Interactive Proofs in Higher-Order Logic with Errors and Application to Concrete Cryptography

A short version of this paper will appear in CSF'26.

Caroline Fontaine [‡], Adrien Koutsos [†], Guillaume Scerri [‡], Théo Vignon [‡]
[†] Inria, Paris, France
[‡] Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, Gif-sur-Yvette, France

***Index Terms*—Computer-aided Cryptography, Concrete Cryptography, Logic, Interactive Proofs.**

***Abstract*—Computer-aided cryptography (CAC) provides strong guarantees through mechanized proofs of security. SQUIRREL is a proof assistant specialized in CAC, but is restricted to the asymptotic setting, which limits its applicability. Recent theoretical work [1] adapted SQUIRREL's underlying logic to the concrete setting through the introduction of a higher-order logic with errors. While this allows to prove precise security bounds on paper, it only provides a low-level logical calculus which lacks an implementation. Thus, it falls short of the CAC aims.**

In this paper, we use this low-level calculus to build the full-fledged set of features used in a proof assistant such as SQUIRREL, with a focus on reachability reasoning. We design higher-level logical mechanisms on top of this logic, including proof context management, introduction patterns, and bound-annotated tactics. To do so, we introduce a proof term calculus with dedicated features for bounds, for which we design an elaborator. This elaborator can automatically infer bound-related manipulations, improving usability and reducing user inputs, and we theoretically argue for its usability through an erasability theorem. All these improvements have been implemented as an extension of SQUIRREL, and we provide empirical evidence of the applicability of our framework through case studies.

I. INTRODUCTION

Communication systems are secured through the use of cryptography. A large variety of *cryptographic designs* are used for that purpose, ranging from cryptographic primitives (e.g., encryption or signature) to security protocols such as TLS [2], or even authenticated data-structures techniques [3] such as Merkle trees [4].

Provable cryptography [5] aims at establishing the security of cryptographic designs using mathematical theorems. It relates the winning probability of a *resource-bounded* adversary with a *security parameter* η — which is related to the secret key space size, and roughly corresponds to the length of the secret keys used — and comes in two flavors. *Asymptotic security* provides qualitative guarantees on the security of a cryptographic design by showing that the adversary winning probability tends to zero fast enough when the security parameter grows. *Concrete security* [6] goes one-step further, and provides precise upper-bounds on the adversary's winning probability. Following the code-centric view of cryptography [7], such results are typically of the form:

$$\Pr\left(\mathcal{G}_{\mathcal{A}}(\eta)\right) \leq \varepsilon(\eta, t)$$

where $\mathcal{G}$ is a program encoding a cryptographic experiment in which an adversary $\mathcal{A}$, running in time at-most t, tries to break a particular cryptographic design w.r.t. the security parameter η. Concrete security proofs are more complex than asymptotic proofs as they usually require low-level manipulations of error bounds, but they provide more precise results. Moreover, they are necessary for particular cryptographic arguments such as hybrid proofs [8], and to analyze the security of some cryptographic designs (e.g., when low-entropy secrets are involved as in a password-authenticated key-exchange [9]).

Computer-aided cryptography [10] provides a very high level of guarantees by mechanizing the cryptographic proofs in a dedicated formal verification framework. For example, SQUIRREL [11], [12] is an interactive theorem prover based of the Computationally Complete Symbolic Attacker (CCSA) logic. This logic, first introduced in [13] and then extended into a higher-order logic in [14], captures cryptographic proofs. SQUIRREL has notably been used to verify the security of the YubiKey protocol [15], of the FOO e-voting protocol [16], and of several post-quantum key-exchange protocols [17]. But because SQUIRREL's current underlying logic is *asymptotic* [14], these proofs suffer from the limitation discussed above: they lack concrete security bounds and only show security for a parametric number of sessions [1]. Our goal is to remedy to this situation and to extend SQUIRREL to a concrete security setting.

Tactic-based proof assistants: To understand the issues we have to address to achieve this goal, it is necessary to discuss how SQUIRREL operates. SQUIRREL is a proof assistant dedicated to the mechanization of cryptographic designs. A SQUIRREL proof is built interactively by the user, in a similar way to ROCQ [18] or LEAN [19]. At any point during a proof, the tool displays the *proof context* to the user, i.e., the list of all known assumptions, and the current proof goal. Then, the user operates on the proof context through *tactics*, which are commands instructing SQUIRREL to modify the context in a sound-by-construction fashion. For example, the user may rewrite a known equality using the rewrite

This work received funding from the France 2030 program managed by the French National Research Agency under grant agreement No. ANR-22-PECY-0006.

tactic, or apply a lemma to the proof goal using apply. Proof assistants such as SQUIRREL rely on a number of mechanisms to reduce the proof burden put on the user. For example, *introduction patterns* [20] ease proof context management, *proof terms* allow to manipulate and combine assumptions, and the *elaborator* [21] reduces user inputs by automatically inferring arguments, discharging hypotheses as subproofs, etc.

Higher-order logic with error: A recent work [1] has shown how to adapt SQUIRREL's underlying logic to a concrete security setting. Concretely, the authors proposed a higher-order logic with errors, which we denote by HO_ε in this paper, and they showed how the asymptotic proof system of [14] could be modified with explicit error annotations to track probabilities of failure. While this provides a first step toward mechanized concrete cryptographic proofs in SQUIRREL, this work falls short of this goal in several ways. First, their contribution is purely theoretical and they did not implement HO_ε in SQUIRREL. Second, they only provided a low-level sequent calculus for HO_ε, which is at a significant distance from the high-level logical mechanisms used in a tactic-based proof assistant such as SQUIRREL — as there is some distance between the calculus of inductive construction [22] and the ROCQ or LEAN proof assistants.

Contributions: In this paper, we lift all these limitations through the following contributions:

i) We extend SQUIRREL to support interactive proofs of higher-order logic with errors by designing *higher-level logical mechanisms* on top of the low-level HO_ε logic of [1], including *proof context management*, *bound-annotated tactics*, and *introduction patterns*.
ii) We propose a *proof term calculus* for HO_ε with novel features related to our particular application domain. Then, we design an *elaborator* for this proof term calculus. This elaborator, which we implemented in SQUIRREL, has dedicated supports for error bounds and can automatically infer many low-level details omitted by the user, reducing boilerplate and improving usability.
iv) We *theoretically* argue for the usability of our elaborator through an *erasability theorem*: roughly, we prove that our elaborator can reconstruct some explicit bound annotations that have been erased from a proof term. Concretely, this means that the user may omit such annotations when passing proof terms arguments to tactics, allowing for more concise and readable proof scripts.
v) We provide *empirical evidence* of the applicability of our framework through three case studies in our extension of SQUIRREL: we prove concrete security bounds for Merkle hash-trees, we formalize the Birthday bound, and we adapt the security proof of the YubiKey protocol of [15] to a concrete setting.

Limitation: This work focuses on interactive proofs of higher-order logic with errors, and its application to concrete security proofs of *correspondence* properties [23], [24]. Notably, we only consider the *reachability* logic of [1], and did not target its *indistinguishability* logic. Indeed, mechanizing concrete indistinguishability proofs with precise advantage bounds is a complex endeavor that requires to extend the verification tool with a complexity analysis framework (e.g., as in [25]), which is out-of-scope of this work.

Artifacts: Our SQUIRREL extension, case studies and companion files are available online [26].

Outline: We present some background in Section II. In Section III, we explain how we adapted SQUIRREL to support approximated reasoning. In Section IV, we describe our novel elaborator and its properties. We present our case studies in Section V, and finally discuss related work in Section VI.

We use colors to help distinguish elements of distinct categories. To ensure accessibility to color-blind readers, colors only encode redundant information.

II. BACKGROUND

We start with some high-level background on tactic-based proof assistants and an informal presentation of SQUIRREL's two-level asymptotic logic.

A. Tactic-Based Proof Assistants

SQUIRREL is a *goal-directed proof assistant*, as many other tools such as ROCQ [18] or LEAN [19], in which users manipulate *goals* representing *judgements* through *tactics*.

a) Judgements and goals: In a first approximation, SQUIRREL's judgements are of the form $\mathbb{E}; \Gamma \vdash \phi$, where: the environment $\mathbb{E}$ is a list of variables declarations and definitions, the *hypotheses* set Γ is a set of formulas, and the *conclusion* ϕ is a formula. The conjunction of $\mathbb{E}$ and Γ forms the *proof context* of the judgement. For now, it will be sufficient to think of Γ or ϕ as higher-order logic formulas (we provide more details later, in Section II-B). A SQUIRREL goal, being of the form

```
Variables: x_1,...,x_n : τ, ...      (* declarations of 𝔼 *)
y_1 := t_1 ... y_l := t_l            (* definitions of 𝔼 *)
H_1: ψ_1
...                  (* hypotheses Γ = {ψ_1, ..., ψ_m} *)
H_m: ψ_m
------------------------------------------------------
φ                                    (* conclusion *)
```

is a beautified representation of a judgement in which hypotheses are named (e.g., H_i refers to ψ_i in the goal above).

b) Logical rules and tactics: A judgement can be shown to be valid using *logical rules*, of the form

$$\frac{\mathbb{E}_1; \Gamma_1 \vdash \phi_1 \qquad \dots \qquad \mathbb{E}_n; \Gamma_n \vdash \phi_n}{\mathbb{E}; \Gamma \vdash \phi}$$

Such rules are said to be *sound* whenever the validity of the premises $(\mathbb{E}_i; \Gamma_i \vdash \phi_i)_{i \in \{1,\dots,n\}}$ entails that of the conclusion $\mathbb{E}; \Gamma \vdash \phi$. *Tactics* operate similarly, by changing the current goal into one or several goal(s) whose validity entails that of the initial goal, and are justified by rules of the logic.

We show an example of a proof in Listing 1. While this example is written in SQUIRREL, it would go similarly in other proof assistants such as ROCQ or LEAN. In the following explanation, and throughout the paper, we refer to line numbers in red, e.g., as 1. We start by declaring the lemma's name and universally quantified parameters (1), and conclusion (2). After entering *proof mode* (3), SQUIRREL displays the goal:

```
1  lemma classical (P,R : bool) (Q : int → bool) :
2      ((P ⇒ R) ∧ ∀ x, Q x ⇒ R) ⇒ (P ∨ ∃ a, Q a) ⇒ R.
3  Proof.
4    intro Hand. intro Hor.
5    destruct Hand as [H H0].
6    case Hor.
7    + apply H.
8      assumption Hor.
9    + destruct Hor as [a H1].
10     apply (H0 a H1).
11 Qed.
```

Listing 1: Proof in a classical setting.

```
1  global lemma two_level (P,R : bool) (Q : int → bool) :
2      ([ P ⇒ R ] ∧̄ [ ∀̇ x, Q x ⇒ R ]) ⇛ [ (P ∨̇ ∃̇ a, Q a) ⇒ R ].
3  Proof.
4    intro Hand. intro Hor.
5    destruct Hand as [H H0].
6    case Hor.
7    + localize H as Hl. apply Hl.
8      assumption Hor.
9    + destruct Hor as [a H1].
10     localize H0 as H0l. apply H0l a H1.
11 Qed.
```

Listing 2: Proof in the two-level asymptotic logic.

```
1  intro [H H0] [Hor | [a H1]].
2  + apply H; assumption Hor.
3  + apply (H0 _ H1).
```

Listing 3: Alternative proof for Listing 1.

```
1  intro [H H0]; intro [Hor | [a H1]].
2  + apply H; assumption Hor.
3  + apply (localize(H0) _ H1).
```

Listing 4: Alternative proof for Listing 2.

```
Variables: P,R:bool,Q:int → bool
──────────────────────────────────────
((P ⇒ R) ∧ ∀ x, Q x ⇒ R) ⇒ (P ∨ ∃ a, Q a) ⇒ R
```

When the conclusion is $\phi \Rightarrow \psi$, the intro tactic moves ϕ from the conclusion to the proof context, which yields a new conclusion ψ, and adds the hypothesis ϕ to the proof context with a given user-chosen name. E.g., after both intros of line 4, the goal becomes:

```
Variables: P,R:bool,Q:int → bool
Hand: (P ⇒ R) ∧ ∀ x, Q x ⇒ R
Hor: P ∨ ∃ a, Q a
──────────────────────────────────────
R
```

The intro tactic allows for *backward reasoning* by modifying the goal's conclusion. It is also possible to do *forward reasoning*, by operating on the proof context. For example, destruct H as [H0 H1] replaces an hypothesis H: $\phi \wedge \psi$ by two hypotheses H0: ϕ and H1: ψ. The case H tactic performs a case disjunction on H: $\phi \vee \psi$ and creates one subgoal for each disjunct of H, where H maps to ϕ (respectively ψ) in the first (respectively second) subgoal. E.g., the call to case (6) creates two goals whose hypotheses are, respectively:

```
H: P ⇒ R                          H: P ⇒ R
H0: ∀ x, Q x ⇒ R      and         H0: ∀ x, Q x ⇒ R        (1)
Hor: P                            Hor: ∃ a, Q a
```

To improve readability, the proof of each created subgoal starts with a *bullet* +. E.g., the script in 9,10 proves the second goal, using the right hypotheses in Eq. (1). This proof starts by destructing the existential (9), replacing Hor with the witness a and introducing a new hypothesis H1: Q a.

While some tactics correspond to a single rule of the logic, tactics often combine many rules together. Such higher-level logical mechanisms are crucial to allow for a smoother user experience and to reduce the proof burden. The call to apply (10) is a good example of that: in a single invocation, it instantiates H0's universally quantified variable x by a, discharges the proof obligation (Q a) using the assumption H1, and concludes the proof by using the formula R obtained as a result. The argument (H0 a H1) of apply is called a *proof term*, and is a powerful tool allowing for concise proofs.

c) Advanced tactics: More advanced features can be used to reduce the size of the proof even further. E.g., we show in Listing 3 an alternative proof of the lemma of Listing 1, which is more concise thanks to several features. First, it uses *intro patterns* [20] to manipulate the proof context in a compact way. E.g., at 1, the first pattern [H H0] splits the conjunction ∧. An or-pattern [A | B] performs a case-analysis on a disjunction ∨, naming the left and right disjunct respectively A and B. Here, the or-pattern has a nested subpattern [a H1] decomposing the existential (∃ a. Q a). After 1, we are in exactly the same situation as after 6 of Listing 1. Second, this more compact version uses *tactic combinator* $\mathsf{tac}_1;\mathsf{tac}_2$ to apply the tactic tac_2 to all subgoals produced by tac_1. Finally, it exploits SQUIRREL's proof term *elaborator* in 3 to omit the argument a in the proof term. Here, the elaborator infers that x should be instantiated by a, by unifying the formula (Q a) given by H1 with the assumption (Q x) expected by H.

