilde{\ell}_c$

1. *If* $\langle \varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_2, \mu_2\rangle : \widetilde{S}$ *then*
   $\langle \varepsilon_r'(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}) \overset{\tilde{\ell}_c}{\triangleright} v_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r'(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}) \overset{\tilde{\ell}_c}{\triangleright} v_2, \mu_2\rangle : \widetilde{S}$
2. *If* $\langle \varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t_1^{\widetilde{S}}, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t_2^{\widetilde{S}}, \mu_2\rangle : \mathcal{C}(\widetilde{S})$ *then*
   $\langle \varepsilon_r'(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}) \overset{\tilde{\ell}_c}{\triangleright} t_1^{\widetilde{S}}, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r'(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}) \overset{\tilde{\ell}_c}{\triangleright} t_2^{\widetilde{S}}, \mu_2\rangle : \mathcal{C}(\widetilde{S})$

Proof. By the logical relation definition and by induction on type $\widetilde{S}$. The only condition to determinate if two values are related that depends on the current PC is that $\tilde{\ell}_r \precsim \ell_o$. But if that condition does not hold, then $\tilde{\ell}_r \neq ?$ and therefore $\widetilde{\tilde{\ell}_r \curlyvee \tilde{\ell}} \precsim \ell_o$ does not hold as well.

□

Lemma 65. *If* $\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v_2, \mu_2\rangle : \widetilde{S}$ *and,* $\varepsilon_{ri}\tilde{\ell}_{ri} \overset{\tilde{\ell}_{ci}}{\triangleright} uval(v_i) \in \text{TERM}_{\widetilde{S}_i}$ *where* $\widetilde{S}_i \lesssim \widetilde{S}$, *then* $\forall \widetilde{S}', \widetilde{S} \lesssim \widetilde{S}', \varepsilon_i \vdash \widetilde{S}_i \lesssim \widetilde{S}'$,
$\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} \varepsilon_1 uval(v_1) :: \widetilde{S}', \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} \varepsilon_2 uval(v_2) :: \widetilde{S}', \mu_1\rangle : \widetilde{S}'$.

Proof. Consider $\widetilde{S}'$ and ε, such that $\varepsilon \vdash \widetilde{S} \lesssim \widetilde{S}'$. By Lemma 63.1, $\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} \varepsilon v_1 :: \widetilde{S}', \mu_1\rangle \approx^{k+1}_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} \varepsilon v_2 :: \widetilde{S}', \mu_2\rangle : \mathcal{C}(\widetilde{S})$. Next we consider the case were evidence combination do not fails. In case of a failure the lemma vacuously holds. Then
$\varepsilon v_i :: \widetilde{S}' \mid \mu_i \overset{\varepsilon_{ri}\tilde{\ell}_{ri}}{\underset{\tilde{\ell}_c}{\longmapsto}} \varepsilon_i uval(v_i) :: \widetilde{S}' \mid \mu_i$ and the result follows using Lemma 61.

□

Lemma 66 (Downward Closed / Monotonicity)**.** *If*

1. $\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v_2, \mu_2\rangle : \widetilde{S}$ *then*
   $\forall j \leq k, \langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} v_1, \mu_1\rangle \approx^j_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} v_2, \mu_2\rangle : \widetilde{S}$
2. $\langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} t_1^{\widetilde{S}}, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} t_2^{\widetilde{S}}, \mu_2\rangle : \mathcal{C}(\widetilde{S})$ *then*
   $\forall j \leq k, \langle \varepsilon_{r1}\tilde{\ell}_{r1} \overset{\tilde{\ell}_{c1}}{\triangleright} t_1^{\widetilde{S}}, \mu_1\rangle \approx^j_{\ell_o} \langle \varepsilon_{r2}\tilde{\ell}_{r2} \overset{\tilde{\ell}_{c2}}{\triangleright} t_2^{\widetilde{S}}, \mu_2\rangle : \mathcal{C}(\widetilde{S})$
3. $\langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle$ *then* $\forall j \leq k, \langle \tilde{\ell}_{r1}, \mu_1\rangle \approx^j_{\ell_o} \langle \tilde{\ell}_{r2}, \mu_2\rangle$

Proof. By induction on type $\widetilde{S}$ and the definition of related stores. □

Lemma 67. *Consider* $\tilde{\ell}_r, \tilde{\ell}_c, \tilde{\ell}_r', \tilde{\ell}, \varepsilon, \ell_o$, *such that* $\varepsilon \vdash \tilde{\ell}' \precsim \tilde{\ell}$ *and* $\varepsilon_r \vdash \tilde{\ell}_r \precsim \tilde{\ell}_c$. *If* $\tilde{\ell}_r \precsim \ell_o$ *or* $\tilde{\ell} \precsim \ell_o$ *does not hold, or if* $\forall \varepsilon_2$, *such that* $\varepsilon_2 \vdash \tilde{\ell} \precsim \ell_o$ *then* $\varepsilon \circ^{\prec} \varepsilon_2$ *is not defined, then* $\forall k > 0$, *such that* $t^{\widetilde{S}} \mid \mu \overset{(\varepsilon_r \mathbin{\widetilde{\curlyvee}} \varepsilon)(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}')}{\underset{(\tilde{\ell}_c \mathbin{\widetilde{\curlyvee}} \tilde{\ell})}{\longmapsto}}{}^{k} t'^{\widetilde{S}} \mid \mu'$, *then*

1. $\forall l^{\widetilde{S}'} \in dom(\mu') \backslash dom(\mu)$, $\mathsf{obs}_{\ell_o}(\varepsilon_r\tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mu'(l^{\widetilde{S}'}))$ *does not hold.*

2. $\forall l^{\widetilde{S'}} \in dom(\mu') \cap dom(\mu) \wedge \mu'(l^{\widetilde{S_l}}) \neq \mu(l^{\widetilde{S'}})$, $\mathsf{obs}_{\ell_o}(\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mu'(l^{\widetilde{S_l}}))$ *does not hold.*

Proof. We use induction on the derivation of $t^{\widetilde{S}}$. The interest cases are the last step of reduction rules for references and assignments.

Let us call $\varepsilon'_r = (\varepsilon_r \mathbin{\widetilde{\Upsilon}} \varepsilon)$. We show the case for the last step of reduction rules for references (the assignment case is analogous):

Then $t^{\widetilde{S}} = \mathsf{ref}^{\widetilde{S'}}_{\varepsilon_\ell}\ \varepsilon_s u$ and

$$\text{(Iref)}\ \frac{\begin{array}{c}\varepsilon'_r \vdash \widetilde{\tilde{\ell}_r \mathbin{\Upsilon} \tilde{\ell}' \preccurlyeq \tilde{\ell}_c \mathbin{\Upsilon} \tilde{\ell}} \\ \varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}') \overset{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}}{\triangleright} u \in \textsc{Term}_{\widetilde{S''}} \\ \varepsilon_s \vdash \widetilde{S''} \lesssim \widetilde{S'} \qquad \varepsilon_\ell \vdash \widetilde{\tilde{\ell}_c \mathbin{\widetilde{\Upsilon}} \tilde{\ell} \preccurlyeq label(\widetilde{S'})}\end{array}}{\varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}') \overset{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}}{\triangleright} \mathsf{ref}^{\widetilde{S'}}_{\varepsilon_\ell}\ \varepsilon_s u \in \textsc{Term}_{\mathsf{Ref}_\perp\ \widetilde{S'}}}$$

Therefore

$$\mathsf{ref}^{\widetilde{S}}_{\varepsilon_\ell}\ \varepsilon_s u \mid \mu \xrightarrow{\varepsilon'_r(\tilde{\ell}_r \widetilde{\Upsilon} \tilde{\ell}') \overset{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}}{\triangleright}} l^{\widetilde{S'}}_\perp \mid \mu[l^{\widetilde{S'}} \mapsto \varepsilon'(u \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}') :: \widetilde{S'}]$$
$$\text{where } l^{\widetilde{S'}} \notin dom(\mu), \varepsilon' = \varepsilon_s \mathbin{\widetilde{\Upsilon}} (\varepsilon'_r \circ^{\preccurlyeq} \varepsilon_\ell)$$

We need to prove that $\varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}') \overset{\tilde{\ell}_c}{\triangleright} \varepsilon'(u \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}') :: \widetilde{S'}$ is not observable at ℓ_o.

If $\nexists \varepsilon_{o1}, \varepsilon_{o1} \vdash \tilde{\ell}_r \preccurlyeq \ell_o$ then as join is monotone $\nexists \varepsilon'_{o1}, \varepsilon'_{o1} \vdash \widetilde{\tilde{\ell}_r \mathbin{\Upsilon} \tilde{\ell}' \preccurlyeq \ell_o}$ and the result holds.

If $\nexists \varepsilon_{o2}, \varepsilon_{o2} \vdash \tilde{\ell} \preccurlyeq \ell_o$.

Then let us assume $\varepsilon'_{o2} \vdash label(\widetilde{S'}) \lesssim \ell_o$ (otherwise the result holds immediately). We have to prove that $(ilbl(\varepsilon_s) \mathbin{\widetilde{\Upsilon}} ((\varepsilon_r \mathbin{\widetilde{\Upsilon}} \varepsilon) \circ^{\preccurlyeq} \varepsilon_\ell)) \circ^{\preccurlyeq} \varepsilon'_{o2}$ is not defined. By Lemma 57, $\nexists \varepsilon'_{o2}, \varepsilon'_{o2} \vdash \widetilde{label(\widetilde{S''}) \mathbin{\Upsilon} \tilde{\ell}_c \mathbin{\Upsilon} \tilde{\ell} \preccurlyeq \ell_o}$. We proceed by contradiction. Let us suppose that $(ilbl(\varepsilon_s) \mathbin{\widetilde{\Upsilon}} ((\varepsilon_r \mathbin{\widetilde{\Upsilon}} \varepsilon) \circ^{\preccurlyeq} \varepsilon_\ell)) \circ^{\preccurlyeq} \varepsilon'_{o2}$ is defined. Then by Lemma 55 and 56 $((ilbl(\varepsilon_s) \mathbin{\widetilde{\Upsilon}} \varepsilon_r) \mathbin{\widetilde{\Upsilon}} \varepsilon) \circ^{\preccurlyeq} ((\widetilde{ilbl(\varepsilon_s) \mathbin{\Upsilon} \varepsilon_\ell}) \circ^{\preccurlyeq} \varepsilon'_{o2})$ is defined. But $((ilbl(\varepsilon_s) \mathbin{\widetilde{\Upsilon}} \varepsilon_\ell) \circ^{\preccurlyeq} \varepsilon'_{o2}) \vdash \widetilde{label(\widetilde{S''}) \mathbin{\Upsilon} \tilde{\ell}_c \mathbin{\Upsilon} \tilde{\ell} \preccurlyeq \ell_o}$, which is a contradiction and then the result holds immediately.