B. A Two-Level Logic for Asymptotic Cryptography

We will now quickly recall the CCSA multi-context logic for asymptotic cryptography [14] and its implementation in SQUIRREL, on which we will focus in the rest of this work.

a) Terms: We consider a set of *base types* $\mathbb{T}$ which includes standard types such as bool, and real. A type τ is either a base type $\tau_b \in \mathbb{T}$, or an arrow type $\tau_0 \to \tau_1$. Terms are simply typed λ-terms built on top of a set of variables $\mathcal{X}$:

$$t \stackrel{\text{def}}{=} x \mid t\ t \mid \lambda(x:\tau).t \qquad (\text{where } x \in \mathcal{X})$$

An *environment* $\mathbb{E}$ is a finite set of *declarations* $(x : \tau)$ and *definitions* of variables $(x : \tau = t)$. We assume that all environments declare the standard local logical operators, e.g.,

$$\dot{\top} : \mathsf{bool} \qquad \dot{\wedge} : \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool} \qquad \dot{\forall}_\tau : (\tau \to \mathsf{bool}) \to \mathsf{bool}$$

which will have the expected interpretation and are used with standard notation. E.g., we may write $\dot{\forall}(x : \mathsf{real}).t_0 \ \dot{\wedge}\ t_1$. We consider a standard type system $\mathbb{E} \vdash t : \tau$ (c.f. [14]), and only consider well-typed terms from now on (the typing environment $\mathbb{E}$ to be used will be clear from the context). A *local formula* $\dot{\phi}$ is a term of type bool. Please note that Local formulas may be of higher order. In the following, for ease of reading we will denote local formulas, connectives, and quantifiers in blue with a dot $\cdot$ over them.

b) Term semantics: We refer the reader to [14] for a detailed presentation of the semantics, and only recall its main features here. This semantics is tailored for cryptography, and is thus a probabilistic semantics parameterized by the security parameter $\eta \in \mathbb{N}$. More precisely, a *model* $\mathbb{M}$ of an environment $\mathbb{E}$ is a first-order model providing an interpretation of:

- each type as a sequence of interpretation domains $(\llbracket\tau\rrbracket^\eta_\mathbb{M})_{\eta\in\mathbb{N}}$ indexed by the security parameter η.
- each term t of type τ in $\mathbb{E}$ as a η-indexed sequence of random variables:

$$(\rho \in \mathbb{T}_{\mathbb{M},\eta}) \mapsto \llbracket t\rrbracket^{\eta,\rho}_\mathbb{M} \in \llbracket\tau\rrbracket^\eta_\mathbb{M}$$

  where $\mathbb{T}_{\mathbb{M},\eta}$ is a set of *finite* random tapes of some fixed length. Note that the random tape ρ provides randomness to the semantics $\llbracket\cdot\rrbracket^{\eta,\rho}_\mathbb{M}$. E.g., to interpret a variable k representing a uniformly sampled random key of length η, $\llbracket\mathsf{k}\rrbracket^{\eta,\rho}_\mathbb{M}$ will retrieve η random bits from ρ.

Standard types and functions must be interpreted as expected, e.g., $\llbracket\mathsf{real}\rrbracket^\eta_\mathbb{M} = \mathbb{R}$ and $\llbracket\dot\wedge\rrbracket^{\eta,\rho}_\mathbb{M}(x,y) = 1$ iff $x = y = 1$.

c) Asymptotic security logic: A typical goal of asymptotic security is to prove that the probability that an adversary breaks a security property is *negligible* in the security parameter η, where a function $f(\eta)$ is *negligible* iff it asymptotically tends to 0 faster than the inverse of any polynomial in η. In the CCSA logic, such probabilistic properties can be captured using the *overwhelming truth predicate* $[\dot\phi]$, which states that a Boolean term $\dot\phi$ is true with overwhelming probability. More precisely, $[\dot\phi]$ is satisfied in a model $\mathbb{M}$ of the logic iff

$$\Pr(\llbracket[\dot\phi]\rrbracket^\eta_\mathbb{M}) \text{ is overwhelming in } \eta,$$

where $f(\eta)$ is overwhelming iff $1 - f(\eta)$ is negligible.

Then, we consider first-order logic using this $[\,\cdot\,]$ predicate:

$$\tilde\phi_\mathsf{a} ::= [\dot\phi] \mid \tilde\phi_\mathsf{a} \mathbin{\tilde\vee} \tilde\phi_\mathsf{a} \mid \tilde\neg\tilde\phi_\mathsf{a} \mid \tilde\forall(x:\tau).\,\tilde\phi_\mathsf{a} \mid \dots$$

We call such connectives and quantifiers *global*, and $\tilde\phi_\mathsf{a}$ is a global formula of the asymptotic logic. For ease of reading, global elements are denoted in the following in red with a tilde $\sim$ over them. As usual in first-order logics, we write $\mathbb{M} \models \tilde\phi_\mathsf{a}$ when $\mathbb{M}$ satisfies $\tilde\phi_\mathsf{a}$, and $\models \tilde\phi_\mathsf{a}$ when $\tilde\phi_\mathsf{a}$ is valid, i.e., when $\tilde\phi_\mathsf{a}$ is satisfied by all models.

This yields a two-level logic, where relations between local inner-logic formulas are captured using the global outer logic.

SQUIRREL's judgements reflect this two-level structure:

- In a *global asymptotic judgement* $\mathbb{E};\Gamma \vdash \tilde\phi_\mathsf{a}$, the *global context* Γ is a finite set of asymptotic global formulas. It is valid iff $(\tilde\wedge\Gamma) \mathrel{\tilde\Rightarrow} \tilde\phi_\mathsf{a}$ is satisfied by any model of $\mathbb{E}$.
- A *local judgement* $\mathbb{E};\Gamma;\Lambda \vdash \dot\phi$ has an additional *local context* Λ (which is a finite set of local formulas) and a local conclusion $\dot\phi$. It is valid iff $\Gamma \vdash [(\dot\wedge\Lambda) \mathrel{\dot\Rightarrow} \dot\phi]$ is valid.

Global hypotheses in Γ are considered w.r.t. an overwhelmingly notion of truth, while local hypotheses in Λ are considered for a fixed value of the random tape. As an example, $[\dot\phi] \mathbin{\tilde\vee} [\dot\neg\dot\phi]$ is not a tautology and only holds when $\dot\phi$ is overwhelmingly true or $\dot\neg\dot\phi$ is overwhelming true. However, the local formula $\dot\phi \mathbin{\dot\vee} \dot\neg\dot\phi$ is tautological, as for any fixed random tape ρ, $\llbracket\dot\phi\rrbracket^{\eta,\rho}_\mathbb{M}$ is either true or false.

When there is no ambiguity, we omit the environment $\mathbb{E}$. E.g., we write $\Gamma \vdash \tilde\phi_\mathsf{a}$ instead of $\mathbb{E};\Gamma \vdash \tilde\phi_\mathsf{a}$. A proof context Δ is a triple $(\mathbb{E};\Gamma;\Lambda)$ composed of a typing environment $\mathbb{E}$, a global (hypothesis) context Γ, and a local (hypothesis) context Λ. For example, if $\Delta = \mathbb{E};\Gamma;\Lambda$ then $\Delta \vdash \dot\phi$ denotes the local judgement $\mathbb{E};\Gamma;\Lambda \vdash \dot\phi$.

d) Squirrel implementation: Global and local judgements support standard tactics, as presented in Section II-A. E.g., case can be used to do a case analysis thanks to the rules:

$$\frac{\Gamma, \tilde\phi_{\mathsf{a},0} \vdash \tilde\phi_\mathsf{a} \quad \Gamma, \tilde\phi_{\mathsf{a},1} \vdash \tilde\phi_\mathsf{a}}{\Gamma, \tilde\phi_{\mathsf{a},0} \mathbin{\tilde\vee} \tilde\phi_{\mathsf{a},1} \vdash \tilde\phi_\mathsf{a}} \qquad \frac{\Gamma;\Lambda, \dot\phi_0 \vdash \dot\phi \quad \Gamma;\Lambda, \dot\phi_1 \vdash \dot\phi}{\Gamma;\Lambda, \dot\phi_0 \mathbin{\dot\vee} \dot\phi_1 \vdash \dot\phi}$$

Global formulas have a standard first-order logic semantics. Thus, the left rule, which operates on the global context, is expected. There is a similar rule operating on the *global* context of a *local judgement*, which is expected too. The right rule is more surprising, as it operates on the local context *under* the overwhelming truth predicate. Essentially, its soundness relies on the union bound property [27] and the fact that the sum of two negligible quantities remains negligible. More generally, all standard logical tactics are supported in both kinds of contexts and judgements [14].

The environment $\mathbb{E}$ does not differentiate between local and global variables. This is because, as shown in [14]:

$$\mathbb{E};\Gamma \vdash [\dot\forall x.\,\dot\phi] \mathrel{\tilde\Leftrightarrow} \tilde\forall x.\,[\dot\phi]$$

is valid, i.e., we can swap a quantification with the $[\,\cdot\,]$ predicate. Thanks to this, if a goal conclusion is $[\dot\forall(x:\tau).\,\dot\phi]$, a call to intro will introduce a new declaration $(x:\tau)$ in the proof context, and will change the conclusion to $[\dot\phi]$.

Finally, some rules mix both kinds of contexts. Notably, hypotheses can move from the global to the local contexts, thanks to the localize tactic based on the rule (from [14]):

$$\frac{\Gamma;\Lambda, \dot\phi_0 \vdash \dot\phi}{\Gamma, [\dot\phi_0];\Lambda \vdash \dot\phi}$$

Indeed, the negligible probability that $\dot\phi$ does not hold is absorbed by the fact that the conclusion $[(\dot\wedge\Lambda) \mathrel{\dot\Rightarrow} \dot\phi]$ only needs to be established up to a small probability of failure. Interestingly, there is no converse rule which would allow to move a local hypothesis to the global context.

Example 1. *We consider in Listing 2 a variation of Listing 1 using the $[\,\cdot\,]$ predicate. Most of the proof remains the same, except that we use* localize *to move a global hypothesis to the local context before applying it to the (local) conclusion (7,9). As in the previous section, more advanced features can reduce the amount of necessary user inputs. For example, in the proof in Listing 4, the localization is automatically inferred at 2, and it is done directly in the proof term at 3 .*

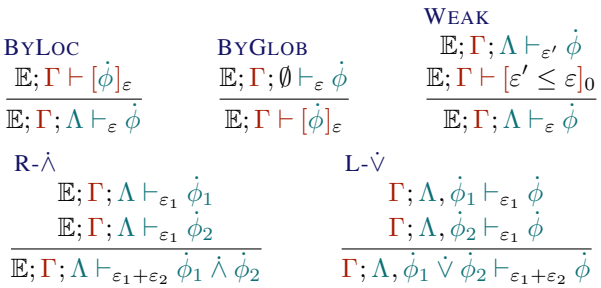


Figure 1: Selected concrete logic rules.

C. A Two-Level Logic for Concrete Cryptography

We now recall the adaptation of this two-level logic to concrete security, as introduced in [1]. Please notice that, contrary to the asymptotic logic presented in the previous section, the concrete security framework from [1] has not been implemented in SQUIRREL before our present work.

a) Concrete security logic: We consider the truth predicate $[\dot{\phi}]_\varepsilon$ with an explicit error bound ε of type real. It is satisfied in a model $\mathbb{M}$ iff

$$\mathsf{Pr}_{\rho \in \mathbb{T}_{\mathbb{M},\eta}} \left([\![\dot{\phi}]\!]^{\eta,\rho}_{\mathbb{M}} = 0 \right) \leq \mathsf{E}_{\rho \in \mathbb{T}_{\mathbb{M},\eta}} [\![\varepsilon]\!]^{\eta,\rho}_{\mathbb{M}},$$

where $\mathsf{E}(X)$ is the expectation of X. Usually, ε will be a deterministic value that does not depend on ρ, e.g., $[\![\varepsilon]\!]^{\eta,\rho}_{\mathbb{M}} = f(\eta)$. In that case, the expectation is simply $f(\eta)$. The *concrete global logic* is first-order logic using the $[\,\cdot\,]_\varepsilon$ predicate:

$$\tilde{\phi} ::= [\dot{\phi}]_\varepsilon \mid \tilde{\phi} \,\tilde{\vee}\, \tilde{\phi} \mid \tilde{\neg} \tilde{\phi} \mid \tilde{\forall}(x : \tau).\, \tilde{\phi} \mid \ldots$$

It yields a two-level logic whose judgements are

$$\mathbb{E}; \Gamma \vdash \tilde{\phi} \qquad \mathbb{E}; \Gamma; \Lambda \vdash_\varepsilon \dot{\phi}$$

where the global context Γ is a set of concrete global formulas. Concrete global judgements have the same semantics as their asymptotic counter-parts. A concrete local judgement $\mathbb{E}; \Gamma; \Lambda \vdash_\varepsilon \dot{\phi}$ is valid iff $\mathbb{E}; \Gamma \vdash [(\dot{\wedge}\Lambda) \Rrightarrow \dot{\phi}]_\varepsilon$ is valid.

Please note that from now on, for the sake of conciseness and as our work only focuses on concrete security, we will simply say global logic instead of concrete global logic.

b) Proof system: This logic is equipped with a sound proof system [1]. The rules dealing with global judgements are similar to the standard first-order logic rules, and are thus omitted here. We only detail a few rules, listed in Fig. 1, that deal with either interactions between the global and local levels, or with error bounds. More rules will be presented later in Section III, along with their corresponding tactics.

The rule ByLoc builds a local judgement from a global formula, transferring the error bound from the formula to the judgement. The rule ByGlob builds a global formula from a local judgement. Crucially, the local context of the local judgement must be empty, to ensure that the formula holds with probability ε without additional assumptions. The Weak rule weakens a bound ε' into a larger bound ε. Finally, the rules R-$\dot{\wedge}$ and L-$\dot{\vee}$ are similar to their first-order logic counter-parts, except that they also sum the errors from each premise.

```
1  global lemma concrete (P,R : bool) (Q : int → bool) (eP : real) (eQ : real):
2    ([ P ⇒ R <: eP ] ∧̇ [ ∀̇ x, Q x ⇒ R <: eQ ]) ⇛
3    [ (P ∨̇ ∃̇ a, Q a) ⇒ R <: eP + eQ ].
4  Proof.
5    intro Hand. intro Hor.
6    destruct Hand as [H H0].
7    case Hor <: eP.
8    + localize H as Hl. apply Hl.
9      assumption Hor.
10   + destruct Hor as [a H1].
11     localize H0 as H0l. apply (H0l a H1).
12     true; smt.
13 Qed.
```

Listing 5: Proof in a concrete setting.

```
1  intro [H H0]; intro [Hor <: eP | [a H1] <: eQ].
2  + auto.  (* bound reasoning *)
3  + apply H; assumption Hor.
4  + apply (localize(H0) _ H1); auto.
```

Listing 6: Alternative proof for Listing 5.

III. Interactive Proofs of HO Logic with Errors

We now present how we extend previous theoretical work [1] to improve the SQUIRREL tool and make it handle concrete security properly. We adapt SQUIRREL's tactics to deal with explicit error bounds, while striving to modify the tactic language as little as possible by limiting bound annotations from the user. As we will see in Section V, our extension can be used to adapt an existing proof to the concrete setting by adding only a few bound annotations.

a) Goals: Concrete global goals are identical to their asymptotic counter-parts except that they use the concrete global logic. On the right, we show a concrete local goal with its global and local contexts, conclusion, and error bound ε. Please remark that local and global hypotheses are displayed together. We present in Listing 5 a concrete logic variation of Listing 1, which is our running example throughout this section. We use this example as support to present the concrete versions of SQUIRREL's tactics later in this section. Here, after calling intro and destruct at 5,6, the goal becomes:

```
G1 : φ̃1
...
Gn : φ̃m
H1 : φ̇1
...
Hm : φ̇m
---------
φ̇ bound ε
```

```
H: [ P ⇒ R <: eP ]
H0: [ ∀̇ x, Q x ⇒ R <: eQ ]
Hor: (P ∨̇ ∃̇ a, Q a)
----------------------
R bound eP + eQ
```

b) Bound space: We use an error term ε, of type real, to bound a probability lying in the interval $[0, 1]$. As explained below, generalizing from $[0, 1]$ to $\mathbb{R}$ reduces the number of proof obligations and allows the exploitation of $\mathbb{R}$'s algebraic structure.

Indeed, bounds are added in forward reasoning steps, for example when combining $[\dot{\phi}_0]_{\varepsilon_0}$ and $[\dot{\phi}_1]_{\varepsilon_1}$ to obtain $[\dot{\phi}_0 \dot{\wedge} \dot{\phi}_1]_{\varepsilon_0+\varepsilon_1}$. The requirement for $\varepsilon_0 + \varepsilon_1$ to belong to $[0, 1]$ can be achieved, e.g., by requiring that $\varepsilon_0 + \varepsilon_1 \leq 1$, or by using a modified addition $+_{\mathsf{m}}$ such that $x +_{\mathsf{m}} y = \min(x + y, 1)$. The former would yield unnecessary proof obligations, while the latter relies on an operator with little algebraic structure (e.g., $(x +_{\mathsf{m}} y) - y$ may be different from x).

Further, backward tactics subtract from the bound rather than adding to it. E.g., the rewrite tactic uses the rule

$$\frac{\Gamma; \Lambda \vdash_{\varepsilon_1 - \varepsilon_2} \dot{\phi}\{s\} \qquad \Gamma \vdash [s = t]_{\varepsilon_2}}{\Gamma; \Lambda \vdash_{\varepsilon_1} \dot{\phi}\{t\}}$$

when rewriting in the conclusion of a local goal. After calling rewrite, the bound ε_1 of the goal has been replaced by $\varepsilon_1 - \varepsilon_2$. Intuitively, this is because the new goal must have a smaller bound to ensure that it can absorb the error ε_2 of the approximated equality being rewritten. Again, requiring that $\varepsilon_1 - \varepsilon_2$ lies in $[0, 1]$ would require a proof obligation or a modified subtraction, which would increase complexity.

Interestingly, most rules are sound without additional checks when using real for bounds. The main exceptions are tactics whose asymptotic counter-parts have no premises, i.e., are closing tactics. This notably includes the assumption and true tactics, respectively based on the rules:

$$\frac{\Gamma; \emptyset \vdash_0 \varepsilon \geq 0 \qquad \dot{\phi} \in \Lambda}{\Gamma; \Lambda \vdash_\varepsilon \dot{\phi}} \qquad \frac{\Gamma; \emptyset \vdash_0 \varepsilon \geq 0}{\Gamma; \Lambda \vdash_\varepsilon \dot{\top}}$$

Thus, the concrete version of assumption only applies when the bound is trivially greater than 0, and true produces a proof obligation requiring to prove that the bound is positive (except when the bound is 0, to avoid circularity issues). Ensuring that bounds are positive is necessary, e.g. even the vacuous-looking judgement $\Gamma; \Lambda \vdash_\varepsilon \dot{\top}$ does not hold when $\varepsilon < 0$, as its semantics requires that $\dot{\top}$ is true with probability greater that $1 - \varepsilon > 1$, which is not possible. In conclusion, having real bounds allows to postpone bound checks to the end of a proof, which significantly reduces the proof burden.

SQUIRREL has built-in reduction rules to discharge basic bound inequalities, while more complex goals can be shown using smt. This tactic has been introduced in [28] for the asymptotic logic. We adapted it to the concrete logic by limiting it to exact reasoning: we only send local hypotheses or global hypotheses of the form $[\dot{\phi}]_0$ to the SMTs, and we restrict smt usage to the exact setting, i.e., to goals where the bound is 0.

Remark 1. *The bounds of [1] are of type* $\overline{\text{real}}$*, that is,* $\mathbb{R} \cup \{+\infty\}$ *equipped with an addition operation* $+$ *for which* $+\infty$ *is absorbing. This structure has interesting algebraic properties (commutativity, associativity), and supports possibly diverging sums of errors, e.g.,* $\sum_{i \in \text{int}}(\varepsilon\ i)$*. But it cannot be extended with subtraction while keeping its algebraic structure, even if adding* $-\infty$*. Our restriction to* real *limits us to finite sums of reals, but allows us to support subtraction, which is crucial for backward reasoning.*

c) Bound annotations: *Global tactics*, i.e., those operating on the global context or conclusion, typically rely on the first-order semantics of the logic and are not specific to the $[\cdot]_\varepsilon$ predicate. Thus, they do not reason on bounds. On the other hand, tactics operating on the local context may modify the bound (though they do not always do so). For many such tactics, the new bound is automatically inferred without requiring bound annotations from the user. This is notably the case for tactics with a single premise. E.g., the rules

$$\frac{\Gamma; \Lambda, \dot{\phi}_0 \vdash_\varepsilon \dot{\phi}_1}{\Gamma; \Lambda \vdash_\varepsilon \dot{\phi}_0 \dot{\Rightarrow} \dot{\phi}_1} \qquad \frac{\Gamma; \Lambda, \dot{\phi}_0, \dot{\phi}_1 \vdash_\varepsilon \dot{\phi}}{\Gamma; \Lambda, \dot{\phi}_1 \dot{\wedge} \dot{\phi}_2 \vdash_\varepsilon \dot{\phi}} \qquad \frac{\Gamma; \Lambda, \dot{\phi}_0 \vdash_{\varepsilon - \varepsilon_0} \dot{\phi}}{\Gamma, [\dot{\phi}_0]_{\varepsilon_0}; \Lambda \vdash_\varepsilon \dot{\phi}}$$

justify, respectively: the local intro tactic (5); calls to destruct on a conjunction in the local context (6,10); and the localize tactic (8,11). (Lines numbers above refer to Listing 5.) Both intro and destruct — on a conjunction — leave the bound unmodified, and localize changes the bound in a goal-directed fashion. Thus, these tactics require no bound annotations. Interestingly, more advanced tactics such as rewrite or apply also require no bound annotations, though they may modify the bound: e.g., rewrite sums-up the errors induced by a sequence of rewritings automatically.

Local tactics that create multiple subgoals often need to distribute the error mass. E.g., the local versions of split and case (based on rules R-$\dot{\wedge}$ and L-$\dot{\vee}$) are backward tactics. To apply them, the user decomposes the bound ε of the current goal into ε_1 and ε_2, such that $\varepsilon = \varepsilon_1 + \varepsilon_2$, using the syntax

$$\text{tac args} <: \varepsilon_1, \ldots, \varepsilon_n$$

where args are standard arguments of the tactic, and the $(\varepsilon_i)_i$ are the *bound annotations*. If the asymptotic version of (tac args) creates n subgoals, we use bound annotation ε_i for the i-th subgoal, and ask for a proof that $\varepsilon_1 + \ldots \varepsilon_n \leq \varepsilon$. The user may omit the last annotation ε_n, which is then set to $\varepsilon - (\varepsilon_1 + \cdots + \varepsilon_{n-1})$. In that case, no additional proof obligation is required. For example, after the call to case at 7, we have the two goals:

```
H: [ P ⇛ R <: eP ]
H0: [ ∀̇ x, Q x ⇛ R <: eQ ]
Hor: P
----------------------
R bound eP
```

```
H: [ P ⇛ R <: eP ]
H0: [ ∀̇ x, Q x ⇛ R <: eQ ]
Hor: ∃̇ a, Q a
----------------------
R bound (eP + eQ) − eP
```

d) Introduction pattern: As seen in Section II-A, introduction patterns are a powerful tool allowing for concise proof context management. Here, we focus on *and/or patterns*, and omit other features that are not impacted by the addition of error bounds. We extend SQUIRREL's introduction patterns with concise bound management. As explained earlier, when operating on the proof context, dealing with conjunctions or global disjunctions does not require any bound annotation, while local disjunctions do. Thus, we use the following syntax for bound-annotated introduction patterns:

$$\text{ip}_\text{s} ::= \text{ip} \mid \text{id} \qquad \text{ip} ::= [\text{ip}_\text{s} \ldots \text{ip}_\text{s}] \mid [\text{ip}_\text{s} \boxed{<: \varepsilon_1}^? \mid \cdots \mid \text{ip}_\text{s} \boxed{<: \varepsilon_n}^?]$$

where $\boxed{<: \varepsilon}^?$ indicates that the bound annotation ε is optional, and id is an identifier used, e.g., to introduce an hypothesis in the proof context with a fresh name id.

The *and pattern* [H$_1$ ... H$_n$] decomposes a conjunction $\wedge_{i \leq n} \phi_i$ and adds the hypotheses $(\text{H}_i : \phi_i)_{i \leq n}$ to the proof context. The *or pattern* [H$_1$ | ... | H$_n$] decomposes a disjunction $\vee_{i \leq n} \phi_i$, producing one goal per disjunction, each introduced

$$\begin{aligned}\mathsf{pt} ::= {} & \mathsf{id} \mid \mathsf{pt} \mathbin{\textcircled{\scriptsize @}} t \mid \mathsf{pt} \mathbin{\textcircled{\scriptsize @}} \mathsf{pt} \mid \mathsf{pt} \mathbin{\textcircled{\scriptsize @}} \_ \\ & \mid \overset{(\diamond)}{\lambda}(x : \tau).\ \mathsf{pt} \mid \tilde{\Pi}(\mathsf{id} : \tilde{\phi}).\ \mathsf{pt} \mid \dot{\Pi}(\mathsf{id} : \dot{\phi}).\ \mathsf{pt} \\ & \mid \%\ \mathsf{pt} \mid \%?\ \mathsf{pt} \mid [\mathsf{pt}] \mid (\mathsf{pt} \lessdot \varepsilon(\mathsf{pt})) \mid (\mathsf{pt} \lessdot \varepsilon(\_))\end{aligned}$$

We let $\diamond \in \{\cdot, \sim\}$. The notation $(\diamond)$ indicates that the annotation $\diamond$ can be omitted in a surface-level proof term pt_s. An annotated proof term $\overline{\mathsf{pt}}$ must not omit any annotations, must not use holes $\_$ nor the may localize construct $\%? \cdot$.

Figure 2: Syntax of annotated and surface-level proof terms.

as $\mathsf{H}_i : \phi_i$. If the disjunction is local, a bound annotation is required to split the error mass, as for the case tactic. Patterns can be nested to provide a subpattern which is recursively applied to the introduced hypothesis. Listing 6 presents an example of a bound-annotated pattern which splits $(\mathsf{P} \dot{\vee} \dot{\exists}\ \mathsf{a}, \mathsf{Q}\ \mathsf{a})$ and assigns the errors $\mathsf{e_P}$ to P and $\mathsf{e_Q}$ to $(\dot{\exists}\ \mathsf{a}, \mathsf{Q}\ \mathsf{a})$. Compared to the asymptotic version of Listing 4, we have an additional bound inequality, which auto easily proves.

IV. Proof Term Language and Elaborator

We first present our proof term language and typing rules in Section IV-A. Then, we describe the design goals of our elaborator in Section IV-B, before presenting it in Section IV-C. Finally, we study its theoretical properties in Section IV-D.

A. Annotated Proof Terms

We start by describing a proof term language for standard higher-order logic, which we then adapt to our two-level logic.

$$\mathsf{pt_h} ::= \mathsf{id} \mid \mathsf{pt_h}\ t \mid \mathsf{pt_h}\ \mathsf{pt_h} \mid \lambda(x : \tau).\ \mathsf{pt_h} \mid \Pi(\mathsf{id} : \phi).\ \mathsf{pt_h}$$

Proof terms allow to derive formulas which are valid in a given context $\mathbb{E}, \Theta$ (here, Θ refers to a context in standard higher-order logic). This is captured by a typing relation $\mathbb{E}; \Theta \vdash \mathsf{pt_h} : \phi$, which means that $\mathsf{pt_h}$ *proves* ϕ in the context $\mathbb{E}; \Theta$. We informally describe the syntax of proof terms and their typing rules, which closely match the proof rules of higher-order logic. An identifier id refers to an hypothesis $\mathsf{id} : \phi$ in Θ, and proves ϕ. If $\mathsf{pt_h}$ proves $\forall x.\ \phi$, then the application $(\mathsf{pt_h}\ t)$ proves $\phi\{x \mapsto t\}$. If $\mathsf{pt_h^0}$ proves $\phi \Rightarrow \psi$ and $\mathsf{pt_h^1}$ proves ϕ, then the application $(\mathsf{pt_h^0}\ \mathsf{pt_h^1})$ proves ψ. The abstraction $\lambda(x : \tau).\ \mathsf{pt_h}$ proves $\forall(x : \tau).\ \phi$ if $\mathsf{pt_h}$ proves ϕ in $\mathbb{E}$ extended with $(x : \tau)$; and the product $\Pi(\mathsf{id} : \phi).\ \mathsf{pt_h}$ proves $\phi \Rightarrow \psi$ if $\mathsf{pt_h}$ proves ψ in Θ extended with $(\mathsf{id} : \phi)$.