If $\forall \varepsilon_{o2}, \varepsilon_{o2} \vdash \tilde{\ell} \preccurlyeq \ell_o$ and $\varepsilon \circ^{\preccurlyeq} \varepsilon_{o2}$ does not hold. Consider $\varepsilon'_{o2} \vdash \widetilde{label(\widetilde{S'}) \preccurlyeq \ell_o}$. Also $ilbl(\varepsilon') = ilbl(\varepsilon_s) \mathbin{\widetilde{\Upsilon}} (\varepsilon'_r \circ^{\preccurlyeq} \varepsilon_\ell) = ilbl(\varepsilon_s) \mathbin{\widetilde{\Upsilon}} ((\varepsilon_r \mathbin{\widetilde{\Upsilon}} \varepsilon) \circ^{\preccurlyeq} \varepsilon_\ell)$. Let us suppose that $(ilbl(\varepsilon_s) \mathbin{\widetilde{\Upsilon}} ((\varepsilon_r \mathbin{\widetilde{\Upsilon}} \varepsilon) \circ^{\preccurlyeq} \varepsilon_\ell)) \circ^{\preccurlyeq} \varepsilon'_{o2}$ is defined. The by Lemma 56 and 55, $((ilbl(\varepsilon_s) \mathbin{\widetilde{\Upsilon}} \varepsilon_r) \mathbin{\widetilde{\Upsilon}} \varepsilon) \circ^{\preccurlyeq} ((ilbl(\varepsilon_s) \mathbin{\widetilde{\Upsilon}} \varepsilon_\ell) \circ^{\preccurlyeq} \varepsilon'_{o2})$ is defined. But by Lemma 54, $((ilbl(\varepsilon_s) \mathbin{\widetilde{\Upsilon}} \varepsilon_r) \mathbin{\widetilde{\Upsilon}} \varepsilon) \circ^{\preccurlyeq} ((\widetilde{ilbl(\varepsilon_s) \mathbin{\Upsilon} \varepsilon_\ell}) \circ^{\preccurlyeq} \varepsilon'_{o2})$ is not defined, and therefore $(ilbl(\varepsilon_s) \mathbin{\widetilde{\Upsilon}} ((\varepsilon_r \mathbin{\widetilde{\Upsilon}} \varepsilon) \circ^{\preccurlyeq} \varepsilon_\ell)) \circ^{\preccurlyeq} \varepsilon'_{o2}$ is not defined and the result holds. □

Lemma 68. *If* $\mathsf{obs}_{\ell_o}(\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon u :: \widetilde{S})$ *does not hold, then if* $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} u \in \textsc{Term}_{\widetilde{S'}}$, $\tilde{\ell}' = label(\widetilde{S'})$, $\tilde{\ell} = label(\widetilde{S})$ *then,* $\forall k > 0$, *such that* $t^{\widetilde{S_t}} \mid \mu \xrightarrow{(\varepsilon_r \widetilde{\Upsilon} ilbl(\varepsilon)) (\tilde{\ell}_r \widetilde{\Upsilon} \tilde{\ell}')}{}^{k}_{(\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell})} t'^{\widetilde{S_t}} \mid \mu'$, *then*

1. $\forall l^{\widetilde{S_l}} \in dom(\mu') \backslash dom(\mu)$, $\mathsf{obs}_{\ell_o}(\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mu'(l^{\widetilde{S_l}}))$ *does not hold.*
2. $\forall l^{\widetilde{S_l}} \in dom(\mu') \cap dom(\mu) \wedge \mu'(l^{\widetilde{S_l}}) \neq \mu(l^{\widetilde{S_l}})$, $\mathsf{obs}_{\ell_o}(\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mu'(l^{\widetilde{S_l}}))$ *does not hold.*

Proof. If $\mathsf{obs}_{\ell_o}(\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon u :: \widetilde{S})$ does not hold, then either / $\exists \varepsilon_{o1}, \varepsilon_{o1} \vdash \tilde{\ell}_r \preccurlyeq \ell_o$, $\nexists \varepsilon_{o2}, \varepsilon_{o2} \vdash \tilde{\ell} \preccurlyeq \ell_o$, or $\exists \varepsilon_{o2}, \varepsilon_{o2} \vdash \tilde{\ell} \preccurlyeq \ell_o$ and $ilbl(\varepsilon) \circ^{\preccurlyeq} \varepsilon_{o2}$ does not hold. Then proof follows directly using Lemma 67. □

Lemma 69. *If* $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 u_1 :: \widetilde{S}, \mu_1 \rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_2 u_2 :: \widetilde{S}, \mu_2 \rangle : \widetilde{S}$ *, and* $\mathsf{obs}_{\ell_o}(\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_i u_i :: \widetilde{S})$ *does not hold for any* $i \in \{1, 2\}$, *then if* $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} u_i \in \textsc{Term}_{\widetilde{S'_i}}$, $\tilde{\ell}'_i = label(\widetilde{S'_i})$, $\tilde{\ell} = label(\widetilde{S})$ *then,* $\forall j < k$, *such that* $t^{\widetilde{S_i}} \mid \mu_i \xrightarrow{(\varepsilon_r \widetilde{\Upsilon} ilbl(\varepsilon_i)) (\tilde{\ell}_r \widetilde{\Upsilon} \tilde{\ell}')}{}^{j}_{(\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell})} t'^{\widetilde{S_i}} \mid \mu'_i$, *then* $\langle \tilde{\ell}_r, \mu'_1 \rangle \approx^{k-j}_{\ell_o} \langle \tilde{\ell}_r, \mu'_2 \rangle$.

Proof. By induction on the derivation of $t^{\widetilde{S_i}}$. The key argument uses Lemma 68, to state that if one of the values are non observables, then all values written to the store are non observables and therefore the resulting stores are related by definition of related stores. □

Lemma 70 (Protection preserves values relation)**.** *If* $\langle \varepsilon'_{ri}(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_1) \overset{\tilde{\ell}'_c}{\triangleright} v_1, \mu_1 \rangle \approx^k_{\ell_o} \langle \varepsilon'_{ri}(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_2) \overset{\tilde{\ell}'_c}{\triangleright} v_2, \mu_2 \rangle : \mathcal{C}(\widetilde{S})$, *and* $\langle \tilde{\ell}_r, \mu_1 \rangle \approx^k_{\ell_o} \langle \tilde{\ell}_r, \mu_2 \rangle$ *then* $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}', \widetilde{S'}}_{\varepsilon'_{r1}, \varepsilon_{\ell 1} \tilde{\ell}_1} \varepsilon v_1, \mu_1 \rangle \approx^{k+2}_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}', \widetilde{S'}}_{\varepsilon'_{r2}, \varepsilon_{\ell 2} \tilde{\ell}_2} \varepsilon v_2, \mu_2 \rangle : \mathcal{C}(\widetilde{S'} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}')$.

Proof. Suppose $v_i = \varepsilon_i u_i :: \widetilde{S}$, and $\varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_i) \overset{\tilde{\ell}_c}{\triangleright} u_i \in \textsc{Term}_{\widetilde{S_u}}$. Let us assume that $\varepsilon'_i = \varepsilon_i \circ^{<:} \varepsilon$ is defined (otherwise the lemma vacuously holds). Then

$$\begin{array}{ll} & \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}', \widetilde{S'}}_{\varepsilon'_{ri}, \varepsilon_{\ell i} \tilde{\ell}_i} \varepsilon(\varepsilon_i u_i :: \widetilde{S}) \mid \mu_i \\ \xrightarrow{\varepsilon_r \tilde{\ell}_r}{}^{1}_{\tilde{\ell}_c} & \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}', \widetilde{S'}}_{\varepsilon'_{ri}, \varepsilon_{\ell i} \tilde{\ell}_i} \varepsilon'_i u_i \mid \mu_i \\ \xrightarrow{\varepsilon_r \tilde{\ell}_r}{}^{1}_{\tilde{\ell}_c} & (\varepsilon'_i \mathbin{\widetilde{\Upsilon}} \varepsilon_{\ell i})(u_i \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_i) :: \widetilde{S'} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}' \mid \mu_i \end{array}$$

Let us call $v'_i = (\varepsilon'_i \mathbin{\widetilde{\Upsilon}} \varepsilon_{\ell i})(u_i \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_i) :: \widetilde{S'} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}'$. If we prove that $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v'_1, \mu_1 \rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v'_2, \mu_2 \rangle : \widetilde{S'} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}'$, then the result holds by backward preservation of the relations (Lemma 61).

By preservation we know that $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v'_i \in \textsc{Term}_{\widetilde{S'} \widetilde{\Upsilon} \tilde{\ell}'}$. We also know that $\langle \tilde{\ell}_r, \mu_1 \rangle \approx^k_{\ell_o} \langle \tilde{\ell}_r, \mu_2 \rangle$. If $\mathsf{obs}_{\ell_o}(\varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}) \overset{\tilde{\ell}'_c}{\triangleright} v_i)$, then $rval(v_1) = rval(v_2) = rval(v'_1) = rval(v'_2)$ and the result holds immediately. If $\mathsf{obs}_{\ell_o}(\varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}) \overset{\tilde{\ell}'_c}{\triangleright} v_i)$ does not hold, then we need to prove that $\mathsf{obs}_{\ell_o}(\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v'_i)$ does not hold as well.

Which is equivalent to prove that if $\mathsf{obs}_{\ell_o}(\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v'_i)$ then $\mathsf{obs}_{\ell_o}(\varepsilon'_r(\tilde{\ell}_r \widetilde{\Upsilon} \tilde{\ell}) \overset{\tilde{\ell}'_c}{\triangleright} v_i)$. If $\varepsilon'_1 \vdash \tilde{\ell}_r \preccurlyeq \ell_o$ and $\varepsilon'_2 \vdash label(\widetilde{S'}) \mathbin{\Upsilon} \tilde{\ell}' \preccurlyeq \ell_o$, as $v'_i = (\varepsilon'_i \mathbin{\widetilde{\Upsilon}} \varepsilon_{\ell i})(u \mathbin{\widetilde{\Upsilon}} \tilde{\ell}) :: \widetilde{S'} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}'$ then $\widetilde{ilbl(\varepsilon'_i \mathbin{\Upsilon} \varepsilon_\ell)} \circ^{\preccurlyeq} \varepsilon'_2$ is defined. By Prop 7, $(\varepsilon'_1 \mathbin{\widetilde{\Upsilon}} \varepsilon'_2) \vdash \tilde{\ell}_r \mathbin{\Upsilon} \widetilde{label(\widetilde{S'}) \mathbin{\Upsilon} \tilde{\ell}} \preccurlyeq \ell_o$ and therefore as the join is monotone, $\exists \varepsilon_1, \varepsilon_1 \vdash \widetilde{(\tilde{\ell}_r \mathbin{\Upsilon} \tilde{\ell})} \preccurlyeq \ell_o$ and $\exists \varepsilon_2, \varepsilon_2 \vdash label(\widetilde{S}) \preccurlyeq \ell_o$.

As $v_i = \varepsilon_i u_i :: \widetilde{S}$ then $\varepsilon_i \vdash \widetilde{S}_{ui} \lesssim \widetilde{S}$. As $\widetilde{ilbl(\varepsilon'_i \mathbin{\widetilde{\Upsilon}} \varepsilon_\ell)} \circ^{\preccurlyeq} \varepsilon'_2$ is defined, therefore $ilbl(\varepsilon'_i \mathbin{\widetilde{\Upsilon}} \varepsilon_\ell) \circ^{\preccurlyeq} \varepsilon'_2 \vdash \widetilde{label(\widetilde{S}_{ui}) \mathbin{\Upsilon} \tilde{\ell} \preccurlyeq \ell_o}$. We need to find evidence ε_2 such that $\varepsilon_2 \vdash label(\widetilde{S}) \preccurlyeq \ell_o$, and that $ilbl(\varepsilon_i) \circ^{\preccurlyeq} \varepsilon_2$ is defined to justify that $(ilbl(\varepsilon_i) \circ^{\preccurlyeq} \varepsilon_2) \vdash label(\widetilde{S}_{ui}) \lesssim \ell_o$.

As the join is monotone, and as $\widetilde{ilbl(\varepsilon'_i \mathbin{\Upsilon} \varepsilon_\ell)} \circ^{\preccurlyeq} \varepsilon'_2$ is defined then $ilbl(\varepsilon'_i) \circ^{\preccurlyeq} \varepsilon'_2$ is defined as well. We know that $\varepsilon'_i = (\varepsilon_i \circ^{\preccurlyeq} \varepsilon) \vdash$

$\widetilde{S}_{ui} \lesssim \widetilde{S}$. Therefore by Lemma 55 if $ilbl(\varepsilon'_i) \circ^{\preccurlyeq} \varepsilon'_2$ is defined then $ilbl(\varepsilon_i) \circ^{\preccurlyeq} (\varepsilon \circ^{\preccurlyeq} \varepsilon'_2)$ is defined but $(\varepsilon \circ^{\preccurlyeq} \varepsilon'_2) \vdash label(\widetilde{S}) \lesssim \ell_o$ and then v_i is observable and the result holds.