We now extend this proof term language to our two-level logic. We present in Fig. 2 the syntax of annotated proof terms, which we describe below. An annotated proof term $\overline{\mathsf{pt}}$ is typed in a context $\mathbb{E}; \Gamma$ to prove a global formula, and in a context $\mathbb{E}; \Gamma; \Lambda$ to prove a local formula with a bound:

$$\mathbb{E}; \Gamma \vdash \overline{\mathsf{pt}} : \tilde{\phi} \qquad \mathbb{E}; \Gamma; \Lambda \vdash \overline{\mathsf{pt}} : (\dot{\phi} \lessdot \varepsilon)$$

These typing judgements reflect the form of our logical judgements: e.g., the right typing judgement corresponds to $\mathbb{E}; \Gamma; \Lambda \vdash_\varepsilon \dot{\phi}$. The logic has a global and local variant of each construct. E.g., for the local logic, $\mathsf{pt_h} \mathbin{\dot{@}} t$ and $\mathsf{pt_h} \mathbin{\dot{@}} \mathsf{pt_h}$ correspond to the application of $\mathsf{pt_h}$ to, respectively, a term and a proof term. We use $\dot{\_}$ to make the space visible to depict it as local. Further, there are additional constructs to move formulas between logical levels. If $\overline{\mathsf{pt}}$ proves $[\dot{\phi}]_\varepsilon$, then $(\%\ \overline{\mathsf{pt}})$ opens the $[\cdot]$ modality and proves $(\dot{\phi} \lessdot \varepsilon)$. Note that $(\%?\ \overline{\mathsf{pt}})$ behaves similarly, except that it does not fail if $\overline{\mathsf{pt}}$ proves a local formula. If $\overline{\mathsf{pt}}$ proves $(\dot{\phi} \lessdot \varepsilon)$, then $[\overline{\mathsf{pt}}]$ introduces the $[\cdot]$ modality and proves $[\dot{\phi}]_\varepsilon$. Crucially, the latter is only allowed if $\overline{\mathsf{pt}}$ does not *use local hypotheses*, as it relies on the ByGlob rule (which requires the local context to be empty). Finally, $(\overline{\mathsf{pt}} \lessdot \varepsilon(\overline{\mathsf{pt}}'))$ proves $(\dot{\phi} \lessdot \varepsilon)$ if $\overline{\mathsf{pt}}$ proves $(\dot{\phi} \lessdot \varepsilon_0)$ and $\overline{\mathsf{pt}}'$ proves $[\varepsilon_0 \leq \varepsilon]_0$. See details and typing rules in Appendix A.

Example 2. *In the left context below,* $\mathsf{L} \mathbin{\dot{@}} (\%\ \mathsf{H})$ *proves* $(\psi \lessdot \varepsilon)$. *In the right context,* $\mathsf{H}_1 \mathbin{\dot{@}} [(\%\ \mathsf{H}_2 \lessdot \varepsilon_1(\mathsf{B}))]$ *proves* $[\psi]_\varepsilon$.

$$\begin{array}{ll} \mathsf{H} : [\dot{\phi}]_\varepsilon & \mathsf{B} : [\varepsilon_2 \leq \varepsilon_1]_0 \\ \mathsf{L} : \dot{\phi} \mathrel{\dot{\Rightarrow}} \psi & \mathsf{H}_1 : [\dot{\phi}]_{\varepsilon_1} \mathrel{\dot{\Rightarrow}} [\psi]_\varepsilon \\ & \mathsf{H}_2 : [\dot{\phi}]_{\varepsilon_2} \end{array}$$

B. Elaborator Design

An elaborator is tasked with completing a partial surface-level proof term into an annotated well-typed proof term, by adding missing constructs to the user-provided surface-level proof term. Below, we describe the design goals of our elaborator, as well as its high-level architecture. We leave the formal description of our procedure to Section IV-C.

Design goals: As we see in Example 2, annotated proof terms can be verbose, as they require to manually deal with locality and weakenings. Clearly, an elaborator could be useful here. E.g., the proof term $\mathsf{L} \mathbin{\dot{@}} (\%\ \mathsf{H})$ could be shortened as $\mathsf{L} \mathbin{\dot{\_}} \mathsf{H}$, since the localization of H and the local annotation on the application $\dot{\_}$ can be inferred from the fact that L is local. Similarly, $\mathsf{H}_1 \mathbin{\dot{@}} [(\%\ \mathsf{H}_2 \lessdot \varepsilon_1(\mathsf{B}))]$ could be shortened as $\mathsf{H}_1 \mathbin{\dot{\_}} \mathsf{H}_2$, by automatically inferring that the bound of $[\dot{\phi}]_{\varepsilon_2}$ must be weakened to ε_1 so as to prove the assumption of H_1. Further, the weakening proof obligation $[\varepsilon_2 \leq \varepsilon_1]_0$ could be discharged to the user, to be proved later.

In summary, we want our elaborator to automatically infer: locality annotations such as $\dot{@}$ and $\dot{\lambda}$. ; localizations $\%\cdot$ and globalizations $[\cdot]$; bounds weakenings $(\cdot \lessdot \cdot)$.

Elaborator architecture: A surface-level proof term $\mathsf{pt_s}$ is a proof term in which annotations may be omitted (see definition in Fig. 2). We use

$$\phi ::= \tilde{\phi} \mid (\dot{\phi} \lessdot \varepsilon).$$

to denote the possible outputs of an elaboration. Due to the two-level nature of our logic, we need to track the local hypotheses used in a proof term. Indeed, this is key to support globalization, as $\mathbb{E}; \Gamma; \Lambda \vdash_\varepsilon \dot{\phi}$ is equivalent to $\mathbb{E}; \Gamma \vdash [\dot{\phi}]_\varepsilon$ iff Λ is empty. To that end, we introduce the notion of *footprint*, which is the set of *local* hypotheses identifiers used in an annotated proof term. More precisely, let $I = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ denote the set of identifiers, and $\mathsf{filter}(\Lambda\ ;\ I)$ be the restriction of Λ to hypotheses $(\mathsf{id} : \dot{\phi})$ with $\mathsf{id} \in I$. Then,

$$\mathbb{E}; \Gamma; \Lambda \vdash_I \overline{\mathsf{pt}} : (\dot{\phi} \lessdot \varepsilon)$$

is valid if $\mathbb{E}; \Gamma; \mathsf{filter}(\Lambda\ ;\ I) \vdash \overline{\mathsf{pt}} : (\dot{\phi} \lessdot \varepsilon)$ is well-typed. Appendix A presents a similar notion for global judgements.

Discussion: We focus here on aspects related to bounds manipulations and locality, and thus only consider a core language with the necessary related features. Adding more constructs to our language, e.g., to deal with conjunctions and disjunctions, would be a standard orthogonal extension.

The formal presentation of our elaborator serves as a theoretical justification for the elaborator we implemented in our extension of SQUIRREL [26]. Please note that our theoretical presentation supports a richer proof term language than the implementation. E.g., while our implementation supports key constructs such as localization $\%? \cdot$ and weakening $(\cdot \lessdot \cdot)$, there is currently no support for λ and Π. Adding the missing features would be straightforward, as the theory already supports them. Conversely, there are some orthogonal features that are implemented but not part of the theoretical presentation (e.g., argument inference by unification).

C. Elaboration Procedure

The elaborator goal is to infer any missing bound and locality information in a surface-level proof term, in order to get an annotated well-typed proof term. The elaborator design is inspired by bi-directional typing [29], with an *inference mode* that builds proof terms from the top down, and a *checking mode* — called *directed elaboration* — that attempts to build a proof term matching a pattern P needed by the inference mode. Directed elaboration is achieved through an elaboration step followed by a *re-elaboration* step that attempts to modify the elaborated proof term to match P.

Elaboration takes as input a surface-level proof term, and produces an annotated proof term, its footprint, and the proven typing judgement. It is written

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt}_s \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}} :_I \phi.$$

The annotated proof term $\overline{\mathsf{pt}}$ returned by the elaboration may be modified latter through re-elaboration. An elaboration is valid if the returned proof term $\overline{\mathsf{pt}}$ is well typed, i.e., if $\mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash_I \overline{\mathsf{pt}} : \phi$.

Directed elaboration is called by the elaboration rules on a surface-level proof term pt_s, and tries to elaborate it into an annotated proof term $\overline{\mathsf{pt}}$ proving ϕ, with the constraint that ϕ should *match* a target pattern P, which is a local or global formula with holes. E.g., using $[\_]_\varepsilon$ allows to target any formula $[\dot{\phi}]_\varepsilon$, for any $\dot{\phi}$. Directed elaboration relies on elaboration to obtain a first annotated proof term $\overline{\mathsf{pt}}_0$, which is further refined using *re-elaboration* (see below). Directed elaboration is written

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash (\mathsf{pt}_s \triangleleft P) \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}} :_I \phi.$$

It is valid when its conclusion is well typed and ϕ matches P.

Re-elaboration rewrites an elaborated proof term in order to match the target pattern. It is written

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \overline{\mathsf{pt}}_0 :_{I_0} \phi_0 \triangleleft P \twoheadrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1$$

where $\mathbb{E}; \Gamma; \Lambda \vdash_{I_0} \overline{\mathsf{pt}}_0 : \phi_0$ is expected to be well typed. It is valid if it produces a proof term $\overline{\mathsf{pt}}_1$ such that $\mathbb{E}; \Gamma; \Lambda \vdash_{I_1} \overline{\mathsf{pt}}_1 : \phi_1$ is well typed and ϕ_1 matches P. Roughly, re-elaboration adds proof obligations and performs valid structural rewriting of proof terms. For example, it may add weakening and the corresponding inequality hypothesis in the context if the pattern forces a specific error bound.

Rules: We present a selected number of elaboration and re-elaboration rules in Fig. 3, and quickly describe them below. Our full set of rules can be found in Appendix B.

Elaboration rules closely match typing rules for the annotated proof terms, calling *directed elaboration* to get premises of the correct form and keeping track of footprints.

- When using an hypothesis identifier id, we add id to the footprint only if it refers to a local hypothesis (see rules E.L.AX and E.G.AX). Rule E.BYLOC elaborates $[\mathsf{pt}]$ to a global formula, provided that pt can be elaborated towards into a local formula without footprint (and a matching ε). Rule E.$\dot{\Pi}$ introduces a local implication, and removes the introduced hypothesis from the footprint.
- Rule E.I.L.$\_.\_$ discharges the proof obligation for a local formula as $\mathsf{H} : \dot{\phi}_0$, adding it to the hypotheses and the footprint. Rule E.I.$\therefore$ corresponds to a local proof cut, and sums the errors coming from both subproofs.

We now describe *re-elaboration rules*.

- Rule RE.CONV returns the elaborated proof term if the elaborated formula already matches the pattern.
- When looking for an error bound ε_0 with an elaborated formula with error bound ε, the rule RE.WEAK discharges the proof that $\varepsilon \leq \varepsilon_0$ and adds a weakening.
- Rule RE.$[\cdot]$.$\dot{\Rightarrow}$ allows for discharging global hypotheses when looking for an atomic global formula.

Example 3. *Assume we have two hypotheses* $\mathsf{H}_0 : [\dot{\phi}]_0 \dot{\Rightarrow} [\dot{\phi} \dot{\Rightarrow} \dot{\phi}_0]_0$ *and* $\mathsf{H}_1 : \dot{\phi}$. *The proof term* $(\mathsf{H}_0\ \mathsf{H}_1)$ *elaborates to* $[\dot{\phi}]_0; \emptyset \vdash_0 \dot{\phi}_0$, *because the directed elaboration will respect the locality of the argument* H_1 *and use it to discharge the local assumption in* H_0. *On the other hand,* $(\mathsf{H}_0\ [\mathsf{H}_1])$ *elaborates to* $\emptyset; \emptyset \vdash [\dot{\phi} \dot{\Rightarrow} \dot{\phi}_0]_0 0$, *since* $[\mathsf{H}_1]$ *is global and thus discharges the global assumption in* H_0.

D. Theoretical Study

We now study the properties of our elaborator, which serve as theoretical justifications for its capacities. After some basic results related to its soundness and completeness w.r.t. our proof terms typing rules, we present two usability results. First, we show that our elaborator is predictable. Second, we present our main erasability theorem, which intuitively states that the elaborator can automatically infer weakening inside some proof terms, except for a single top-level weakening.

Basic properties: Our elaboration procedure is deterministic, terminates and only produces valid judgements. These are straightforward properties of the rules.

The elaborator is complete w.r.t. annotated proof terms, in the sense that any well-typed annotated proof term can be obtained as a result of an elaboration — otherwise, some proof

Elaboration rules

$$\textsc{E.L.Ax}\quad \frac{\Delta_0 = \mathbb{E}_0;\Gamma_0;\Lambda_0 \qquad (\mathsf{id},\dot\phi)\in\Lambda_0}{\Delta_0 \vdash \mathsf{id} \rightsquigarrow \Delta_0 \vdash \mathsf{id} :_{\{\mathsf{id}\}} (\dot\phi \lessdot 0)}$$

$$\textsc{E.G.Ax}\quad \frac{\Delta_0 = \mathbb{E}_0;\Gamma_0;\Lambda_0 \qquad (\mathsf{id},\tilde\phi)\in\Gamma_0}{\Delta_0 \vdash \mathsf{id} \rightsquigarrow \Delta_0 \vdash \mathsf{id} :_{\emptyset} \tilde\phi}$$

$$\textsc{E.ByLoc}\quad \frac{\begin{array}{c}\Delta_0 \vdash (\mathsf{pt} \lhd (\_ \lessdot \_))\\ \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_{\emptyset} (\dot\phi \lessdot \varepsilon)\end{array}}{\Delta_0 \vdash [\mathsf{pt}] \rightsquigarrow \Delta_1 \vdash [\overline{\mathsf{pt}}] :_{\emptyset} [\dot\phi]_\varepsilon}$$

$$\textsc{E.I.}\dot\sqcup\quad \frac{\begin{array}{c}\Delta_0 \vdash (\mathsf{pt}_0 \lhd (\_ \lessdot \_)) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_0 :_{I_0} (\dot\phi_0 \lessdot \varepsilon_0)\\ \Delta_1 \vdash (\mathsf{pt}_1 \lhd (\dot\phi_0 \dot\Rightarrow \_ \lessdot \_)) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_1 :_{I_1} (\dot\phi_0 \dot\Rightarrow \dot\phi \lessdot \varepsilon_1)\end{array}}{\Delta_0 \vdash \mathsf{pt}_1 \dot\sqcup \mathsf{pt}_0 \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_1 \dot\sqcup \overline{\mathsf{pt}}_0 :_{I_0\cup I_1} (\dot\phi \lessdot \varepsilon_0+\varepsilon_1)}$$

$$\textsc{E.}\dot\Pi\quad \frac{\mathbb{E}_0;\Gamma_0;\Lambda_0,(\mathsf{id}:\dot\phi) \vdash (\mathsf{pt} \lhd (\_ \lessdot \_)) \rightsquigarrow \mathbb{E}_1;\Gamma_1;\Lambda_1 \vdash \overline{\mathsf{pt}} :_I (\dot\phi_1 \lessdot \varepsilon)}{\begin{array}{c}\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash \dot\Pi(\mathsf{id}:\dot\phi_0).\ \mathsf{pt}\\ \rightsquigarrow \mathbb{E}_1;\Gamma_1;\Lambda_1 \vdash \dot\Pi(\mathsf{id}:\dot\phi_0).\ \overline{\mathsf{pt}} :_{I\setminus\{\mathsf{id}\}} (\dot\phi_0 \dot\Rightarrow \dot\phi_1 \lessdot \varepsilon)\end{array}}$$

$$\textsc{E.I.L.}\_\_\quad \frac{\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash \mathsf{pt} \rightsquigarrow \mathbb{E}_1;\Gamma_1;\Lambda_1 \vdash \overline{\mathsf{pt}} :_I (\dot\phi_0 \dot\Rightarrow \dot\phi \lessdot \varepsilon)}{\begin{array}{c}\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash \mathsf{pt} \_\ \_\\ \rightsquigarrow \mathbb{E}_1;\Gamma_1;\Lambda_1,(H:\dot\phi_0) \vdash \overline{\mathsf{pt}} \dot\sqcup H :_{I\cup H} (\dot\phi \lessdot \varepsilon)\end{array}}$$

Directed elaboration and re-elaboration rules

$$\textsc{DE.RE}\quad \frac{\begin{array}{c}\Delta_0 \vdash \mathsf{pt} \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1\\ \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1 \lhd P \twoheadrightarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2\end{array}}{\Delta_0 \vdash (\mathsf{pt} \lhd P) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2}$$

$$\textsc{RE.Conv}\quad \frac{\mathbb{E}_0 \vdash \phi \overset{?}{\leq} P}{\Delta_0 \vdash \overline{\mathsf{pt}} :_I \phi \lhd P \twoheadrightarrow \Delta_0 \vdash \overline{\mathsf{pt}} :_I \phi}$$

$$\textsc{RE.Weak}\quad \frac{\mathbb{E}_0 \vdash \dot\phi \overset{?}{\leq} \dot\phi_0 \qquad \mathbb{E}_0 \vdash \varepsilon \overset{?}{\not\leq} \varepsilon_0}{\begin{array}{c}\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash \overline{\mathsf{pt}} :_I (\dot\phi \lessdot \varepsilon) \lhd (\dot\phi_0 \lessdot \varepsilon_0)\\ \twoheadrightarrow \mathbb{E}_0;\Gamma_0,(H:[\varepsilon\leq\varepsilon_0]_0);\Lambda_0 \vdash (\overline{\mathsf{pt}} \lessdot \varepsilon_0(H)) :_I (\dot\phi \lessdot \varepsilon_0)\end{array}}$$

$$\textsc{RE.}[\cdot].\dot\Rightarrow\quad \frac{\begin{array}{c}\mathbb{E}_0;\Gamma_0,(H:\tilde\phi_0);\Lambda_0 \vdash \overline{\mathsf{pt}}_0 \dot\sqcup H :_{\emptyset} \tilde\phi_1 \lhd [\dot\phi_0]_{\varepsilon_0}\\ \twoheadrightarrow \mathbb{E}_1;\Gamma_1;\Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_{\emptyset} [\dot\phi_1]_{\varepsilon_1}\end{array}}{\begin{array}{c}\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash \overline{\mathsf{pt}}_0 :_{\emptyset} \tilde\phi_0 \dot\Rightarrow \tilde\phi_1 \lhd [\dot\phi_0]_{\varepsilon_0}\\ \twoheadrightarrow \mathbb{E}_1;\Gamma_1;\Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_{\emptyset} [\dot\phi_1]_{\varepsilon_1}\end{array}}$$

We use the shorthand $\Delta = \mathbb{E};\Gamma;\Lambda$ in rules that do not manipulate the context.

Figure 3: Elaboration and re-elaboration rules.

terms would be out-of-reach for the user, which would be limiting. This property easily follows from the fact that a well-typed annotated proof term is elaborated into itself, and that annotated proof terms are a subset of surface-level proof terms. More precisely, if $\overline{\mathsf{pt}}$ is such that $\mathbb{E};\Gamma;\Lambda \vdash \overline{\mathsf{pt}} : \dot\phi$ is derivable, then there exist pt_s and I such that

$$\mathbb{E};\Gamma;\Lambda \vdash \mathsf{pt}_s \rightsquigarrow \mathbb{E};\Gamma;\Lambda \vdash \overline{\mathsf{pt}} :_I \dot\phi$$

We have a similar result when typing $\overline{\mathsf{pt}}$ in a global context.

Predictability: Elaboration should be *predictable*, i.e., it should only add constructs to the surface-level proof terms, and not produce a proof out of thin air. More precisely, if

$$\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash \mathsf{pt} \rightsquigarrow \mathbb{E}_1;\Gamma_1;\Lambda_1 \vdash \overline{\mathsf{pt}} :_I \phi$$

we show that pt is a minor of $\overline{\mathsf{pt}}$, i.e., pt can be obtained by removing constructs and annotations from $\overline{\mathsf{pt}}$. This follows from the fact that our rules only enrich proof terms by adding weakenings, discharges and locality information.

Erasability: We want the elaborator to allow for removing weakenings from a proof term. To capture this idea, we introduce a relation $\xrightarrow{\spadesuit}$ between surface-level proof-terms, which is the rewrite relation

$$(\mathsf{pt}_0 \lessdot \varepsilon(\mathsf{pt}_1)) \xrightarrow{\spadesuit} \%?\,\mathsf{pt}_0$$

Note that the result of the erasure of a weakening might need to be localized using $\%?\,\mathsf{pt}_0$, as it could result in a global formula (since our surface-level proof term might omit the localization step, letting the elaborator recover it).

Ideally, we would like to show that if some pt_a is such that

$$\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash \mathsf{pt}_a \rightsquigarrow \mathbb{E}_1;\Gamma_1;\Lambda_1 \vdash \overline{\mathsf{pt}} :_I \phi$$

is derivable, then for any pt_b such that $\mathsf{pt}_a \xrightarrow{\spadesuit} \mathsf{pt}_b$,

$$\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash \mathsf{pt}_b \rightsquigarrow \mathbb{E}_1;\Gamma_1;\Lambda_1 \vdash \overline{\mathsf{pt}} :_I \phi$$

is derivable, i.e., we recover the elaboration $\overline{\mathsf{pt}}$ and conclusion ϕ. Unfortunately, this does not hold, as shown below.

Example 4. *Consider the scenario where $\mathsf{pt}_a = \overline{\mathsf{pt}}$ is already well typed and is the proof term $(\overline{\mathsf{pt}}_0 \lessdot \varepsilon_1(\mathsf{pt}_i))$. Then, if $\overline{\mathsf{pt}}_0$ proves $(\dot\phi \lessdot \varepsilon_0)$ and pt_i proves $[\varepsilon_0 \leq \varepsilon_1]_0$, we have that pt_a proves $\phi = (\dot\phi \lessdot \varepsilon_1)$. Erasing the top-level weakening in pt_a yields $\mathsf{pt}_b = \overline{\mathsf{pt}}_0$, which is also well typed and elaborates into itself but with a different conclusion $(\dot\phi \lessdot \varepsilon_0)$. Thus,*

1) *erasing a top-level weakening may result in a formula with a smaller (i.e., stronger) error bound;*
2) *when we erase a weakening, we lose the subproof $[\varepsilon_0 \leq \varepsilon_1]_0$ establishing that the weakening is valid.*

Design	Setting	LoC	Δ
YubiKey	Asym./Concrete	485/488	1
Birthday Bound	Exact/Concrete	120/141	30
Merkle Trees	Exact/Concrete	274/359	41

Δ indicates the number of differing LoCs in the proof scripts only, excluding declarations and comments.

Table I: Overview of our case-studies.

Note that the change of bounds coming from the erasure of an *inner* weakening could be recovered by the elaborator, as it may automatically re-insert a weakening to match a target bound (c.f. rule RE.WEAK). Nonetheless, the subproof of the weakening inequality is still lost.

In the scenario of Example 4, the distance between the elaboration of pt_a and its erasure pt_b is acceptable, since pt_b will yield a stronger bound, and since the inequality proof $[\varepsilon_0 \leq \varepsilon_1]_0$ can be recovered later through $\overline{\mathsf{pt}}_i$. Actually, we will show that this is always the case, and that erasing weakenings only yield stronger elaborations.

We capture this idea by introducing a relation $\Rrightarrow$, which will be used to relate the two judgements obtained from the elaboration of a proof term and its erasure. This relation, defined in Fig. 12 of the appendices, is such that

$$\mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \phi_2 \Rrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \phi_1$$

holds when the context Γ_2, Λ_2 extends Γ_1, Λ_1 with discharged inequalities stemming from erasures, and the error bounds in ϕ_1 are greater than the ones in ϕ_2. Intuitively, this means that if all discharged hypotheses in Γ_2, Λ_2 are proved, then ϕ_2 is more precise than ϕ_1. Because dealing with quantifiers introduce difficulties, we restrict our result to proof terms without term applications and λs. We are now ready to state our erasure theorem, whose proof can be found in Appendix E.

Theorem 1 (Erasability)**.** *If* pt_s *is a proof term such that* $\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt}_s \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_I \phi_1$ *is derivable and* pt_s *contains no term applications and* λ*s, then for any* pt *such that* $\mathsf{pt}_s \xrightarrow{\spadesuit} \mathsf{pt}$ *we have*

1) $\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt} \rightsquigarrow \mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \overline{\mathsf{pt}}_2 :_I \phi_2$ *is derivable;*
2) $\mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \phi_2 \Rrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \phi_1$.

V. Case Studies

We apply our framework on three case studies, chosen to serve two purposes. First, they allow us to validate the usefulness and applicability of our approach. Second, they allow to evaluate the impact of moving a proof to a concrete setting from an asymptotic or exact one. In other words, to evaluate the cost of using our approximated logic to compute precise error bounds compared to qualitative approaches.

Case-studies: These three case-studies are: the YubiKey protocol, the Birthday bound, and Merkle trees. In each case, we apply the following methodology. We first start from an exact or asymptotic formalization: for YubiKey, we use the development from [15]; for the Birthday bound and Merkle trees, we write it ourselves. Then, we adapt these proofs to a concrete setting, thus obtaining precise error bounds. SQUIRREL's developments are available online [26], and are summarized in Table I. We discuss them in details in Section V-A and Section V-B.

Summary of findings: We observe that extending an existing proof with concrete error bounds leads to a small or moderate proof overhead, and adds little complexity. We identify two main sources of additional work when adapting a formalization. First, concrete proofs require to reason on the bounds. In our examples, this is mostly lightweight reasoning that we discharge using smt (in a single occurrence, for the birthday bound, we had to write a manual $\approx$ 10 LoC proof by hand). Second, in our case-studies, we perform proofs by induction on an element x of an inductive type upon which the bound depends. When doing an induction at the local level, the bound cannot depend on x. Thus, when switching to a concrete setting, we have to use global-level induction, which requires a moderate amount of additional book-keeping.

A. YubiKey and Birthday Bound

a) YubiKey: The YubiKey protocol (as described in [30]) is a counter-based authentication protocol. In [15], a SQUIRREL analysis showed, in the asymptotic model: i) the absence of replay attacks; ii) monotonicity of the counter; iii) an injective authentication property.

We adapt this security analysis to a concrete setting. The first two properties do not rely on cryptographic assumptions, and thus hold exactly without any change to the proof script. The last property relies on the INT-CTXT cryptographic assumption, which SQUIRREL supports through a built-in tactic intctxt. We adapt intctxt to a concrete setting to introduce an additional error reflecting the adversary's advantage in breaking the assumption. The proof requires minimal adaptation: we add the INT-CTXT advantage to the lemma, and a single tactic for bound reasoning. Note that before adapting the proof, we changed a few lines of the original proof that used deprecated tactics (that we did not adapt to a concrete setting).

b) Birthday Bound: The birthday bound used in cryptography states that the probability for a collision to exist in a list of n independently sampled value is bounded by $\frac{n(n-1)}{2} \times \mathsf{p_{col}}$, where $\mathsf{p_{col}}$ bounds collision probability between two elements.

To formalize this in SQUIRREL, we consider an array **name** n: lval → lval of values sampled independently at random using the special construct **name**. Then, we prove that

NoDuplicates L ⇛ [no_collision L <: (|L| * (|L| – 1)) / 2 * $\mathsf{p_{col}}$],

where L is a list of distinct positions in the array (thanks to the NoDuplicates L assumption).

We proceed in two steps. First, we prove a variant in which we replace the name n by a perfect surjective function. In this variant, the property above holds with error 0. Then, we prove the birthday bound by adapting the previous proof.