□

Lemma 71 (Protection preserves terms relation)**.** *If* $\langle \varepsilon'_{ri}(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_1) \overset{\tilde{\ell}'_c}{\triangleright} t_1^{\widetilde{S}}, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon'_{ri}(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_2) \overset{\tilde{\ell}'_c}{\triangleright} t_2^{\widetilde{S}}, \mu_2\rangle : \mathcal{C}(\widetilde{S})$*, then if* $\langle \tilde{\ell}_r, \mu_1\rangle \approx^k_{\ell_o} \langle \tilde{\ell}_r, \mu_2\rangle$ *and* $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}', \widetilde{S}'}_{\varepsilon'_{ri}, \varepsilon_{\ell i} \tilde{\ell}_i} \varepsilon t_i^{\widetilde{S}} \in \textsc{Term}_{\widetilde{S}'}$*, then* $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}', \widetilde{S}'}_{\varepsilon'_{r1}, \varepsilon_{\ell i} \tilde{\ell}_1} \varepsilon t_1^{\widetilde{S}}, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}', \widetilde{S}'}_{\varepsilon'_{r2}, \varepsilon_{\ell i} \tilde{\ell}_2} \varepsilon t_2^{\widetilde{S}}, \mu_2\rangle : \mathcal{C}(\widetilde{S}' \mathbin{\widetilde{\curlyvee}} \tilde{\ell}')$.

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

$$t_i^{\widetilde{S}} \mid \mu_i \overset{\varepsilon'_{ri}(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_i)}{\longmapsto}{}^{j}_{\tilde{\ell}'_c} t'^{\widetilde{S}}_i \mid \mu'_i$$

and that $\langle \tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_1, \mu'_1\rangle \approx^j_{\ell_o} \langle \tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_2, \mu'_2\rangle, \mu_i \twoheadrightarrow \mu'_i$. As $t^{\widetilde{S}}$ is reduced using evidence label $\varepsilon'_{ri}(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_i)$ then using Lemma 67, if a value is non observable with evidence label $\varepsilon'_{ri}(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_i)$ then is non observable with evidence label $\varepsilon_r \tilde{\ell}_r$ and therefore $\langle \tilde{\ell}_r, \mu'_1\rangle \approx^j_{\ell_o} \langle \tilde{\ell}_r, \mu'_2\rangle, \mu_i \twoheadrightarrow \mu'_i$. If $t'^{\widetilde{S}}_i$ are reducible after $k-1$ steps, then the result holds immediately by (Rprot). The interest case if $t'^{\widetilde{S}}_i$ are irreducible after $j < k$ steps:

Suppose that after j steps $t'^{\widetilde{S}}_i = v_i$, then $\langle \varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_1) \overset{\tilde{\ell}'_c}{\triangleright} v_1, \mu'_1\rangle \approx^{k-j}_{\ell_o} \langle \varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}_2) \overset{\tilde{\ell}'_c}{\triangleright} v_2, \mu'_2\rangle : \widetilde{S}$. By Lemma 70 and by backward preservation of the relations (Lemma 61) the result holds.

□

Lemma 72. *Consider simple values* $u_i \in \textsc{Term}_{\widetilde{S}_i}$ *and* $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 u_1 :: \widetilde{S}, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_2 u_2 :: \widetilde{S}, \mu_2\rangle : \widetilde{S}$. *If* $\varepsilon \vdash \tilde{\ell} \lesssim \widetilde{S}$*, then*

$$\begin{array}{rl} & \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} (\varepsilon'_1 \mathbin{\widetilde{\curlyvee}} \varepsilon)(u_1 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}) :: \widetilde{S}, \mu_1\rangle \\ \approx^k_{\ell_o} & \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} (\varepsilon'_2 \mathbin{\widetilde{\curlyvee}} \varepsilon))(u_2 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}) :: \widetilde{S}, \mu_2\rangle : \widetilde{S} \end{array}$$

Proof. By definition of related values, considering that the label stamping can make the new values non observable and that join of evidences does not introduce imprecision. □

Proposition 73 (Noninterference)**.** *If* $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}} \in \textsc{Term}_{\widetilde{S}}$, $\mu_i \in \textsc{Store}$, $t^{\widetilde{S}} \vdash \mu_i, \Gamma = FV(t^{\widetilde{S}})$*, and* $\forall k \geq 0, \Gamma \vdash \langle \varepsilon_r \tilde{\ell}_r, \tilde{\ell}_c, \sigma_1, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r, \tilde{\ell}_c, \sigma_2, \mu_2\rangle$*, then* $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_1(t^{\widetilde{S}}), \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_2(t^{\widetilde{S}}), \mu_2\rangle : \mathcal{C}(\widetilde{S})$.

Proof. By induction on the derivation of term $t \in \textsc{Term}_{\widetilde{S}}$. Considering the last step used in the derivation:

Case (*x*). $t^{\widetilde{S}} = x^{\widetilde{S}}$ so $\Gamma = \{x^{\widetilde{S}}\}$. $\Gamma \vdash \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_1, \mu\rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_2, \mu\rangle$ implies by definition that $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_1(x^{\widetilde{S}}), \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_2(x^{\widetilde{S}}), \mu_2\rangle : \widetilde{S}$.

Case (*b*). $t^{\widetilde{S}} = b_{\tilde{\ell}}$. By definition of substitution, $\sigma_1(b_{\tilde{\ell}}) = \sigma_2(b_{\tilde{\ell}}) = b_{\tilde{\ell}}$. By definition, $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} b_{\tilde{\ell}}, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} b_{\tilde{\ell}}, \mu_2\rangle : \mathsf{Bool}_{\tilde{\ell}}$ as required.

Case (*l*). $t^{\widetilde{S}} = l^{\widetilde{S}_1}_{\tilde{\ell}_1}$ where $\widetilde{S} = \mathsf{Ref}_{\tilde{\ell}_1} \widetilde{S}_1$. By definition of substitution, $\sigma_1(l^{\widetilde{S}_1}_{\tilde{\ell}_1}) = \sigma_2(l^{\widetilde{S}_1}_{\tilde{\ell}_1}) = l^{\widetilde{S}_1}_{\tilde{\ell}_1}$. We know that $\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} l^{\widetilde{S}_1}_{\tilde{\ell}_1} \in \textsc{Term}_{\mathsf{Ref}_{\tilde{\ell}_1} \widetilde{S}_1}$. By definition, $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} l^{\widetilde{S}_1}_{\tilde{\ell}_1}, \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} l^{\widetilde{S}_1}_{\tilde{\ell}_1}, \mu_2\rangle : \mathsf{Ref}_{\tilde{\ell}_1} \widetilde{S}_1$ as required.

Case (λ). $t^{\widetilde{S}} = (\lambda^{\tilde{\ell}'_c} x^{\widetilde{S}_1}.t^{\widetilde{S}_2})_{\tilde{\ell}}$. Then $\widetilde{S} = \widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_2$.
By definition of substitution, assuming $x^{\widetilde{S}_1} \notin dom(\sigma_i)$, and Lemma 60:

$$\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_i(t^{\widetilde{S}}) = \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} (\lambda^{\tilde{\ell}'_c} x^{\widetilde{S}_1}.\sigma_i(t^{\widetilde{S}_2}))_{\tilde{\ell}} \in \textsc{Term}_{\widetilde{S}}$$

If $\tilde{\ell} \not\preccurlyeq \ell_o$ or $\tilde{\ell}_r \not\preccurlyeq \ell_o$, the result holds trivially because all function values are related in such cases. Assume $\tilde{\ell} \preccurlyeq \ell_o$ and $\tilde{\ell}_r \preccurlyeq \ell_o$, $j \leq k$, μ'_1, μ'_2 such that $\mu_i \twoheadrightarrow \mu'_i$ and $\langle \tilde{\ell}_r, \mu'_1\rangle \approx^j_{\ell_o} \langle \tilde{\ell}_r, \mu'_2\rangle$, and assume two values v_1 and v_2 such that $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_1, \mu'_1\rangle \approx^j_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_2, \mu'_2\rangle : \widetilde{S}'_1$. Consider $\widetilde{S}' = \widetilde{S}''_1 \xrightarrow{\tilde{\ell}''_c}_{\tilde{\ell}''} \widetilde{S}''_2$ and ε_1 such that $\varepsilon_1 \vdash \widetilde{S}_1 \xrightarrow{\tilde{\ell}'_c}_{\tilde{\ell}} \widetilde{S}_2 \lesssim \widetilde{S}'$ and that $\varepsilon_2 \vdash \widetilde{S}'_1 \lesssim \widetilde{S}''_1$.

For simplicity, let us annotate $\widetilde{S}'_2 = \widetilde{S}''_2 \mathbin{\widetilde{\curlyvee}} \tilde{\ell}''$. We need to show that:

$$\begin{array}{rl} & \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1(\lambda^{\tilde{\ell}'_c} x^{\widetilde{S}_1}.\sigma_1(t^{\widetilde{S}_2}))_{\tilde{\ell}} @^{\widetilde{S}'}_{\varepsilon_\ell} \varepsilon_2 v_1, \mu'_1\rangle \\ \approx^j_{\ell_o} & \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1(\lambda^{\tilde{\ell}'_c} x^{\widetilde{S}_1}.\sigma_2(t^{\widetilde{S}_2}))_{\tilde{\ell}} @^{\widetilde{S}'}_{\varepsilon_\ell} \varepsilon_2 v_2, \mu'_2\rangle \quad : \mathcal{C}(\widetilde{S}'_2) \end{array}$$

Each v_i is either a bare value u_i or a casted value $\varepsilon_{2i} u_i :: \widetilde{S}'_1$. In the latter case, the application expression combines evidence, which may fail with **error**. If it succeeds, we call the combined evidence ε'_{2i}. The application rule then applies: it may fail with **error** if the evidence ε'_{2i} cannot be combined with the evidence for the function parameter. Everytime a failure is produced product of evidence combination, then the relation vacuously holds. We therefore consider the only interesting case, where reductions always succeed. Let us call $e\ell = \varepsilon_r \tilde{\ell}_r$, then:

$$\begin{array}{rl} & \varepsilon_1(\lambda^{\tilde{\ell}'_c} x^{\widetilde{S}_1}.\sigma_i(t^{\widetilde{S}_2}))_{\tilde{\ell}} @^{\widetilde{S}'} \varepsilon'_{2i} u_i \mid \mu'_i \\ \xrightarrow{e\ell}_{\tilde{\ell}_c} & \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}'', \widetilde{S}''_2}_{\varepsilon'_r, \varepsilon_\ell \tilde{\ell}} \varepsilon_p([\varepsilon_{ai} u_i :: \widetilde{S}_1 / x^{\widetilde{S}_1}]\sigma_i(t^{\widetilde{S}_2})) \mid \mu'_i \end{array}$$

where ε_ℓ, ε_p and ε_{ai} are the new evidences for the label, return value and argument, respectively. We then extend the substitutions to map $x^{\widetilde{S}_1}$ to the casted arguments:

$$\sigma'_i = \sigma_i\{x^{\widetilde{S}_1} \mapsto \varepsilon_{ai} u_i :: \widetilde{S}_1\}$$

We know that $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_1, \mu'_1\rangle \approx^j_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_2, \mu'_2\rangle$ and consider $e\ell \overset{\tilde{\ell}_c}{\triangleright} u_i \in \textsc{Term}_{\widetilde{S}_{ui}}$ then $\varepsilon_{ai} \vdash \widetilde{S}_{ui} \lesssim \widetilde{S}_1$ and $\varepsilon_{ai} = (\varepsilon_{2i} \circ^{<:} \varepsilon_2) \circ^{<:} idom(\varepsilon_1)$, therefore using Lemma 65 $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} (\varepsilon_{a1} u_1 :: \widetilde{S}_1), \mu'_1\rangle \approx^j_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} (\varepsilon_{a2} u_2 :: \widetilde{S}_1), \mu'_2\rangle : \widetilde{S}_1$
So as $\mu_i \twoheadrightarrow \mu'_i$ then by Lemma 58, $\Gamma, x^{\widetilde{S}_1} \vdash \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma'_1, \mu'_1\rangle \approx^j_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma'_2, \mu'_2\rangle$. We also know that $\varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}) \overset{\tilde{\ell}'_c}{\triangleright} t^{\widetilde{S}_2} \in \textsc{Term}_{\widetilde{S}_2}$.
But as $\varepsilon'_r \vdash \widetilde{\tilde{\ell}_r \curlyvee \tilde{\ell} \preccurlyeq \tilde{\ell}'_c}$, by Lemma 59 then