B. Merkle Trees

A Merkle tree [4] is a secure data structure allowing to represent finite sets with efficient proofs of memberships.

```
(* A finite type indexing leaves of the Merkle tree *)
type lval[finite].
inductive tree =
  | Leaf : lval → tree
  | Node : tree → tree → tree.

(* A cryptographic hash function *)
op fhash : message → message.

(* Recursive computation of the hash of a tree *)
let rec hash_tree (t : tree) = fhash (hash_tree0 t)
and hash_tree0 t with
  | Leaf m → encode_leaf m
  | Node l r → encode_node (hash_tree l, hash_tree r).

inductive side = 0 : side | 1 : side.
inductive proof =
  | Emp : proof
  | Cons : side → message → proof → proof.

(* If p proves that a tree tsub with hash hsub is a
   sub-tree of t, computes hash_tree t. *)
let rec hash_path (hsub : message) (p : proof) : message  with
  | Emp → hsub
  | Cons i h_{i-1} psub →
    let h_i = hash_path hsub psub in
    fhash (encode_node (h_0, h_1)) k.

(* Checks that p proves that tsub (with hash hsub) is a sub-tree
   of a tree t (with hash h). *)
let verify (hsub : message) (p : proof) (h : message) : bool =
  hash_path hsub p = h.

(* Descend in t following the proof p, and retrieve the corresponding sub-tree. *)
let rec lookup ( (p,t) : proof * tree) : tree with
  | (Emp, _) → t
  | (Cons i _ psub, Node t_0 t_1) → lookup (psub, t_i)
  | (Cons _ _ _, Leaf _) → witness.        (* lookup failed, return default value *)
```

For the sake of conciseness, we omit some details such as termination annotations for recursive functions, or the definitions of the encoding functions. The symbol witness_τ denotes an arbitrary value of type τ, and we omit τ when it is clear from context.

Figure 4: A Merkle hash tree implementation in SQUIRREL.

a) Merkle-trees: We present our SQUIRREL formalization in Fig. 4. We consider an abstract type lval representing the values stored in a Merkle tree. A Merkle tree, of type tree, is either a leaf (Leaf d) storing a value d of type lval, or a binary node (Node l r) where l and r are two Merkle trees. A Merkle tree t can be efficiently and concisely characterized by its hash (hash_tree t), which is computed recursively over the structure of the tree by applying a cryptographic hash function fhash on each leaf or node of the tree: for a leaf (Leaf d), it hashes the data d using fhash; for a node (Node l r), it applies fhash to the pair of the hashes of the subtrees l and r. Finally, we use non-ambiguous encoding functions.

b) Membership proofs: In a membership proof, a prover must prove to a verifier that a tree t_{sub} is a subtree of another tree t. This proof only relies on the hashes of t_{sub} and t, and a proof of linear size in the length of the path from the root of t to t_{sub}. The verifier does not need to know the trees t_{sub} and t, which allows for efficient proofs of membership. Fig. 5 also formalizes membership proofs. A proof of type proof is either an empty proof Emp if $t_{sub} = t$; or an element Cons i h_{i-1} p_i if t = Node t_0 t_1, where the sub-proof p_i shows that t_{sub} is a subtree of t_i, and h_{i-1} is the hash of the other subtree t_{1-i}.

The hash of a tree can be recomputed from the hash of a subtree and a valid membership proof for that subtree using the function hash_path in Fig. 4. Then, (verify hsub p h) checks that p proves that t_{sub} (of hash hsub) is a subtree of t (of hash h), by checking that the computation of the hash of t using the membership proof p gives the expected hash h.

c) Security property: Our main security lemma

$$\dot\forall\ (p : \text{proof})\ (t : \text{tree})\ (t_{sub} : \text{tree}),\ \text{verify}\ (\text{hash\_tree}\ t_{sub})\ p\ (\text{hash\_tree}\ t) \dot\Rightarrow \text{lookup}\ (p,t) = t_{sub}.$$

states that if a proof p proves that t_{sub} is a subtree of t, then following the path p in t indeed gives the expected subtree t_{sub}. This property is unachievable under realistic assumptions, as it does not allow for any possibility of failure. Still, we present a first proof assuming a *perfect injective hash*, which provides useful intuitions. Later, we replace this assumption by a more realistic one, only assuming that the hash is *computationally collision-resistant*, which will require to weaken the security property by allowing for some error probability.

d) Security proof (perfect hash): We present in the left part of Fig. 5 the SQUIRREL lemma and proof, assuming an idealized hash satisfying

$$\dot\forall\ (x : \text{message})\ (y : \text{message}),\ \text{fhash}\ x = \text{fhash}\ y \dot\Rightarrow x = y.$$

The proof is by induction over the length n of the membership proof p, generalized over p and t (6). Induction over the nat inductive type includes a case analysis leading to two subcases, for n=Z (7) and n=S n_0 (8). When n=Z (7), we have p = Emp, and the hypothesis on verify tells us that hash_tree t = hash_tree t_{sub}. We conclude by using an auxiliary lemma (whose proof we omit):

```
lemma hash_tree_injective (t : tree) (t' : tree) :
  hash_tree t = hash_tree t' ⇛ t = t'.
```

and the fact that lookup(Emp,t) = t. When n=S n_0 (8), we prove by case-analysis that p = Cons i h_{1-i} p_i (9). Then, we introduce the length and verification assumptions, respectively named L and H, in the context (10). After some simplifications, omitted here, and if we let h_i = hash_path (hash_tree t_{sub}) p_i, we get

```
H: hash_tree t = fhash (encode_node h_0 h_1)
```

By using the injectivity of fhash in hypothesis H (11), we can strip-off the top-most hash and get

```
hash_tree0 t = encode_node h_0 h_1
```

By case-analysis over t (13), and since our encodings are non-ambiguous, we can easily show that t cannot be a leaf (14). At this point of the proof, the only remaining case to handle is t = Node t_0 t_1 (15). After some simplifications and using the injectivity of the node encoding in H, we get

```
H: (hash_tree t_i, hash_tree t_{1-i}) = (h_0,h_1)
```

Coming back to the definition of h_i, this notably entails that

$$\text{hash\_tree}\ t_i = \text{hash\_path}\ (\text{hash\_tree}\ t_{sub})\ p_i \qquad (\dagger)$$

Further, we know that lookup descends in the branch t_i of t, and we must prove that lookup (pi, ti) = t_{sub}. By applying the induction hypothesis (16), it only remains to show that p_i proves that t_{sub} is a subtree of t_i. We easily conclude (17) using H, as this is exactly the property in Eq. (†) above.

e) Collision-resistance: We now present a proof of security under the more realistic assumption that fhash is collision-

```
lemma verify_correct (n : nat) (p : proof) (t : tree) (t_sub : tree) :
  length p = n ⇒
  verify (hash_tree t_sub) p (hash_tree t) ⇒
  lookup (p,t) = t_sub.
Proof.
  generalize p t; induction n.                    (* ind. + case analysis on n *)
  + ...; by apply hash_tree_injective.                    (* case n = Z *)
  + intro n_0 IH p t.                                     (* case n = S n_0 *)
    case p; ...          (* since n = S n_0, we have p = Cons i h_{1-i} p_i *)
    intro L H; ...
    apply fhash_injective in H.                  (* use injectivity of fhash *)
    ...
    case t.                                          (* case analysis on t *)
    - ...      (* case t = Leaf _, impossible (non–ambiguous encoding) *)
    - ...                                                 (* case t = Node t_0 t_1 *)
      apply IH.                                  (* apply induction hypothesis *)
      ...                                        (* we conclude easily using H *)
Qed.
```

The left proof assumes a perfect injective hash, while the right proof relies on a collision-resistance assumption.

```
global lemma verify_correct
  (n : nat[const]) (p : proof[adv]) (t : tree[adv]) (t_sub : tree[const, adv]) :
  [ length p = n ⇒
    verify (hash_tree t_sub) p (hash_tree t) ⇒
    lookup (p,t) = t_sub
    <: tree_error t_sub + path_error n ].
Proof.
  generalize p t. induction ∼concrete ∼general n.
  intro n IH p t.
  have [C | [n0 C] ] := case_const_nat n; ...
  + ...; by apply hash_tree_injective.          (* pay error |t_sub| * e_col *)
  + case ∼tags p <: z, error; ...          (* move error to p = Cons ... *)
    intro L H; ...
    apply localize(fhash_cr _ _) in H.          (* pay collision error of e_col *)
    ...
    case ∼tags t <: z, error – e_col; ...     (* move error to t = Node(_,_) *)
    - ...                                  (* exact reasoning, no error loss *)
    - ...
      apply IH n_0; ...                                  (* pay error |t_i| * e_col *)
      ...                                  (* additional basic bound reasoning *)
Qed.
```

We use the shorthand error = $\text{tree}_{\text{error}}\ t_{\text{sub}} + \text{path}_{\text{error}}\ n$.

Figure 5: Proofs of security for the Merkle tree data-structure in SQUIRREL.

resistant [31] (CR), which states that it is computationally infeasible to produce collisions for fhash. To model it, we assume from now on that fhash is a keyed hash function, where the key (of type hkey) is randomly sampled:

```
name k : hkey.        op fhash : message → hkey → message.
```

We assume that it is computationally impossible to produce two values x and y such that $x \neq y$ and (fhash x = fhash y), except with some small probability of error. We let e_{col} be an abstract real value representing this probability. CR is a computational property that only applies to values x and y that can be computed in polynomial-time (in SQUIRREL, a quantification over a variable x can be restricted to be polynomial-time using the adv tag). Furthermore, CR allows x and y to depend on the key k, which is thus given as argument to x and y. Thus, we model CR using the axiom

```
axiom fhash_cr (x, y : hkey → message[adv]) :
  fhash (x k) k = fhash (y k) k ⇒ x k = y k <: e_col.
```

f) Approximated security property: We must now account for a probability of error. We will apply the collision-resistance assumption once per node in the membership proof p, plus once for each node in the subtree t_{sub}. Thus, we will have an error of $\text{tree}_{\text{error}}\ t_{\text{sub}} + \text{path}_{\text{error}}\ n$, where $\text{tree}_{\text{error}}\ t = (\text{size}\ t) * e_{\text{col}}$ and $\text{path}_{\text{error}}\ n = n * e_{\text{col}}$ (we omit the conversions between int, nat, and real). In the lemma given on the right part of Fig. 5, we require that the membership proof for Merkle trees can be produced in polynomial-time through the adv tag. We require that n is a constant value through the const tag, as we use the global case-disjunction tactic in the proof. We make the same assumption on t_{sub}, which we need for a similar reason in the proof of the approximated version of hash_tree_injective (whose proof we omit):

```
global lemma hash_tree_injective (t : tree[adv]) (t' : tree[adv,const]):
  [ hash_tree t = hash_tree t' ⇒ t = t' <: tree_error t' ].
```

g) Security proof (collision-resistance): The approximated proof on the right of Fig. 5 closely follows exact proof, though there are several differences. First, we use a different induction rule to avoid unnecessarily paying for errors, as the standard rule on inductive data-type would eagerly localize the induction hypothesis. Second, because of this, global case analysis is not built in the induction rule. Instead, we manually perform a global case analysis (10), which requires no advantage reasoning. Local case analyses must describe how the error mass is moved around (12, 16). Here, we always move all the mass to a single case, as the other case is shown to be impossible with probability 1. Note that we use a special, more precise, version of the case tactic, which keeps track of the tags such as adv. Finally, we carefully track the error. We pay ($\text{tree}_{\text{error}}\ t_{\text{sub}}$) when we apply the hash_tree_injective lemma (11), which is done only once when n = Z. Then, we pay e_{col} to use collision resistance (14) in main step of the induction. This step is repeated n times, leading to an accumulated error of $n * e_{\text{col}}$, which is exactly $\text{path}_{\text{error}}\ n$.

h) Discussion: Because CR is a computational assumption, we must establish that we apply it on polynomial-time computable terms, which SQUIRREL tries to check using the adv tag. We use user-defined recursive functions to define our operations on Merkle trees. Unfortunately, SQUIRREL relies on a basic fully automated complexity checker which does not support recursion. Thus, we tell SQUIRREL to admit that these functions are polynomial-time. It would be interesting to see whether this limitation of the tool could be solved, e.g., using techniques from implicit complexity [32].

VI. RELATED WORK

a) CAC: We quickly compare our work with the main CAC tools. CryptoVerif [33] is a highly automated tool based on bespoke games transformations. While it can automatically infer advantage bounds without any annotations from the user, it does not support generic mathematical reasoning nor inductive reasoning, which puts the analysis of stateful protocols and inductive data-structures out-of-reach of this tool.

SSPROVE [34] and EASYCRYPT [35] are two tactic-based tools for CAC, the former as a ROCQ framework, and the

latter as a standalone tool. While both support reasoning on bounds for imperative programs, e.g., via a union-bound Hoare logic [27], they do not support approximated reasoning for higher-order formulas as we do. Further, they have no dedicated supports for bounds in their proof term elaborators.

b) Elaboration: Elaborators are crucial to improve the usability of proof assistants, and have mostly been studied in the context of the calculus of inductive constructions, e.g., see [21], [36]. However, they focus on coercion, which we do not consider here. Our elaborator architecture takes inspiration from *bidirectional typing* [29]. Roughly, inference and checking in bi-directional typing correspond to, respectively, elaboration and re-elaboration. Finally, directed elaboration is inspired from [37]. We are not aware of any usability result comparable to our erasability theorem for other elaborators.

c) Multi-level contexts: Two-level contexts in SQUIRREL have been introduced in [15], though that work only considers the asymptotic logic, and does not allow for error bounds. The idea of having multi-level contexts has been explored in other logics. Notably, *Iris Proof Mode* [38], [39] (IPM) brings multi-level supports for separation logic in Iris [40]. Also, in [38], the IPM-managed separation logic context is separated from the ROCQ managed standard context. We are not aware of any work building a tactic language for a graded multi-level context as we do (for us, the grading is the error bound).

d) Others: In [41], the authors introduce Eris, a program logic that enables reasoning about error probability bounds for probabilistic program by handling errors as a separation logic resource. Here, we have a logic closer to standard higher logic, using a graded modality for errors. Our focus is on usability, by designing (such as a proof-term elaborator) built on top of that logic, which is not present in [41].

VII. CONCLUSION

We designed an extension of SQUIRREL supporting concrete mechanized proofs of reachability properties while only requiring a limited amount of user bound annotations. We designed a proof term language and elaborator which can infer bound annotations and weakenings, and proved its usefulness through a theoretical erasure result. We applied our extension to case studies, providing evidence that reachability proofs can tractably be adapted to a concrete setting.

As future work, we would like our extension to support concrete indistinguishability proofs. While the theoretical foundations for this have already been laid out [1], it is not clear how to do so without drowning users with the tedious complexity reasonings that are pervasive in such proofs. Another research direction would be to study the interaction between locality inference and coercion.

REFERENCES

[1] D. Baelde, C. Fontaine, A. Koutsos, G. Scerri, and T. Vignon, "A probabilistic logic for concrete security," in *CSF*. IEEE, 2024, pp. 324–339.

[2] RFC, "The transport layer security (TLS) protocol version 1.3," https://datatracker.ietf.org/doc/html/rfc8446, 2018.

[3] R. Tamassia, "Authenticated data structures," in *ESA*, ser. Lecture Notes in Computer Science, vol. 2832. Springer, 2003, pp. 2–5.

[4] R. C. Merkle, "A digital signature based on a conventional encryption function," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 293. Springer, 1987, pp. 369–378.

[5] S. Goldwasser and S. Micali, "Probabilistic encryption and how to play mental poker keeping secret all partial information," in *STOC*. ACM, 1982, pp. 365–377.

[6] M. Bellare, "Practice-oriented provable security," in *Lectures on Data Security*, ser. Lecture Notes in Computer Science, vol. 1561. Springer, 1998, pp. 1–15.

[7] M. Bellare and P. Rogaway, "The security of triple encryption and a framework for code-based game-playing proofs," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 4004. Springer, 2006, pp. 409–426.

[8] M. Fischlin and A. Mittelbach, "An overview of the hybrid argument," *IACR Cryptol. ePrint Arch.*, p. 88, 2021.

[9] S. M. Bellovin and M. Merritt, "Encrypted key exchange: password-based protocols secure against dictionary attacks," in *S&P*. IEEE Computer Society, 1992, pp. 72–84.

[10] M. Barbosa, G. Barthe, K. Bhargavan, B. Blanchet, C. Cremers, K. Liao, and B. Parno, "Sok: Computer-aided cryptography," in *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 2021, pp. 777–795.

[11] T. S. P. development team, "Squirrel Prover," 2026. [Online]. Available: https://github.com/squirrel-prover/squirrel-prover

[12] D. Baelde, S. Delaune, C. Jacomme, A. Koutsos, and S. Moreau, "An interactive prover for protocol verification in the computational model," in *SP*. IEEE, 2021, pp. 537–554.

[13] G. Bana and H. Comon-Lundh, "A computationally complete symbolic attacker for equivalence properties," in *CCS*. ACM, 2014, pp. 609–620.

[14] D. Baelde, A. Koutsos, and J. Lallemand, "A higher-order indistinguishability logic for cryptographic reasoning," in *LICS*, 2023, pp. 1–13.

[15] D. Baelde, S. Delaune, A. Koutsos, and S. Moreau, "Cracking the stateful nut: Computational proofs of stateful security protocols using the Squirrel proof assistant," in *CSF*. IEEE, 2022, pp. 289–304.

[16] D. Baelde, A. Koutsos, and J. Sauvage, "Leveraging Cryptographic Simulator Synthesis for Formally Verifying the FOO E-Voting Protocol," in *Usenix 2026 - 35th Usenix security symposium*, Baltimore, United States, Aug. 2026. [Online]. Available: https://inria.hal.science/hal-05453231

[17] C. Cremers, C. Fontaine, and C. Jacomme, "A logic and an interactive prover for the computational post-quantum security of protocols," in *SP*. IEEE, 2022, pp. 125–141.

[18] The Rocq development team, "The rocq proof assistant," 2026. [Online]. Available: https://rocq-prover.org/

[19] The Lean development team, "The lean proof assistant," 2026. [Online]. Available: https://lean-lang.org/

[20] T. R. development team, "The rocq documentation for introduction patterns," 2026. [Online]. Available: https://rocq-prover.org/doc/V9.1.0/refman/proof-engine/tactics.html#intro-patterns

[21] A. Asperti, W. Ricciotti, C. S. Coen, and E. Tassi, "A bi-directional refinement algorithm for the calculus of (co)inductive constructions," *Logical Methods in Computer Science*, vol. Volume 8, Issue 1, Mar 2012. [Online]. Available: https://lmcs.episciences.org/1044

[22] C. Paulin-Mohring, "Inductive definitions in the system coq - rules and properties," in *TLCA*, ser. Lecture Notes in Computer Science, vol. 664. Springer, 1993, pp. 328–345.

[23] T. Y. C. Woo and S. S. Lam, "A semantic model for authentication protocols," in *Proceedings 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1993, pp. 178–194.

[24] G. Lowe, "A hierarchy of authentication specifications," in *Proceedings 10th Computer Security Foundations Workshop*, 1997, pp. 31–43.

[25] M. Barbosa, G. Barthe, B. Grégoire, A. Koutsos, and P. Strub, "Mechanized proofs of adversarial complexity and application to universal composability," *ACM Trans. Priv. Secur.*, vol. 26, no. 3, pp. 41:1–41:34, 2023.

[26] C. Fontaine, A. Koutsos, G. Scerri, and T. Vignon, "Interactive proofs in higher-order logic with errors and application to concrete cryptography," May 2026. [Online]. Available: https://doi.org/10.5281/zenodo.20285718

[27] G. Barthe, M. Gaboardi, B. Grégoire, J. Hsu, and P. Strub, "A program logic for union bounds," in *ICALP*, ser. LIPIcs, vol. 55. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, pp. 107:1–107:15.

[28] D. Baelde, S. Delaune, and S. Riou, "Smt-based automation for overwhelming truth," in *38th IEEE Computer Security Foundations*

Symposium, CSF 2025, Santa Cruz, CA, USA, June 16-20, 2025. IEEE, 2025, pp. 505–520. [Online]. Available: https://doi.org/10.1109/CSF64896.2025.00019

[29] J. Dunfield and N. Krishnaswami, "Bidirectional typing," *ACM Comput. Surv.*, vol. 54, no. 5, May 2021. [Online]. Available: https://doi.org/10.1145/3450952

[30] R. Künnemann, "Foundations for analyzing security APIs in the symbolic and computational model. (fondations d'analyse de APIs de sécurité dans le modèle symbolique et calculatoire)," Ph.D. dissertation, École normale supérieure de Cachan, France, 2014. [Online]. Available: https://tel.archives-ouvertes.fr/tel-00942459

[31] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*, 2nd ed. Chapman & Hall/CRC, 2014.

[32] S. J. Bellantoni and S. A. Cook, "A new recursion-theoretic characterization of the polytime functions (extended abstract)," in *STOC*. ACM, 1992, pp. 283–293.

[33] B. Blanchet, "A computationally sound mechanized prover for security protocols," *IEEE Trans. Dependable Secur. Comput.*, vol. 5, no. 4, pp. 193–207, 2008. [Online]. Available: https://doi.org/10.1109/TDSC.2007.1005

[34] P. G. Haselwarter, E. Rivas, A. Van Muylder, T. Winterhalter, C. Abate, N. Sidorenco, C. Hritcu, K. Maillard, and B. Spitters, "Ssprove: A foundational framework for modular cryptographic proofs in coq," *ACM Trans. Program. Lang. Syst.*, vol. 45, no. 3, pp. 15:1–15:61, 2023.

[35] G. Barthe, B. Grégoire, S. Heraud, and S. Zanella-Béguelin, "Computer-aided security proofs for the working cryptographer," in *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, ser. Lecture Notes in Computer Science, P. Rogaway, Ed., vol. 6841. Springer, 2011, pp. 71–90. [Online]. Available: https://doi.org/10.1007/978-3-642-22792-9_5

[36] A. Saïbi, "Typing algorithm in type theory with inheritance," in *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL '97. New York, NY, USA: Association for Computing Machinery, 1997, p. 292–301. [Online]. Available: https://doi.org/10.1145/263699.263742

[37] M. Lennon-Bertrand, "Complete Bidirectional Typing for the Calculus of Inductive Constructions," in *12th International Conference on Interactive Theorem Proving (ITP 2021)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), L. Cohen and C. Kaliszyk, Eds., vol. 193. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, pp. 24:1–24:19. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITP.2021.24

[38] R. Krebbers, A. Timany, and L. Birkedal, "Interactive proofs in higher-order concurrent separation logic," in *POPL*. ACM, 2017, pp. 205–217.

[39] R. Krebbers, J. Jourdan, R. Jung, J. Tassarotti, J. Kaiser, A. Timany, A. Charguéraud, and D. Dreyer, "Mosel: a general, extensible modal framework for interactive proofs in separation logic," *Proc. ACM Program. Lang.*, vol. 2, no. ICFP, pp. 77:1–77:30, 2018.

[40] R. Jung, R. Krebbers, J.-H. Jourdan, A. Bizjak, L. Birkedal, and D. Dreyer, "Iris from the ground up: A modular foundation for higher-order concurrent separation logic," *Journal of Functional Programming*, 2018.

[41] A. Aguirre, P. G. Haselwarter, M. de Medeiros, K. H. Li, S. O. Gregersen, J. Tassarotti, and L. Birkedal, "Error credits: Resourceful reasoning about error bounds for higher-order probabilistic programs," *Proc. ACM Program. Lang.*, vol. 8, no. ICFP, Aug. 2024. [Online]. Available: https://doi.org/10.1145/3674635

Appendix A
Proof Terms Syntax and Typing Rules

a) Logical proof system: We will justify the soundness of our proof term typing rules using the logical rules of [1], with one exception. The authors of [1] provided the following right rule for local quantification:

$$\frac{\mathrm{L}_{\varepsilon}.\text{R-}\forall \qquad \mathbb{E}, (x : \tau); \Gamma; \Lambda \vdash_{\varepsilon} \dot{\phi}}{\mathbb{E}; \Gamma; \Lambda \vdash_{\varepsilon} \dot{\forall}(x : \tau).\, \dot{\phi}}$$

$$\begin{array}{lll}
\overline{\mathsf{pt}} ::= \mathsf{id} & & \\
\quad \mid \ \% \, \overline{\mathsf{pt}} & \mid \ [\overline{\mathsf{pt}}] & \\
\quad \mid \ \overline{\mathsf{pt}} \, \dot{\circ} \, \overline{\mathsf{pt}} & \mid \ \overline{\mathsf{pt}} \, \tilde{\circ} \, \overline{\mathsf{pt}} & \\
\quad \mid \ \overline{\mathsf{pt}} \, \dot{\circ} \, t & \mid \ \overline{\mathsf{pt}} \, \tilde{\circ} \, t & \\
\quad \mid \ \dot{\lambda}(x : \tau).\, \overline{\mathsf{pt}} & \mid \ \tilde{\lambda}(x : \tau).\, \overline{\mathsf{pt}} & \\
\quad \mid \ \dot{\Pi}(\mathsf{id} : \dot{\phi}).\, \overline{\mathsf{pt}} & \mid \ \tilde{\Pi}(\mathsf{id} : \tilde{\phi}).\, \overline{\mathsf{pt}} & \\
\quad \mid \ (\overline{\mathsf{pt}} \lessdot \varepsilon(\overline{\mathsf{pt}})) & &
\end{array}$$

Figure 6: Syntax of annotated proof-terms.

Note that since we assume that both judgements are well-formed, we must have that x is not free in ε. In this work, we use a more general rule that does not suffer from this limitation:

$$\frac{\mathrm{L}^{\mathsf{m}}_{\varepsilon}.\text{R-}\forall \qquad \mathbb{E}, (x : \tau); \Gamma; \Lambda \vdash_{\varepsilon} \dot{\phi} \qquad \tau \text{ is finite}}{\mathbb{E}; \Gamma; \Lambda \vdash_{\varepsilon_{\mathsf{m}}} \dot{\forall}(x : \tau).\, \dot{\phi}}$$

where $\varepsilon_{\mathsf{m}} = \max_x(\varepsilon)$. Here, x may appear freely in ε. As discussed in Remark 1, we must also restrict this rule to finite types to ensure that bounds are always well-defined values of type real. We start by proving the correction of this rule before moving to our proof term syntax and typing rules.

Proposition 1. *The rule* $\mathrm{L}^{\mathsf{m}}_{\varepsilon}.\text{R-}\forall$ *is sound.*

Proof. From the premise, we know that for any model $\mathbb{M} : \mathbb{E}$, we have that

$$\mathbb{M} \models \tilde{\forall}(x : \tau).\, (\tilde{\wedge}\Gamma) \mathrel{\tilde{\Rightarrow}} [(\dot{\wedge}\Lambda) \mathrel{\dot{\Rightarrow}} \dot{\phi}]_{\varepsilon}$$

By weakening and using the fact that $[\varepsilon \leq \max_x(\varepsilon)]_0$, we know that

$$\mathbb{M} \models \tilde{\forall}(x : \tau).\, (\tilde{\wedge}\Gamma) \mathrel{\tilde{\Rightarrow}} [(\dot{\wedge}\Lambda) \mathrel{\dot{\Rightarrow}} \dot{\phi}]_{\max_x(\varepsilon)}$$

Observe that x does not appear freely in Γ (otherwise, the conclusion of the rule would not be well-formed). Thus, we can push the global quantification downwards:

$$\mathbb{M} \models (\tilde{\wedge}\Gamma) \mathrel{\tilde{\Rightarrow}} \tilde{\forall}(x : \tau).\, [(\dot{\wedge}\Lambda) \mathrel{\dot{\Rightarrow}} \dot{\phi}]_{\max_x(\varepsilon)}$$

We know that x does not appear freely in $\max_x(\varepsilon)$. Thus, using [1, Proposition 2], we have that

$$\mathbb{M} \models (\tilde{\wedge}\Gamma) \mathrel{\tilde{\Rightarrow}} [\dot{\forall}(x : \tau).\, (\dot{\wedge}\Lambda) \mathrel{\dot{\Rightarrow}} \dot{\phi}]_{\max_x(\varepsilon)}$$

Again, we know that x does not appear freely in Λ since the conclusion of the rule is well-formed. Thus, we have

$$\mathbb{M} \models (\tilde{\wedge}\Gamma) \mathrel{\tilde{\Rightarrow}} [(\dot{\wedge}\Lambda) \mathrel{\dot{\Rightarrow}} \dot{\forall}(x : \tau).\, \dot{\phi}]_{\max_x(\varepsilon)} \tag{2}$$

Which concludes this proof. □

Axioms rules

$$\textsc{Ty.L.Ax}\ \frac{(\mathsf{pt}, \dot\phi) \in \Lambda}{\mathbb{E};\Gamma;\Lambda \vdash \mathsf{pt} : (\dot\phi \prec 0)} \qquad \textsc{Ty.G.Ax}\ \frac{(\mathsf{pt}, \tilde\phi) \in \Gamma}{\mathbb{E};\Gamma \vdash \mathsf{pt} : \tilde\phi}$$