$\Gamma, x^{\widetilde{S}_1} \vdash \langle \varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}), \tilde{\ell}'_c, \sigma'_1, \mu'_1\rangle \approx^j_{\ell_o} \langle \varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}) \overset{\tilde{\ell}'_c}{\triangleright} \sigma'_2, \mu'_2\rangle$. Then by induction hypothesis:

$$\langle \varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}) \overset{\tilde{\ell}'_c}{\triangleright} \sigma'_1(t^{\widetilde{S}_2}), \mu'_1\rangle \approx^{j-1}_{\ell_o} \langle \varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}) \overset{\tilde{\ell}'_c}{\triangleright} \sigma'_2(t^{\widetilde{S}_2}), \mu'_2\rangle : \mathcal{C}(\widetilde{S}_2)$$

Finally, by Lemma 71:

$$\begin{aligned} &\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}'', \widetilde{S}''_2}_{\varepsilon'_r, \varepsilon_\ell \tilde{\ell}} \varepsilon_p \sigma'_1(t^{\widetilde{S}_2}), \mu'_1\rangle \\ \approx^j_{\ell_o} \quad &\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}'_c, \tilde{\ell}'', \widetilde{S}''_2}_{\varepsilon'_r, \varepsilon_l \tilde{\ell}} \varepsilon_p \sigma'_2(t^{\widetilde{S}_2}), \mu'_2\rangle : \mathcal{C}(\widetilde{S}'_2) \end{aligned}$$

By backward preservation of the relations (Lemma 61), this implies that:

$$\begin{aligned} &\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 (\lambda^{\tilde{\ell}'_c} x^{\widetilde{S}_1}.\sigma_1(t^{\widetilde{S}_2}))_{\tilde{\ell}} \mathbin{@}^{\widetilde{S}'} \varepsilon_2 v_1, \mu_1\rangle \\ \approx^j_{\ell_o} \quad &\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 (\lambda^{\tilde{\ell}'_c} x^{\widetilde{S}_1}.\sigma_2(t^{\widetilde{S}_2}))_{\tilde{\ell}} \mathbin{@}^{\widetilde{S}'} \varepsilon_2 v_2, \mu_2\rangle \quad : \mathcal{C}(\widetilde{S}'_2) \end{aligned}$$

Case (!). $t^{\widetilde{S}} = !^{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1} \varepsilon t^{\widetilde{S}'_1}$. Then $\widetilde{S} = \widetilde{S}_1 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}$.
By definition of substitution:

$$\sigma_i(t^{\widetilde{S}}) = !^{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1} \varepsilon \sigma_i(t^{\widetilde{S}'_1})$$

We have to show that

$$\begin{aligned} &\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} !^{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1} \varepsilon \sigma_i(t^{\widetilde{S}'_1}), \mu_1\rangle \\ \approx^k_{\ell_o} \quad &\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} !^{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1} \varepsilon \sigma_i(t^{\widetilde{S}'_1}), \mu_2\rangle : \mathcal{C}(\widetilde{S}_1 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}) \end{aligned}$$

By Lemma 60:

$$\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} !^{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1} \varepsilon \sigma_i(t^{\widetilde{S}'_1}) \in \textsc{Term}_{\widetilde{S}_1 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}}$$

By induction hypotheses:

$$\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_1(t^{\widetilde{S}'_1}), \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_2(t^{\widetilde{S}'_1}), \mu_2\rangle : \mathcal{C}(\widetilde{S}'_1)$$

Consider $j < k$, and $e\ell = \varepsilon_r \tilde{\ell}_r$, then by definition of related computations

$$\begin{aligned} &\sigma_i(t^{\widetilde{S}'_1}) \mid \mu_i \overset{e\ell}{\longmapsto}{}^{j}_{\tilde{\ell}_c} t_i^{\widetilde{S}'_1} \mid \mu'_i \wedge \\ &\qquad \langle \tilde{\ell}_r, \mu'_1\rangle \approx^{k-j}_{\ell_o} \langle \tilde{\ell}_r, \mu'_2\rangle \wedge \mu_i \twoheadrightarrow \mu'_i \\ &\wedge \mathit{irred}(t_i^{\widetilde{S}'_1}) \Rightarrow \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t_1^{\widetilde{S}'_1}, \mu'_1\rangle \approx^{k-j}_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t_2^{\widetilde{S}'_1}, \mu'_2\rangle : \widetilde{S}'_1 \end{aligned}$$

Where $\widetilde{S}'_1 = \mathsf{Ref}_{\tilde{\ell}'}\ \widetilde{S}''_1$. If terms $t_i^{\widetilde{S}'_1}$ are reducible after $j = k-1$ steps, then $!^{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1} \varepsilon \sigma_i(t^{\widetilde{S}'_1}) \mid \mu_i \overset{e\ell}{\longmapsto}{}^{j}_{\tilde{\ell}_c} !^{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1} \varepsilon t_i'^{\widetilde{S}'_1} \mid \mu'_i$ and the result holds.
If after at most j steps $t_i^{\widetilde{S}'_1}$ is irreducible it means that for some $j' \leq j$, $!^{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1} \varepsilon \sigma_i(t^{\widetilde{S}'_1}) \mid \mu_i \overset{e\ell}{\longmapsto}{}^{j'}_{\tilde{\ell}_c} !^{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1} \varepsilon v_i \mid \mu'_i$. If $j' = j$ then we use the same same argument for reducible terms and the result holds.
Let us consider now $j' < j$. Then $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_1, \mu'_1\rangle \approx^{k-j'}_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_2, \mu'_2\rangle : \widetilde{S}'_1$. By Lemma 37, each v_i is either a location $(l_{i\,\tilde{\ell}'}^{\widetilde{S}''_1})$ or a casted location $\varepsilon_i(l_{i\,\tilde{\ell}''}^{\widetilde{S}'''_1}) :: \widetilde{S}'_1$. In case a value v_{ij} is a casted value, then the whole term $\sigma_i(t^{\widetilde{S}})$ can take a step by (Rg), combining ε with ε_i. Such a step either fails, or succeeds with a new combined evidence. Therefore, either:

$$\sigma_i(t^{\widetilde{S}}) \mid \mu_i \overset{e\ell}{\longmapsto}{}^{j'}_{\tilde{\ell}_c} \textbf{error}$$

in which case we do not care since we only consider termination-insensitive noninterference, or:

$$\begin{aligned} \sigma_i(t^{\widetilde{S}}) \mid \mu \quad &\overset{e\ell}{\longmapsto}{}^{j'+1}_{\tilde{\ell}_c} \quad !^{\mathsf{Ref}_{\tilde{\ell}}\ \widetilde{S}_1} \varepsilon'_i l_{i\,\tilde{\ell}''}^{\widetilde{S}'''_1} \mid \mu'_i \\ &\overset{e\ell}{\longmapsto}{}^{1}_{\tilde{\ell}_c} \quad \mathsf{prot}^{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}, \tilde{\ell}, \widetilde{S}}_{(\varepsilon_r \widetilde{\Upsilon} \mathit{ilbl}(\varepsilon'_i)), \mathit{ilbl}(\varepsilon'_i) \tilde{\ell}''} \mathit{iref}(\varepsilon'_i) v'_i \end{aligned}$$

with $v'_i = \mu'_i(l_{i\,\tilde{\ell}''}^{\widetilde{S}'''_1})$. As $\langle \tilde{\ell}_r, \mu'_1\rangle \approx^{k-j}_{\ell_o} \langle \tilde{\ell}_r, \mu'_2\rangle$ then $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v'_1, \mu'_1\rangle \approx^{k-j'}_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v'_2, \mu'_2\rangle : \widetilde{S}'''$. As $(\varepsilon_r \mathbin{\widetilde{\Upsilon}} \mathit{ilbl}(\varepsilon'_i)) \vdash \tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}' \preccurlyeq \tilde{\ell}_c \mathbin{\widetilde{\Upsilon}} \tilde{\ell}$, by Lemma 64.1, $\langle \varepsilon_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}) \overset{\tilde{\ell}_c}{\triangleright} v'_1, \mu'_1\rangle \approx^{k-j'}_{\ell_o} \langle \varepsilon_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}) \overset{\tilde{\ell}_c}{\triangleright} v'_2, \mu'_2\rangle : \widetilde{S}'''$. By Lemma 70 then

$$\begin{aligned} &\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}, \tilde{\ell}, \widetilde{S}}_{(\varepsilon_r \widetilde{\Upsilon} \mathit{ilbl}(\varepsilon'_1)), \mathit{ilbl}(\varepsilon'_1) \tilde{\ell}''} \mathit{iref}(\varepsilon'_1) v'_1, \mu'_1\rangle \\ \approx^{k-j'}_{\ell_o} \quad &\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}, \tilde{\ell}, \widetilde{S}}_{(\varepsilon_r \widetilde{\Upsilon} \mathit{ilbl}(\varepsilon'_1)), \mathit{ilbl}(\varepsilon'_2) \tilde{\ell}''} \mathit{iref}(\varepsilon'_1) v'_2, \mu'_2\rangle : \widetilde{S}''' \end{aligned}$$

Finally, by backward preservation of the relations (Lemma 61) the result holds.

Case (:=). $t^{\widetilde{S}} = \varepsilon_1 t_1^{\widetilde{S}_1} :=^{\varepsilon_\ell} \varepsilon_2 t_2^{\widetilde{S}_2}$. Then $\widetilde{S} = \mathsf{Unit}_\perp$.
By definition of substitution:

$$\sigma_i(t^{\widetilde{S}}) = \varepsilon_1 \sigma_i(t^{\widetilde{S}_1}) :=^{\varepsilon_\ell} \varepsilon_2 \sigma_i(t^{\widetilde{S}_2})$$

and Lemma 60:

$$\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 \sigma_i(t^{\widetilde{S}_1}) :=^{\varepsilon_\ell} \varepsilon_2 \sigma_i(t^{\widetilde{S}_2}) \in \textsc{Term}_{\mathsf{Unit}_\perp}$$

We have to show that

$$\begin{aligned} &\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 \sigma_1(t^{\widetilde{S}_1}) :=^{\varepsilon_\ell} \varepsilon_2 \sigma_1(t^{\widetilde{S}_2}), \mu_1\rangle \\ \approx^k_{\ell_o} \quad &\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 \sigma_2(t^{\widetilde{S}_1}) :=^{\varepsilon_\ell} \varepsilon_2 \sigma_2(t^{\widetilde{S}_2}), \mu_2\rangle : \mathcal{C}(\widetilde{S}) \end{aligned}$$

By induction hypotheses

$$\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_1(t^{\widetilde{S}_1}), \mu_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_2(t^{\widetilde{S}_1}), \mu_2\rangle : \mathcal{C}(\widetilde{S}_1)$$

Suppose $j_1 < k$, $e\ell = \varepsilon_r \tilde{\ell}_r$, and that $\sigma_i(t^{\widetilde{S}_1})$ are irreducible after j_1 steps (otherwise the result holds immediately). Then by definition of related computations:

$$\begin{aligned} &\sigma_i(t^{\widetilde{S}_1}) \mid \mu_i \overset{e\ell}{\longmapsto}{}^{j_1}_{\tilde{\ell}_c} v_i \mid \mu'_i \wedge \\ &\qquad \mu_i \twoheadrightarrow \mu'_i \wedge \langle \tilde{\ell}_r, \mu'_1\rangle \approx^{k-j_1}_{\ell_o} \langle \tilde{\ell}_r, \mu'_2\rangle \\ &\qquad \Longrightarrow \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_1, \mu'_1\rangle \approx^{k-j_1}_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_2, \mu'_2\rangle : \widetilde{S}_1 \end{aligned}$$

As $\mu_i \twoheadrightarrow \mu'_i$ and $\langle \tilde{\ell}_r, \mu'_1\rangle \approx^{k-j_1}_{\ell_o} \langle \tilde{\ell}_r, \mu'_2\rangle$ then by Lemma 58, $\langle \varepsilon_r \tilde{\ell}_r, \tilde{\ell}_c, \sigma_1, \mu'_1\rangle \approx^{k-j_1}_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r, \tilde{\ell}_c, \sigma_2, \mu'_2\rangle$. By induction hypotheses:

$$\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_1(t^{\widetilde{S}_2}), \mu'_1\rangle \approx^k_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_2(t^{\widetilde{S}_2}), \mu'_2\rangle : \mathcal{C}(\widetilde{S}_2)$$

Again, consider $j_2 = k - j_1$, if after j_2 steps $\sigma_1(t^{\widetilde{S}_2})$ is reducible or is a value, the result holds immediately. The interest case if after $j'_2 < j_2$ steps $\sigma_1(t^{\widetilde{S}_2})$ reduces to values v'_i:

$$\begin{aligned} &\sigma_i(t^{\widetilde{S}_2}) \mid \mu'_i \overset{e\ell}{\longmapsto}{}^{j'_2}_{\tilde{\ell}_c} v'_i \mid \mu''_i \wedge \mu'_i \twoheadrightarrow \mu''_i \wedge \\ &\qquad \langle \tilde{\ell}_r, \mu''_1\rangle \approx^{k-j_1-j'_2}_{\ell_o} \langle \tilde{\ell}_r, \mu''_2\rangle \\ &\Longrightarrow \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v'_1, \mu''_1\rangle \approx^{k-j_1-j'_2}_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v'_2, \mu''_2\rangle : \widetilde{S}_2 \end{aligned}$$

Then

$$\sigma_i(t^{\widetilde{S}}) \mid \mu_i \xrightarrow{e\ell}{}^{j_1+j_2'}_{\tilde{\ell}_c} \varepsilon_1 v_i :\overset{\varepsilon_\ell}{=}{}^{\tilde{\ell}_c} \varepsilon_2 v_i' \mid \mu_i'' \wedge \mu_i \twoheadrightarrow \mu_i'' \wedge \langle \tilde{\ell}_r, \mu_1'' \rangle \approx_{\ell_o}^{k-j_1-j_2'} \langle \tilde{\ell}_r, \mu_2'' \rangle$$

Now v_i and v_i' can be bare values or casted values. In the case of casted values we can combine evidence, which may fail with **error**. We assume that all evidence combinations succed, otherwise the relation vacuously holds. As both values v_i are related at some reference type, then by canonical forms (Lemma 37) they both must be identical locations $l^{\widetilde{S}_1'}$ for some $\widetilde{S}_1' \lesssim \widetilde{S}_1$. :

$$\begin{array}{rl} & \varepsilon_1 v_i :\overset{\varepsilon_\ell}{=} \varepsilon_2 v_i' \mid \mu_i'' \\ \xrightarrow{e\ell}{}^{2}_{\tilde{\ell}_c} & \varepsilon_{1i}' l^{\widetilde{S}_1'}_{\tilde{\ell}} :\overset{\varepsilon_\ell}{=} \varepsilon_{2i}' u_i' \mid \mu_i'' \\ \xrightarrow{e\ell}{}^{1}_{\tilde{\ell}_c} & \mathsf{unit}_\perp \mid \mu_i''' \end{array}$$

Where $\mu_i''' = \mu_i''[l^{\widetilde{S}_1'} \mapsto \varepsilon_{1i}''(u_i' \mathbin{\widetilde{\Upsilon}} (\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell})) :: \widetilde{S}_1']$. As $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_1', \mu_1'' \rangle \approx_{\ell_o}^{k-j_1-j_2'} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_2', \mu_2'' \rangle : \widetilde{S}_2$ then by Lemma 65, $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_{21}' u_1' :: \widetilde{S}_1', \mu_1'' \rangle \approx_{\ell_o}^{k-j_1-j_2'} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_{22}' u_2' :: \widetilde{S}_1', \mu_1'' \rangle : \widetilde{S}_1'$. As $\varepsilon' \vdash \tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell} \preccurlyeq \mathit{label}(\widetilde{S}_1')$ and $\varepsilon_{1i}'' = \varepsilon_{2i}'' \mathbin{\widetilde{\Upsilon}} \varepsilon'$, by Lemma 72,

$$\begin{array}{rl} & \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_{11}''(u_1' \mathbin{\widetilde{\Upsilon}} (\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell})) :: \widetilde{S}_1', \mu_1'' \rangle \\ \approx_{\ell_o}^{k-j_1-j_2'} & \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_{12}''(u_2' \mathbin{\widetilde{\Upsilon}} (\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell})) :: \widetilde{S}_1', \mu_1'' \rangle \end{array}$$

As every values are related at type Unit, we only have to prove that $\langle \tilde{\ell}_r, \mu_1''' \rangle \approx_{\ell_o}^{k-j_1-j_2'-3} \langle \tilde{\ell}_r, \mu_1''' \rangle$, but using monotonicity (Lemma 66), it is trivial to prove that because both stores update the same location $l^{\widetilde{S}_1'}$ to values that are related and then the result holds.

Case (ref). $t^{\widetilde{S}} = \mathsf{ref}^{\widetilde{S}_1}_{\varepsilon_\ell} \varepsilon t^{\widetilde{S}_1'}$. Then $\widetilde{S} = \mathsf{Ref}_\perp \widetilde{S}_1$.
By definition of substitution:

$$\sigma_i(t^{\widetilde{S}}) = \mathsf{ref}^{\widetilde{S}_1}_{\varepsilon_\ell} \varepsilon \sigma_i(t^{\widetilde{S}_1'})$$

and Lemma 60:

$$\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{ref}^{\widetilde{S}_1}_{\varepsilon_\ell} \varepsilon \sigma_i(t^{\widetilde{S}_1'}) \in \textsc{Term}_{\mathsf{Ref}_\perp \widetilde{S}_1}$$

We have to show that

$$\begin{array}{rl} & \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{ref}^{\widetilde{S}_1}_{\varepsilon_\ell} \varepsilon \sigma_1(t^{\widetilde{S}_1'}), \mu_1 \rangle \\ \approx_{\ell_o}^{k} & \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{ref}^{\widetilde{S}_1}_{\varepsilon_\ell} \varepsilon \sigma_2(t^{\widetilde{S}_1'}), \mu_2 \rangle : \mathcal{C}(\widetilde{S}_1 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}) \end{array}$$

By induction hypotheses:

$$\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_1(t^{\widetilde{S}_1'}), \mu \rangle \approx_{\ell_o}^{k} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \sigma_2(t^{\widetilde{S}_1'}), \mu \rangle : \mathcal{C}(\widetilde{S}_1')$$

Consider $j < k$ and $e\ell = \varepsilon_r \tilde{\ell}_r$, by definition of related computations

$$\begin{array}{l} \sigma_i(t^{\widetilde{S}_1'}) \mid \mu_i \xrightarrow{e\ell}{}^{j}_{\tilde{\ell}_c} t_i^{\widetilde{S}_1'} \mid \mu_i' \wedge \\ \qquad \langle \tilde{\ell}_r, \mu_1' \rangle \approx_{\ell_o}^{k-j} \langle \tilde{\ell}_r, \mu_2' \rangle \wedge \mu_i \twoheadrightarrow \mu_i' \\ \wedge \mathit{irred}(t_i^{\widetilde{S}_1'}) \Rightarrow \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t_1^{\widetilde{S}_1'}, \mu_1' \rangle \approx_{\ell_o}^{k-j} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t_2^{\widetilde{S}_1'}, \mu_2' \rangle : \widetilde{S}_1' \end{array}$$

If terms $t_i^{\widetilde{S}_1'}$ are reducible after $j = k-1$ steps, then $\mathsf{ref}^{\widetilde{S}_1}_{\varepsilon_\ell} \varepsilon \sigma_i(t^{\widetilde{S}_1'}) \mid \mu_i \xrightarrow{e\ell}{}^{j}_{\tilde{\ell}_c} \mathsf{ref}^{\widetilde{S}_1}_{\varepsilon_\ell} \varepsilon t_i'^{\widetilde{S}_1'} \mid \mu_i'$ and the result holds.

If after at most j steps $t_i^{\widetilde{S}_1'}$ is irreducible, it means that for some $j' \leq j$ $\mathsf{ref}^{\widetilde{S}_1}_{\varepsilon_\ell} \varepsilon \sigma_i(t^{\widetilde{S}_1'}) \mid \mu_i \xrightarrow{e\ell}{}^{j'}_{\tilde{\ell}_c} \mathsf{ref}^{\widetilde{S}_1}_{\varepsilon_\ell} \varepsilon v_i \mid \mu_i'$. 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 37, each v_i is either a base value u_i or a casted base value $\varepsilon_i u_i :: \widetilde{S}_1'$. In case a value v_{ij} is a casted value, then the whole term $\sigma_i(t^{\widetilde{S}})$ can take a step by (Rg), combining ε with ε_i. Such a step either fails, or succeeds with a new combined evidence. Therefore, either:

$$\sigma_i(t^{\widetilde{S}}) \mid \mu_i \xrightarrow{e\ell}{}^{j'}_{\tilde{\ell}_c} \textbf{error}$$

in which case we do not care since we only consider termination-insensitive noninterference, or:

$$\begin{array}{lll} \sigma_i(t^{\widetilde{S}}) \mid \mu & \xrightarrow{e\ell}{}^{j'+1}_{\tilde{\ell}_c} & \mathsf{ref}^{\widetilde{S}_1}_{\varepsilon_\ell} \varepsilon_i' u_i \mid \mu_i' \\ & \xrightarrow{e\ell}{}^{1}_{\tilde{\ell}_c} & l^{\widetilde{S}_1}_{\perp} \mid \mu_i'' \end{array}$$

with, $\mu_i'' = \mu_i'[l^{\widetilde{S}_1} \mapsto \varepsilon_i'(u_i \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_r) :: \widetilde{S}_1]$. Where $\varepsilon_i'' = \varepsilon_i' \mathbin{\widetilde{\Upsilon}} (\varepsilon_r \circ^{\preccurlyeq} \varepsilon_\ell)$. We know that if $u_i \in \textsc{Term}_{\widetilde{S}_i}$, then $\varepsilon_i \vdash \widetilde{S}_i \lesssim \widetilde{S}_1$. Also, as $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_1, \mu_1' \rangle \approx_{\ell_o}^{k-j'} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_2, \mu_2' \rangle : \widetilde{S}_1'$ then by Lemma 65, $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1' u_1 :: \widetilde{S}_1, \mu_1' \rangle \approx_{\ell_o}^{k-j'} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_2' u_2 :: \widetilde{S}_1, \mu_2' \rangle : \widetilde{S}_1'$ and as $(\varepsilon_r \circ^{\preccurlyeq} \varepsilon_\ell) \vdash \tilde{\ell}_r \mathbin{\widetilde{\preccurlyeq}} \mathit{label}(\widetilde{S}_1)$, then by Lemma 72 and Lemma 66, $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_2''(u_i \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_r) :: \widetilde{S}_1, \mu_1' \rangle \approx_{\ell_o}^{k-j'-2} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_2''(u_2 \mathbin{\widetilde{\Upsilon}} \tilde{\ell}_r) :: \widetilde{S}_1, \mu_2' \rangle : \widetilde{S}_1'$

By definition of related stores $\langle \tilde{\ell}_r, \mu_1'' \rangle \approx_{\ell_o}^{k-j'} \langle \tilde{\ell}_r, \mu_2'' \rangle$. Then by Monotonicity of the relation (Lemma 66) $\langle \tilde{\ell}_r, \mu_1'' \rangle \approx_{\ell_o}^{k-j'-2} \langle \tilde{\ell}_r, \mu_2'' \rangle$ and the result holds.

Case ($\oplus$). $t^{\widetilde{S}} = \varepsilon_1 t^{\widetilde{S}_1} \oplus^{\tilde{\ell}} \varepsilon_2 t^{\widetilde{S}_2}$

By definition of substitution:

$$\sigma_i(t^{\widetilde{S}}) = \varepsilon_1 \sigma_i(t^{\widetilde{S}_1}) \oplus^{\tilde{\ell}} \varepsilon_2 \sigma_i(t^{\widetilde{S}_2})$$

and Lemma 60:

$$\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 \sigma_i(t^{\widetilde{S}_1}) \oplus^{\tilde{\ell}} \varepsilon_2 \sigma_i(t^{\widetilde{S}_2}) \in \textsc{Term}_{\widetilde{S}}$$

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 $j_1 + j_2 < k - 3$ and $e\ell = \varepsilon_r \tilde{\ell}_r$ where:

$$\begin{array}{l} \sigma_i(t^{\widetilde{S}_1}) \mid \mu_i \xrightarrow{e\ell}{}^{j_1}_{\tilde{\ell}_c} v_{i1} \mid \mu_i' \wedge \\ \qquad \langle \tilde{\ell}_r, \mu_1' \rangle \approx_{\ell_o}^{k-j_1} \langle \tilde{\ell}_r, \mu_2' \rangle \wedge \mu_i \twoheadrightarrow \mu_i' \Rightarrow \\ \qquad\qquad \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{11}, \mu_1' \rangle \approx_{\ell_o}^{k-j_1} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{21}, \mu_2' \rangle : \widetilde{S}_1 \end{array}$$

$$\begin{array}{l} \sigma_i(t^{\widetilde{S}_2}) \mid \mu_i' \xrightarrow{e\ell}{}^{j_2}_{\tilde{\ell}_c} v_{i2} \mid \mu_i'' \wedge \\ \qquad \langle \tilde{\ell}_r, \mu_1'' \rangle \approx_{\ell_o}^{k-j_1-j_2} \langle \tilde{\ell}_r, \mu_2'' \rangle \wedge \mu_i' \twoheadrightarrow \mu_i'' \Rightarrow \\ \qquad \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{12}, \mu_1'' \rangle \approx_{\ell_o}^{k-j_1-j_2} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{22}, \mu_2'' \rangle : \widetilde{S}_1 \end{array}$$

By Lemma 37, each v_{ij} is either a boolean $(b_{ij})_{\tilde{\ell}_{ij}}$ or a casted boolean $\varepsilon_{ij}(b_{ij})_{\tilde{\ell}'_{ij}} :: \widetilde{S}_j$. In case a value v_{ij} is a casted value, then the whole term $\sigma_i(t^{\widetilde{S}})$ 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:

$$\sigma_i(t^{\widetilde{S}}) \mid \mu_i \xrightarrow{e\ell}{}^{j_1+j_2}_{\tilde{\ell}_c} \textbf{error}$$

in which case we do not care since we only consider termination-insensitive noninterference, or:

$$\begin{array}{rl} & \sigma_i(t^{\widetilde{S}}) \mid \mu_i'' \\ \longmapsto^{j_1+j_2+2} & \varepsilon'_{i1}(b_{i1})_{\tilde{\ell}'_{i1}} \oplus^{\tilde{\ell}} \varepsilon'_{i2}(b_{i2})_{\tilde{\ell}'_{i2}} \mid \mu_i'' \\ \longmapsto^{1} & \varepsilon'_i(b_i)_{\tilde{\ell}'_i} :: \mathsf{Bool}_{\tilde{\ell}} \mid \mu_i'' \end{array}$$

with $b_i = b_{i1}[\![\oplus]\!]b_{i2}$ and $\tilde{\ell}'_i = \tilde{\ell}'_{i1} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}'_{i2}$. It remains to show that:

$$\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} (b_1)_{\tilde{\ell}'_1}, \mu_1'' \rangle \approx^{k-j_1-j_2-3}_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} (b_2)_{\tilde{\ell}'_2}, \mu_2'' \rangle : \mathsf{Bool}_{\tilde{\ell}}$$

If $\tilde{\ell} \not\widetilde{\preccurlyeq} \ell_o$, then the result holds trivially because all boolean values are related. If $\exists \varepsilon_2,\ \varepsilon_2 \vdash \tilde{\ell} \preccurlyeq \ell_o$ but $\varepsilon_{ij} \circ^{\preccurlyeq} \varepsilon_2$ is not defined, by Lemma 55, $(\varepsilon_1 \circ^{\preccurlyeq} \varepsilon_{ij}) \circ^{\preccurlyeq} \varepsilon_2$ is also not defined. As $\varepsilon'_{ij} = (\varepsilon_1 \circ^{\preccurlyeq} \varepsilon_{ij})$ and as the join is monotone, $(\varepsilon'_{i1} \mathbin{\widetilde{\Upsilon}} \varepsilon'_{ij}) \circ^{\preccurlyeq} \varepsilon_2$ is also not defined, and the values are not observables. If the value is observable, then if $\tilde{\ell} \mathbin{\widetilde{\preccurlyeq}} \ell_o$, then also $\tilde{\ell}'_i \mathbin{\widetilde{\preccurlyeq}} \ell_o$ (if the transitivity does not hold the values are not observable), which means by definition of $\approx_{\ell_o}$ on boolean values, that $b_{11} = b_{21}$ and $b_{12} = b_{22}$, so $b_1 = b_2$.

Case (app). $t^{\widetilde{S}} = \varepsilon_1 t^{\widetilde{S}_1} \mathbin{@}^{\widetilde{S}_{11} \xrightarrow{\tilde{\ell}'_c} \tilde{\ell} \widetilde{S}_{12}}_{\varepsilon_\ell} \varepsilon_2 t^{\widetilde{S}_2}$
with $\varepsilon_1 \vdash \widetilde{S}_1 \lesssim S_{11} \to_{\tilde{\ell}} S_{12}$, $\varepsilon_2 \vdash \widetilde{S}_2 \lesssim \widetilde{S}_{11}$, and $\widetilde{S} = \widetilde{S}_{12} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}$.
We omit the $\mathbin{@}^{\widetilde{S}_{11} \xrightarrow{\tilde{\ell}'_c} \tilde{\ell} \widetilde{S}_{12}}_{\varepsilon_\ell}$ operator in applications below.

By definition of substitution:

$$\sigma_i(t^{\widetilde{S}}) = \varepsilon_1 \sigma_i(t^{\widetilde{S}_1})\ \varepsilon_2 \sigma_i(t^{\widetilde{S}_2})$$

and Lemma 60:

$$\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon_1 \sigma_i(t^{\widetilde{S}_1})\ \varepsilon_2 \sigma_i(t^{\widetilde{S}_2}) \in \textsc{Term}_{\widetilde{S}}$$

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

$$\begin{array}{l} \sigma_i(t^{\widetilde{S}_1}) \mid \mu_i \xrightarrow{e\ell}{}^{j_1}_{\tilde{\ell}_c} v_{i1} \mid \mu_i' \Rightarrow \wedge \\ \qquad \langle \tilde{\ell}_r, \mu_1' \rangle \approx^{k-j_1}_{\ell_o} \langle \tilde{\ell}_r, \mu_2' \rangle \wedge \mu_i \rightarrowtriangle \mu_i' \\ \qquad\quad \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{11}, \mu_1' \rangle \approx^{k-j_1}_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{21}, \mu_2' \rangle : \widetilde{S}_1 \end{array}$$

$$\begin{array}{l} \sigma_i(t^{\widetilde{S}_2}) \mid \mu_i' \xrightarrow{e\ell}{}^{j_2}_{\tilde{\ell}_c} v_{i2} \mid \mu_i'' \Rightarrow \wedge \\ \qquad \langle \tilde{\ell}_r, \mu_1'' \rangle \approx^{k-j_1-j_2}_{\ell_o} \langle \tilde{\ell}_r, \mu_2'' \rangle \wedge \mu_i' \rightarrowtriangle \mu_i'' \\ \qquad\quad \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{12}, \mu_1'' \rangle \approx^{k-j_1-j_2}_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{22}, \mu_2'' \rangle : \widetilde{S}_2 \end{array}$$

Then

$$\sigma_i(t^{\widetilde{S}}) \mid \mu_i \xrightarrow{e\ell}{}^{j_1+j_2}_{\tilde{\ell}_c} \varepsilon_1 v_{11}\ \varepsilon_2 v_{12} \mid \mu_i''$$

By definition of $\approx_{\ell_o}$ at values of function type, using ε_1 and ε_2 to justify the subtyping relations, we have:

$$\begin{array}{rl} & \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} (\varepsilon_1 v_{11}\ \varepsilon_2 v_{12}), \mu_1'' \rangle \\ \approx^{k-j_1-j_2}_{\ell_o} & \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} (\varepsilon_1 v_{21}\ \varepsilon_2 v_{22}), \mu_2'' \rangle : \mathcal{C}(\widetilde{S}_{12} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}) \end{array}$$

Finally, by backward preservation of the relations (Lemma 61) the result holds.

Case (if). $t^{\widetilde{S}} = \mathsf{if}\ {}^{\tilde{\ell}}\varepsilon_1 t^{\widetilde{S}_1}\ \mathsf{then}\ \varepsilon_2 t^{\widetilde{S}_2}\ \mathsf{else}\ \varepsilon_3 t^{\widetilde{S}_3}$, with $\tilde{\ell}' = \mathit{label}(\widetilde{S}_1), \varepsilon'_r = (\varepsilon_r \mathbin{\widetilde{\Upsilon}} \mathit{ilbl}(\varepsilon_1)), \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} t^{\widetilde{S}_1} \in \textsc{Term}_{\widetilde{S}_1}$, $\varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}') \overset{(\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell})}{\triangleright} t^{\widetilde{S}_2} \in \textsc{Term}_{\widetilde{S}_2}$, $\varepsilon'_r(\tilde{\ell}_r \mathbin{\widetilde{\Upsilon}} \tilde{\ell}') \overset{(\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell})}{\triangleright} t^{\widetilde{S}_3} \in \textsc{Term}_{\widetilde{S}_3}$
$\varepsilon_1 \vdash \widetilde{S}_1 \lesssim \mathsf{Bool}_{\tilde{\ell}}$ and $\widetilde{S} = (\widetilde{S}_2 \mathbin{\widetilde{\vee}} \widetilde{S}_3) \mathbin{\widetilde{\Upsilon}} \tilde{\ell}$
By definition of substitution:

$$\sigma_i(t^{\widetilde{S}}) = \mathsf{if}\ {}^{\tilde{\ell}}\varepsilon_1 \sigma_i(t^{\widetilde{S}_1})\ \mathsf{then}\ \varepsilon_2 \sigma_i(t^{\widetilde{S}_2})\ \mathsf{else}\ \varepsilon_3 \sigma_i(t^{\widetilde{S}_3})$$

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:

$$\begin{array}{l} \sigma_i(t^{\widetilde{S}_1}) \mid \mu_i \xrightarrow{\varepsilon_r \tilde{\ell}_r}{}^{j_1}_{\tilde{\ell}_c} v_{i1} \mid \mu_i' \wedge \\ \qquad \langle \tilde{\ell}_r, \mu_1' \rangle \approx^{k-j_1}_{\ell_o} \langle \tilde{\ell}_r, \mu_2' \rangle \wedge \mu_i \rightarrowtriangle \mu_i' \\ \qquad\quad \Rightarrow \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{11}, \mu_1' \rangle \approx^{k-j_1}_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{21}, \mu_2' \rangle : \widetilde{S}_1 \end{array}$$

By Lemma 37, each v_{i1} is either a boolean $(b_{i1})_{\tilde{\ell}_{i1}}$ or a casted boolean $\varepsilon_{i1}(b_{i1})_{\tilde{\ell}'_{i1}} :: \widetilde{S}_1$. In either case, $\widetilde{S}_1 \lesssim \mathsf{Bool}_{\tilde{\ell}_1}$ implies $\widetilde{S}_1 = \mathsf{Bool}_{\tilde{\ell}'_1}$. In case a value v_{i1} is a casted value, then the whole term $\sigma_i(t^{\widetilde{S}})$ can take a step by (Rg), combining ε_i with ε_{i1}. Such a step either fails, or succeeds with a new combined evidence. Therefore, either:

$$\sigma_i(t^{\widetilde{S}}) \mid \mu_i \xrightarrow{e\ell}{}^{j_1+1}_{\tilde{\ell}_c} \textbf{error}$$

in which case we do not care since we only consider termination-insensitive noninterference, or:

$$\begin{array}{l} \sigma_i(t^{\widetilde{S}}) \mid \mu_i \xrightarrow{\varepsilon_r \tilde{\ell}_r}{}^{j_1+1}_{\tilde{\ell}_c} \\ \qquad \mathsf{if}\ {}^{\tilde{\ell}}\varepsilon'_{i1}(b_{i1})_{\tilde{\ell}'_{i1}}\ \mathsf{then}\ \varepsilon_2 \sigma_i(t^{\widetilde{S}_2})\ \mathsf{else}\ \varepsilon_3 \sigma_i(t^{\widetilde{S}_3}) \mid \mu_i' \end{array}$$

We proceed by analyzing two cases depending if $\mathsf{obs}_{\ell_o}((b_{i1})_{\tilde{\ell}'_1})$ holds or not.

Consider that $\mathsf{obs}_{\ell_o}(\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{i1})$ does not hold. Then we do not know which branch each program is going to reduce to. Let us assume to worst scenario and every substitution reduce to a different branch of the conditional (the other cases are similar):

$$\sigma_1(t^{\widetilde{S}}) \mid \mu_1 \xrightarrow{\varepsilon_r \tilde{\ell}_r}{}^{j_1+2}_{\tilde{\ell}_c} \mathsf{prot}^{(\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}), \tilde{\ell}, \widetilde{S}}_{(\varepsilon_r \widetilde{\Upsilon} \mathit{ilbl}(\varepsilon'_{11})), \mathit{ilbl}(\varepsilon'_{11}) \tilde{\ell}'_{11}} \varepsilon_2 \sigma_1(t^{\widetilde{S}_2}) \mid \mu_1'$$

$$\sigma_2(t^{\widetilde{S}}) \mid \mu_2 \xrightarrow{\varepsilon_r \tilde{\ell}_r}{}^{j_1+2}_{\tilde{\ell}_c} \mathsf{prot}^{(\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}), \tilde{\ell}, \widetilde{S}}_{(\varepsilon_r \widetilde{\Upsilon} \mathit{ilbl}(\varepsilon'_{21})), \mathit{ilbl}(\varepsilon'_{21}) \tilde{\ell}'_{21}} \varepsilon_3 \sigma_2(t^{\widetilde{S}_3}) \mid \mu_2'$$

Now if after at least j_2 more steps both branches reduce to a value (let us assume no cast errors are produced, otherwise the lemma vacuously holds):

$$\sigma_i(t^{\widetilde{S}_{i'}}) \mid \mu_i' \xrightarrow{(\varepsilon_r \widetilde{\Upsilon} \mathit{ilbl}(\varepsilon'_{i1}))(\tilde{\ell}_r \widetilde{\Upsilon} \tilde{\ell}')}{}^{j_2}_{\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}} \varepsilon'_{i'} v_{i'} \mid \mu_i''$$

where $i \in \{1, 2\}$ and $i' = i + 1$. Therefore:

$$\begin{array}{rl} & \mathsf{prot}^{(\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}), \tilde{\ell}, \widetilde{S}}_{(\varepsilon_r \widetilde{\Upsilon} \mathit{ilbl}(\varepsilon'_{i1})), \mathit{ilbl}(\varepsilon'_{i1}) \tilde{\ell}'_{i1}} \varepsilon_{i'} \sigma_i(t^{\widetilde{S}_{i'}}) \mid \mu_i' \\ \xrightarrow{\varepsilon_r \tilde{\ell}_r}{}^{j_2}_{\tilde{\ell}_c} & \mathsf{prot}^{(\tilde{\ell}_c \widetilde{\Upsilon} \tilde{\ell}), \tilde{\ell}, \widetilde{S}}_{(\varepsilon_r \widetilde{\Upsilon} \mathit{ilbl}(\varepsilon'_{i1})), \mathit{ilbl}(\varepsilon'_{i1}) \tilde{\ell}'_{i1}} \varepsilon'_{i'} u_{i'} \mid \mu_i'' \\ \xrightarrow{\varepsilon_r \tilde{\ell}_r}{}^{1}_{\tilde{\ell}_c} & (\varepsilon'_{i'} \mathbin{\widetilde{\Upsilon}} \mathit{ilbl}(\varepsilon'_{i1}))(u_{i'} \mathbin{\widetilde{\Upsilon}} \tilde{\ell}'_{i1}) :: \widetilde{S} \mathbin{\widetilde{\Upsilon}} \tilde{\ell} \mid \mu_i'' \end{array}$$

We need to prove that the resulting values are not observables (otherwise the values can be different and therefore the logical relation fails).

If $\mathsf{obs}_{\ell_o}(v_{i1})$ does not hold then either $\not\exists \varepsilon_{o1}, \varepsilon_{o1} \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \ell_o$, $\not\exists \varepsilon_{o2}, \varepsilon_{o2} \vdash \tilde{\ell}'_{i1} \preccurlyeq \ell_o$, or $ilbl(\varepsilon_{i1}) \circ^{\preccurlyeq} \varepsilon_{o2}$ does not hold.

If $\not\exists \varepsilon_{o1}, \varepsilon_{o1} \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \ell_o$ then, as join is monotone, $\not\exists \varepsilon'_{o1}, \varepsilon'_{o1} \vdash \tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'_{i1} \mathrel{\widetilde{\preccurlyeq}} \ell_o$.

If $\not\exists \varepsilon_{o2}, \varepsilon_{o2} \vdash \tilde{\ell}'_1 \preccurlyeq \ell_o$. Let us assume $\varepsilon'_{o2} \vdash label(\widetilde{S}) \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \preccurlyeq \ell_o$ (otherwise the result holds immediately).

We have to prove that $(ilbl(\varepsilon'_{i'}) \mathbin{\widetilde{\curlyvee}} (ilbl(\varepsilon_{i1}) \circ^{<:} ilbl(\varepsilon_1))) \circ^{\preccurlyeq} \varepsilon'_{o2}$ is not defined. By Lemma 57, $\not\exists \varepsilon'_{o2}, \varepsilon'_{o2} \vdash label(\widetilde{S}_{i'}) \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'_1 \preccurlyeq \ell_o$. We proceed by contradiction. Let us suppose that $(ilbl(\varepsilon'_{i'}) \mathbin{\widetilde{\curlyvee}} (ilbl(\varepsilon_{i1}) \circ^{<:} ilbl(\varepsilon_1))) \circ^{\preccurlyeq} \varepsilon'_{o2}$ is defined. Then by Lemma 55 and 56 $(ilbl(\varepsilon'_{i'}) \mathbin{\widetilde{\curlyvee}} ilbl(\varepsilon_{i1})) \circ^{<:} ((ilbl(\varepsilon'_{i'}) \mathbin{\widetilde{\curlyvee}} ilbl(\varepsilon_1)) \circ^{\preccurlyeq} \varepsilon'_{o2})$ is defined. But $(ilbl(\varepsilon'_{i'}) \mathbin{\widetilde{\curlyvee}} ilbl(\varepsilon_1)) \circ^{\preccurlyeq} \varepsilon'_{o2} \vdash label(\widetilde{S}_{i'}) \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'_1 \preccurlyeq \ell_o$, which is a contradiction and then the result holds immediately.

If $\forall \varepsilon_{o1}, \varepsilon_{o1} \vdash \tilde{\ell}_r \mathrel{\widetilde{\preccurlyeq}} \ell_o, \exists \varepsilon_{o2}, \varepsilon_{o2} \vdash \tilde{\ell}'_1 \preccurlyeq \ell_o$, but $ilbl(\varepsilon_{i1}) \circ^{\preccurlyeq} \varepsilon_{o2}$ does not hold. Then let us assume that $\varepsilon'_{o2} \vdash label(\widetilde{S}) \mathbin{\widetilde{\curlyvee}} \tilde{\ell} \preccurlyeq \ell_o$ (otherwise the values are not observable immediately). We need to prove that $(ilbl(\varepsilon'_{i'}) \mathbin{\widetilde{\curlyvee}} (ilbl(\varepsilon_{i1}) \circ^{<:} ilbl(\varepsilon_1))) \circ^{\preccurlyeq} \varepsilon'_{o2}$ does not hold. We proceed by contradiction. Let us assume that it is defined. But by Lemma 55 and 56, $(ilbl(\varepsilon'_{i'}) \mathbin{\widetilde{\curlyvee}} ilbl(\varepsilon_{i1})) \circ^{<:} ((ilbl(\varepsilon'_{i'}) \mathbin{\widetilde{\curlyvee}} ilbl(\varepsilon_1)) \circ^{\preccurlyeq} \varepsilon'_{o2})$ is defined as well. But as $ilbl(\varepsilon_{i1}) \circ^{\preccurlyeq} \varepsilon_{o2}$ it is not defined, by Lemma 54 $(ilbl(\varepsilon'_{i'}) \mathbin{\widetilde{\curlyvee}} ilbl(\varepsilon_{i1})) \circ^{<:} ((ilbl(\varepsilon'_{i'}) \mathbin{\widetilde{\curlyvee}} ilbl(\varepsilon_1)) \circ^{\preccurlyeq} \varepsilon'_{o2})$ is not defined which is a contradiction and the result holds.

Also by Lemma 63.1, if $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{11}, \mu'_1 \rangle \approx^{k-j_1}_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{21}, \mu'_2 \rangle : \widetilde{S}_1$ then $\langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon v_{11} :: \mathsf{Bool}_{\tilde{\ell}}, \mu'_1 \rangle \approx^{k-j_1}_{\ell_o} \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \varepsilon v_{21} :: \mathsf{Bool}_{\tilde{\ell}}, \mu'_2 \rangle : \widetilde{S}_1$. As the values are not observables then by Lemma 69, $\langle \tilde{\ell}_r, \mu''_1 \rangle \approx^{k-j_1-j_2}_{\ell_o} \langle \tilde{\ell}_r, \mu''_2 \rangle$ and the result holds.

Consider now that $\mathsf{obs}_{\ell_o}(\varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} v_{i1})$ for $i \in \{1, 2\}$. Then by definition of $\approx_{\ell_o}$ on boolean values, $b_{11} = b_{21}$ Because $b_{11} = b_{21}$, both $\sigma_1(t^{\widetilde{S}})$ and $\sigma_2(t^{\widetilde{S}})$ step into the same branch of the conditional. Let us assume the condition is true (the other case is similar).:

$$\sigma_i(t^{\widetilde{S}}) \mid \mu_i \xrightarrow{\varepsilon_r \tilde{\ell}_r \overset{j_1+1}{\longmapsto} \tilde{\ell}_c} \mathsf{prot}^{(\tilde{\ell}_c \widetilde{\curlyvee} \tilde{\ell}), \tilde{\ell}, \widetilde{S}}_{\varepsilon'_{ri}, ilbl(\varepsilon'_{i1}) \tilde{\ell}'_{i1}} \varepsilon_i \sigma_i(t^{\widetilde{S}_2}) \mid \mu'_i$$

Where $\varepsilon'_{ri} = (\varepsilon_r \mathbin{\widetilde{\curlyvee}} ilbl(\varepsilon'_{i1}))$. But $\varepsilon'_{ri}(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'_{i1}) \overset{\tilde{\ell}_c \widetilde{\curlyvee} \tilde{\ell}}{\triangleright} \sigma_i(t^{\widetilde{S}_2}) \in \textsc{Term}_{\widetilde{S}_2}$ and by Lemma 59 and Lemma 66,

$$\begin{array}{rl} & \Gamma \vdash \langle \varepsilon'_{ri}(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'_{i1}), \tilde{\ell}_c \mathbin{\widetilde{\curlyvee}} \tilde{\ell}, \sigma_1, \mu'_1 \rangle \\ \approx^{k-j1-1}_{\ell_o} & \langle \varepsilon'_{ri}(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'_{i1}), \tilde{\ell}_c \mathbin{\widetilde{\curlyvee}} \tilde{\ell}, \sigma_2, \mu'_2 \rangle \end{array}$$

Then by induction hypothesis,

$$\begin{array}{rl} & \langle \varepsilon'_{ri}(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'_{11}) \overset{\tilde{\ell}_c \widetilde{\curlyvee} \tilde{\ell}}{\triangleright} \sigma_1(t^{\widetilde{S}_2}), \mu'_1 \rangle \\ \approx^{k-j_1-1}_{\ell_o} & \langle \varepsilon'_{ri}(\tilde{\ell}_r \mathbin{\widetilde{\curlyvee}} \tilde{\ell}'_{21}) \overset{\tilde{\ell}_c \widetilde{\curlyvee} \tilde{\ell}}{\triangleright} \sigma_2(t^{\widetilde{S}_2}), \mu'_2 \rangle : \mathcal{C}(\widetilde{S}_2) \end{array}$$

And by Lemma 71,

$$\begin{array}{rl} & \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{(\tilde{\ell}_c \widetilde{\curlyvee} \tilde{\ell}), \tilde{\ell}, \widetilde{S}}_{\varepsilon'_{ri}, ilbl(\varepsilon'_{11}) \tilde{\ell}'_{11}} \varepsilon_1 \sigma_1(t^{\widetilde{S}_2}), \mu'_1 \rangle \\ \approx^{k-j1-1}_{\ell_o} & \langle \varepsilon_r \tilde{\ell}_r \overset{\tilde{\ell}_c}{\triangleright} \mathsf{prot}^{(\tilde{\ell}_c \widetilde{\curlyvee} \tilde{\ell}), \tilde{\ell}, \widetilde{S}}_{\varepsilon'_{ri}, ilbl(\varepsilon'_{21}) \tilde{\ell}'_{21}} \varepsilon_2 \sigma_2(t^{\widetilde{S}_2}), \mu'_1 \rangle \end{array}$$

And the result holds by backward preservation of the relations (Lemma 61).

Case (prot). Direct by using Lemma 71.

□

E. Examples of reduction

Consider a simple lattice with four confidentiality levels, $\bot \preccurlyeq \mathsf{L} \preccurlyeq \mathsf{H} \preccurlyeq \top$, a location l_L of a low security value of type Bool_L, and the following *extrinsic* program definitions:

$$(\lambda^{\mathsf{L}} x : \mathsf{Bool}_? . \mathsf{if}\ x\ \mathsf{then}\ l{:=}\mathsf{true}_\mathsf{L}\ \mathsf{else}\ \mathsf{unit}_\mathsf{H}) \mathsf{true}_\mathsf{H}$$

For conciseness, we only annotate the labels of evidences and types. Also we annotate $[\bot, \top] = ?$, and $\langle \ell, \ell \rangle = \ell$. The intrinsic program is reduced as follows:

$$\begin{array}{ll} & (\lambda^{\mathsf{L}} \mathsf{if}\ \langle ?, ? \rangle x^{?}\ \mathsf{then}\ \langle \bot, [\mathsf{H}, \top] \rangle (\langle \mathsf{L}, \mathsf{L} \rangle l^{\mathsf{L}}_{\mathsf{L}} \overset{\langle \mathsf{L}, \mathsf{L} \rangle}{:=} \langle \mathsf{L}, \mathsf{L} \rangle \mathsf{true}_\mathsf{L}) \\ & \quad \mathsf{else}\ \langle \mathsf{H}, [\mathsf{H}, \top] \rangle \mathsf{unit}_\mathsf{H})_\mathsf{L} \langle \mathsf{H}, [\mathsf{H}, \top] \rangle\ \mathsf{true}_\mathsf{H} \\ \overset{\langle \bot, \bot \rangle \bot}{\longmapsto}_\bot & \mathsf{prot}_{\langle \mathsf{L}, \mathsf{L} \rangle, \langle \mathsf{L}, \mathsf{L} \rangle \mathsf{L}} \varepsilon_1 (\mathsf{if}\ \langle ?, ? \rangle \langle \mathsf{H}, [\mathsf{H}, \top] \rangle \mathsf{true}_\mathsf{H} :: ? \\ & \quad \mathsf{then}\ \langle \bot, [\mathsf{H}, \top] \rangle (\langle \mathsf{L}, \mathsf{L} \rangle l^{\mathsf{L}}_{\mathsf{L}} \overset{\langle \mathsf{L}, \mathsf{L} \rangle}{:=} \langle \mathsf{L}, \mathsf{L} \rangle \mathsf{true}_\mathsf{L})\ \mathsf{else} \\ & \qquad \langle \mathsf{H}, [\mathsf{H}, \top] \rangle \mathsf{unit}_\mathsf{H}) \end{array}$$

Then the body of prot is reduced as follows:

$$\begin{array}{ll} & \varepsilon_1 (\mathsf{if}\ \langle ?, ? \rangle \langle \mathsf{H}, [\mathsf{H}, \top] \rangle \mathsf{true}_\mathsf{H} :: ? \\ & \quad \mathsf{then}\ \langle \bot, [\mathsf{H}, \top] \rangle (\langle \mathsf{L}, \mathsf{L} \rangle l^{\mathsf{L}}_{\mathsf{L}} \overset{\langle \mathsf{L}, \mathsf{L} \rangle}{:=} \langle \mathsf{L}, \mathsf{L} \rangle \mathsf{true}_\mathsf{L})\ \mathsf{else} \\ & \qquad \langle \mathsf{H}, [\mathsf{H}, \top] \rangle \mathsf{unit}_\mathsf{H}) \\ \overset{\langle \mathsf{L}, \mathsf{L} \rangle \mathsf{L}}{\longmapsto}_\mathsf{L} & \varepsilon_1 (\mathsf{if}\ \langle \mathsf{H}, [\mathsf{H}, \top] \rangle \mathsf{true}_\mathsf{H} \\ & \quad \mathsf{then}\ \langle \bot, [\mathsf{H}, \top] \rangle (\langle \mathsf{L}, \mathsf{L} \rangle l^{\mathsf{L}}_{\mathsf{L}} \overset{\langle \mathsf{L}, \mathsf{L} \rangle}{:=} \langle \mathsf{L}, \mathsf{L} \rangle \mathsf{true}_\mathsf{L})\ \mathsf{else} \\ & \qquad \langle \mathsf{H}, [\mathsf{H}, \top] \rangle \mathsf{unit}_\mathsf{H}) \\ \overset{\langle \mathsf{L}, \mathsf{L} \rangle \mathsf{L}}{\longmapsto}_\mathsf{L} & \mathsf{prot}_{\langle \mathsf{H}, [\mathsf{H}, \top] \rangle, \langle \mathsf{H}, [\mathsf{H}, \top] \rangle \mathsf{H}} \langle \bot, [\mathsf{H}, \top] \rangle (\ \langle \mathsf{L}, \mathsf{L} \rangle l^{\mathsf{L}}_{\mathsf{L}} \overset{\langle \mathsf{L}, \mathsf{L} \rangle}{:=} \langle \mathsf{L}, \mathsf{L} \rangle \mathsf{true}_\mathsf{L}\ ) \end{array}$$

Once again the body of the new prot is reduced as follows:

$$\begin{array}{ll} & \langle \mathsf{L}, \mathsf{L} \rangle l^{\mathsf{L}}_{\mathsf{L}} \overset{\langle \mathsf{L}, \mathsf{L} \rangle}{:=} \langle \mathsf{L}, \mathsf{L} \rangle \mathsf{true}_\mathsf{L} \\ \overset{\langle \mathsf{H}, [\mathsf{H}, \top] \rangle \mathsf{H}}{\longmapsto}_? & \textbf{error} \end{array}$$

because

$$\begin{array}{l} ((\langle \mathsf{H}, [\mathsf{H}, \top] \rangle \mathbin{\widetilde{\curlyvee}} \langle \mathsf{L}, \mathsf{L} \rangle) \circ^{\preccurlyeq} \langle \mathsf{L}, \mathsf{L} \rangle) \circ^{\preccurlyeq} \langle \mathsf{L}, \mathsf{L} \rangle = \\ \qquad (\triangle^{\preccurlyeq}(\mathsf{H}, [\mathsf{H}, \top] \sqcap \mathsf{L}, \mathsf{L})) \circ^{\preccurlyeq} \langle \mathsf{L}, \mathsf{L} \rangle \end{array}$$

which is undefined because $[\mathsf{H}, \top] \sqcap \mathsf{L}$ is undefined.

This is the expected behavior because the first branch of the conditional breaks noninterference in the presence of a higher security context. The same function applied with a different argument:

$$(\lambda^{L} x : \mathsf{Bool}_? . \mathsf{if}\ x\ \mathsf{then}\ l := \mathsf{true}_\mathsf{L}\ \mathsf{else}\ \mathsf{unit}_\mathsf{H})\ \mathsf{false}_\mathsf{H}$$

reduces as follows:

$$\begin{array}{ll} & (\lambda^{\mathsf{L}} \mathsf{if}\ \langle ?, ? \rangle x^{?}\ \mathsf{then}\ \langle \bot, [\mathsf{H}, \top] \rangle (\langle \mathsf{L}, \mathsf{L} \rangle l^{\mathsf{L}}_{\mathsf{L}} \overset{\langle \mathsf{L}, \mathsf{L} \rangle}{:=} \langle \mathsf{L}, \mathsf{L} \rangle \mathsf{true}_\mathsf{L}) \\ & \quad \mathsf{else} \\ & \qquad \langle \mathsf{H}, [\mathsf{H}, \top] \rangle \mathsf{unit}_\mathsf{H}) \\ \overset{\langle \bot, \bot \rangle \bot}{\longmapsto}_\bot & \mathsf{prot}_{\langle \mathsf{L}, \mathsf{L} \rangle, \langle \mathsf{L}, \mathsf{L} \rangle \mathsf{L}} \varepsilon_1 (\mathsf{if}\ \langle ?, ? \rangle \langle \mathsf{H}, [\mathsf{H}, \top] \rangle \mathsf{true}_\mathsf{H} :: ? \\ & \quad \mathsf{then}\ \langle \bot, [\mathsf{H}, \top] \rangle (\langle \mathsf{L}, \mathsf{L} \rangle l^{\mathsf{L}}_{\mathsf{L}} \overset{\langle \mathsf{L}, \mathsf{L} \rangle}{:=} \langle \mathsf{L}, \mathsf{L} \rangle \mathsf{true}_\mathsf{L})\ \mathsf{else} \\ & \qquad \langle \mathsf{H}, [\mathsf{H}, \top] \rangle \mathsf{unit}_\mathsf{H}) \end{array}$$

Then the body of prot is reduced as follows:

$$
\begin{array}{ll}
 & \varepsilon_1(\mathsf{if}\ \langle ?, ?\rangle\, \langle \mathsf{H}, [\mathsf{H}, \top]\rangle\, \mathsf{true}_{\mathsf{H}} :: ? \\
 & \quad \mathsf{then}\ \langle \bot, [\mathsf{H}, \top]\rangle\, (\langle \mathsf{L}, \mathsf{L}\rangle\, l^{\mathsf{L}}_{\mathsf{L}} \overset{\langle \mathsf{L},\mathsf{L}\rangle}{:=} \langle \mathsf{L}, \mathsf{L}\rangle\, \mathsf{true}_{\mathsf{L}})\ \mathsf{else} \\
 & \qquad\qquad \langle \mathsf{H}, [\mathsf{H}, \top]\rangle\, \mathsf{unit}_{\mathsf{H}}) \\
\overset{\langle \mathsf{L},\mathsf{L}\rangle \mathsf{L}}{\longmapsto}_{\mathsf{L}} & \varepsilon_1(\mathsf{if}\ \langle \mathsf{H}, [\mathsf{H}, \top]\rangle\, \mathsf{true}_{\mathsf{H}} \\
 & \quad \mathsf{then}\ \langle \bot, [\mathsf{H}, \top]\rangle\, (\langle \mathsf{L}, \mathsf{L}\rangle\, l^{\mathsf{L}}_{\mathsf{L}} \overset{\langle \mathsf{L},\mathsf{L}\rangle}{:=} \langle \mathsf{L}, \mathsf{L}\rangle\, \mathsf{true}_{\mathsf{L}})\ \mathsf{else} \\
 & \qquad\qquad \langle \mathsf{H}, [\mathsf{H}, \top]\rangle\, \mathsf{unit}_{\mathsf{H}}) \\
\overset{\langle \mathsf{L},\mathsf{L}\rangle \mathsf{L}}{\longmapsto}_{\mathsf{L}} & \mathsf{prot}_{\langle \mathsf{H},[\mathsf{H},\top]\rangle, \langle \mathsf{H},[\mathsf{H},\top]\rangle \mathsf{H}}\ \langle \mathsf{H}, [\mathsf{H}, \top]\rangle\, \mathsf{unit}_{\mathsf{H}} \\
\overset{\langle \mathsf{L},\mathsf{L}\rangle \mathsf{L}}{\longmapsto}_{\mathsf{L}} & \langle \mathsf{H}, [\mathsf{H}, \top]\rangle\, \mathsf{unit}_{\mathsf{H}} :: ?
\end{array}
$$

no error were produced, because now the second branch of the conditional does not break noninterference.