Implications rules

$$\textsc{Ty.I.}\dot{\sqsubset}\ \frac{\mathbb{E};\Gamma;\Lambda \vdash \mathsf{pt} : (\dot\phi_0 \Rightarrow \dot\phi \prec \varepsilon_1) \qquad \mathbb{E};\Gamma;\Lambda \vdash \mathsf{pt}' : (\dot\phi_0 \prec \varepsilon_2)}{\mathbb{E};\Gamma;\Lambda \vdash \mathsf{pt} \mathbin{\dot\sqsubset} \mathsf{pt}' : (\dot\phi \prec \varepsilon_1 + \varepsilon_2)} \qquad \textsc{Ty.I.}\tilde{\sqsubset}\ \frac{\mathbb{E};\Gamma \vdash \mathsf{pt} : \tilde\phi_0 \mathrel{\tilde\Rightarrow} \tilde\phi \qquad \mathbb{E};\Gamma \vdash \mathsf{pt}' : \tilde\phi_0}{\mathbb{E};\Gamma \vdash \mathsf{pt} \mathbin{\tilde\sqsubset} \mathsf{pt}' : \tilde\phi}$$

$$\textsc{Ty.}\dot\Pi\ \frac{\mathbb{E};\Gamma;\Lambda, (\mathsf{id} : \dot\phi) \vdash \mathsf{pt} : (\dot\phi_0 \prec \varepsilon)}{\mathbb{E};\Gamma;\Lambda \vdash \dot\Pi(\mathsf{id} : \dot\phi).\ \mathsf{pt} : (\dot\phi \mathrel{\dot\Rightarrow} \dot\phi_0 \prec \varepsilon)} \qquad \textsc{Ty.}\tilde\Pi\ \frac{\mathbb{E};\Gamma, (\mathsf{id} : \tilde\phi) \vdash \mathsf{pt} : \tilde\phi_0}{\mathbb{E};\Gamma \vdash \tilde\Pi(\mathsf{id} : \tilde\phi).\ \mathsf{pt} : \tilde\phi \mathrel{\tilde\Rightarrow} \tilde\phi_0}$$

Universal quantification rules

$$\textsc{Ty.T.}\dot{\sqsubset}\ \frac{\mathbb{E};\Gamma;\Lambda \vdash \mathsf{pt} : (\dot\forall(x : \tau).\dot\phi \prec \varepsilon) \qquad \mathbb{E} \vdash t : \tau}{\mathbb{E};\Gamma;\Lambda \vdash \mathsf{pt} \mathbin{\dot\sqsubset} t : (\dot\phi[x \mapsto t] \prec \varepsilon)} \qquad \textsc{Ty.T.}\tilde{\sqsubset}\ \frac{\mathbb{E};\Gamma \vdash \mathsf{pt} : \tilde\forall(x : \tau).\tilde\phi \qquad \mathbb{E} \vdash t : \tau}{\mathbb{E};\Gamma \vdash \mathsf{pt} \mathbin{\tilde\sqsubset} t : \tilde\phi[x \mapsto t]}$$

$$\textsc{Ty.}\dot\lambda\ \frac{\mathbb{E}, (x : \tau);\Gamma;\Lambda \vdash \mathsf{pt} : (\dot\phi \prec \varepsilon) \quad \tau \text{ is finite}}{\mathbb{E};\Gamma;\Lambda \vdash \dot\lambda(x : \tau).\ \mathsf{pt} : (\dot\forall(x : \tau).\dot\phi \prec \max_x(\varepsilon))} \qquad \textsc{Ty.}\tilde\lambda\ \frac{\mathbb{E}, (x : \tau);\Gamma \vdash \mathsf{pt} : \tilde\phi}{\mathbb{E};\Gamma \vdash \tilde\lambda(x : \tau).\ \mathsf{pt} : \tilde\forall(x : \tau).\tilde\phi}$$

Localisation and weakening rules

$$\textsc{Ty.ByLoc}\ \frac{\mathbb{E};\Gamma \vdash \mathsf{pt} : [\dot\phi]_\varepsilon}{\mathbb{E};\Gamma;\Lambda \vdash \%\,\mathsf{pt} : (\dot\phi \prec \varepsilon)} \qquad \textsc{Ty.ByGlob}\ \frac{\mathbb{E};\Gamma;\emptyset \vdash \mathsf{pt} : (\dot\phi \prec \varepsilon)}{\mathbb{E};\Gamma \vdash [\mathsf{pt}] : [\dot\phi]_\varepsilon} \qquad \textsc{Ty.Weak}\ \frac{\mathbb{E};\Gamma;\Lambda \vdash \mathsf{pt} : (\dot\phi \prec \varepsilon') \qquad \mathbb{E};\Gamma \vdash \mathsf{pt}' : [\varepsilon' \le \varepsilon]_0}{\mathbb{E};\Gamma;\Lambda \vdash (\mathsf{pt} \prec \varepsilon(\mathsf{pt}')) : (\dot\phi \prec \varepsilon)}$$

Figure 7: Proof term typing rules.

b) Proof terms and typing rules: We recall the syntax of annotated proof terms in Fig. 6. A proof term context $\mathbb{E};\Gamma;\Lambda$ is well formed when all free variables of Γ and Λ are bound by $\mathbb{E}$. A proof term pt is well formed w.r.t. a well-formed context $\mathbb{E};\Gamma;\Lambda$ when all free variables of pt are bound by $\mathbb{E}$, and all hypothesis identifiers in pt are bound by $\Gamma;\Lambda$. From now on, we only consider well-formed proof term contexts and proof terms.

We present the proof term typing rules in Fig. 7. These rules are immediate translation of the logical rules of [1] (except that we replace L_ε.R-$\forall$ by $\mathrm{L}^{\mathrm{m}}_\varepsilon$.R-$\forall$). E.g., the rule ByLoc, which we recall in Fig. 1, corresponds to Ty.ByLoc. Thus, the proposition immediately follows from the soundness of the proof-system of [1] and Proposition 1.

Proposition 2. *We have the following soundness results:*

- *If* $\mathbb{E};\Gamma;\Lambda \vdash \overline{\mathsf{pt}} : (\dot\phi \prec \varepsilon)$ *is derivable then* $\mathbb{E};\Gamma;\Lambda \vdash_\varepsilon \dot\phi$ *is valid.*
- *If* $\mathbb{E};\Gamma \vdash \overline{\mathsf{pt}} : \tilde\phi$ *is derivable then* $\mathbb{E};\Gamma \vdash \tilde\phi$ *is valid.*

We use $\equiv_\alpha$ to denote equality modulo α between global formulas and terms.

Proposition 3 (Uniqueness of the derivated type)**.**

- *if* $\mathbb{E};\Gamma;\Lambda \vdash \overline{\mathsf{pt}} : (\dot\phi \prec \varepsilon)$ *and* $\mathbb{E};\Gamma;\Lambda \vdash \overline{\mathsf{pt}} : (\dot\phi_0 \prec \varepsilon')$ *are both derivable, then* $\dot\phi \equiv_\alpha \dot\phi_0$ *and* $\varepsilon \equiv_\alpha \varepsilon'$.
- *if* $\mathbb{E};\Gamma \vdash \overline{\mathsf{pt}} : \tilde\phi$ *and* $\mathbb{E};\Gamma \vdash \overline{\mathsf{pt}} : \tilde\phi_0$ *then* $\tilde\phi \equiv_\alpha \tilde\phi_0$.

Proof. This directly follows from the fact that there is only one rule for each possible top-level constructor of an annotated proof term. □

Hypotheses footprint: A footprint I is a list of identifiers $\mathsf{id}_1, \ldots, \mathsf{id}_n$, and is used to record the hypotheses used by a proof term. If I is a footprint and Λ a local proof context, we let $\mathsf{filter}(\Lambda\ ;\ I)$ be the restriction of Λ to hypotheses $\mathsf{id} : \dot\phi$ such that $\mathsf{id} \in I$. We similarly define $\mathsf{filter}(\Gamma\ ;\ I)$ for any global proof context Γ.

Then, we introduce proof term typing judgements with memory footprints:

$$\mathbb{E};\Gamma;\Lambda \vdash_I \overline{\mathsf{pt}} : (\dot\phi \prec \varepsilon) \qquad \mathbb{E};\Gamma;\Lambda \vdash_I \overline{\mathsf{pt}} : \tilde\phi$$

and we say that:

- $\mathbb{E};\Gamma;\Lambda \vdash_I \overline{\mathsf{pt}} : (\dot\phi \prec \varepsilon)$ is valid if $\mathbb{E};\Gamma;\mathsf{filter}(\Lambda\ ;\ I) \vdash \overline{\mathsf{pt}} : (\dot\phi \prec \varepsilon)$ is derivable.
- $\mathbb{E};\Gamma;\Lambda \vdash_I \overline{\mathsf{pt}} : \tilde\phi$ is valid if $\mathbb{E};\Gamma \vdash \overline{\mathsf{pt}} : \tilde\phi$ is derivable and we have $\mathsf{filter}(\Lambda\ ;\ I) = \emptyset$.

Appendix B
Proof Term Elaborator

Recall that a proof context Δ is a triple $(\mathbb{E};\Gamma;\Lambda)$ comprising a typing environment $\mathbb{E}$, a global context Γ, and a local context Λ. Let $\Delta = (\mathbb{E};\Gamma;\Lambda)$ be a proof context and $(x : \tau)$

be a variable declaration. Then $\Delta, (x : \tau)$ is the proof-context Δ in which the environment $\mathbb{E}$ is extended with $(x : \tau)$, i.e.,

$$\Delta, (x : \tau) \stackrel{\text{def}}{=} \Big((\mathbb{E}, (x : \tau)); \Gamma; \Lambda \Big)$$

We define similarly $\Delta, (\mathsf{id} : \tilde{\phi})$ and $\Delta, (\mathsf{id} : \dot{\phi})$, by extending, respectively, Γ and Λ.

A. Rules

We present the basic, logic,-specific and constructor elaboration rules in Fig. 8, and the applications rules in Fig. 9. The directed elaboration and re-elaboration rules are in Fig. 10.

a) Patterns and matching: A *pattern* P w.r.t. and environment $\mathbb{E}$ is a formula ϕ well-typed in $\mathbb{E}$ with a number of distinguished variables $\{\_1, \ldots, \_n\}$ called *holes*, such that no hole occur more than once in P. We usually do not number hole variables and, e.g., write $\_ \tilde{\wedge} \_$ instead of $\_1 \tilde{\wedge} \_2$. We require that patterns only use holes to denote missing global or local sub-formulas, or missing bounds: we do not allow holes in all generality in *terms*. For example, $\_ \tilde{\wedge} \_$, $\_ \dot{\wedge} \_$, and $[\_]\_$ are allowed, but $[\_]\_ + \_$ is not.

We say that ϕ matches P, which we write $\mathbb{E}_0 \vdash \phi \stackrel{?}{\leq} P$, if there exists a substitution σ whose domain is exactly the holes of P such that $P\sigma \equiv_\alpha \phi$.

B. Soundness of Elaboration

Definition 1 (Elaboration)**.** The elaboration

$$\Delta_0 \vdash \mathsf{pt} \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_I \phi$$

is valid if it returns a well-typed proof term typing judgement

$$\Delta_1 \vdash_I \overline{\mathsf{pt}} : \phi.$$

In that case, we have that $\overline{\mathsf{pt}}$ is an elaboration of pt.

Definition 2 (Directed elaboration)**.** The directed elaboration

$$\Delta_0 \vdash (\mathsf{pt} \ \triangleleft \ P) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_I \phi$$

is valid if

1) $\Delta_1 \vdash_I \overline{\mathsf{pt}} : \phi$ is well-typed;
2) ϕ matches P.

Definition 3 (Re-elaboration)**.** The re-elaboration

$$\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_{I_0} \phi_0 \triangleleft P \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1$$

is valid if

1) The well-typedness of $\Delta_0 \vdash_{I_0} \overline{\mathsf{pt}}_0 : \phi_0$ implies that of $\Delta_1 \vdash_{I_1} \overline{\mathsf{pt}}_1 : \phi_1$;
2) ϕ_1 matches P.

a) Context generalization: We present some technical results that will be useful for proving the soundness of our system.

First, if $\Delta_0 = \mathbb{E}_0; \Gamma_0; \Lambda_0$ and $\Delta_1 = \mathbb{E}_1; \Gamma_1; \Lambda_1$ are two proof contexts, we let

$$\Delta_0, \Delta_1 = \mathbb{E}_0, \mathbb{E}_1; \Gamma_0, \Gamma_1; \Lambda_0, \Lambda_1$$

be the concatenation of both contexts. We define this operation even if Δ_0 or Δ_1 are not well-formed context.

Definition 4 (Order on context)**.**
A context Δ_0 is larger than Δ_1, which we write $\Delta_0 \geq \Delta_1$, if there exists Δ_0' such that $\Delta_1 \equiv_\alpha \Delta_0, \Delta_0'$.

Intuitively, Δ_0 is larger than Δ_1 if the latter contains more hypotheses than the former — there more hypotheses they are, the more restrictive a context is, i.e., it makes more assumptions.

Lemma 1 (Elaboration judgements specializations)**.**

- $\Delta_0 \geq \Delta_1$ *if any of the following judgements is derivable:*

$$\Delta_0 \vdash \mathsf{pt} \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_I \phi$$
$$\Delta_0 \vdash (\mathsf{pt} \ \triangleleft \ P) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_I \phi$$
$$\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_I \phi_0 \triangleleft P \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_\emptyset \phi_1$$

- *If* $\Delta_0 \geq \Delta_1$ *and* pt *is well-formed in* Δ_0*, then it is also well-formed in* Δ_1.

Lemma 2 (Effect of elaborations on the proof context)**.**
If $\Delta_0 \geq \Delta_1$ *and* pt *is well-formed in* Δ_0*, then*

- Re-elaboration:

$$\Delta_0 \vdash \mathsf{pt} :_I \phi_0 \triangleleft P \twoheadrightarrow \Delta_2 \vdash \overline{\mathsf{pt}} :_I \phi$$
$$\textit{iff} \ \ \Delta_1 \vdash \mathsf{pt} :_I \phi_0 \triangleleft P \twoheadrightarrow \Delta_3 \vdash \overline{\mathsf{pt}} :_I \phi.$$

- Elaboration:

$$\Delta_0 \vdash \mathsf{pt} \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}} :_I \phi$$
$$\textit{iff} \ \ \Delta_1 \vdash \mathsf{pt} \rightsquigarrow \Delta_3 \vdash \overline{\mathsf{pt}} :_I \phi.$$

- Directed elaboration:

$$\Delta_0 \vdash (\mathsf{pt} \ \triangleleft \ P) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}} :_I \phi$$
$$\textit{iff} \ \ \Delta_1 \vdash (\mathsf{pt} \ \triangleleft \ P) \rightsquigarrow \Delta_3 \vdash \overline{\mathsf{pt}} :_I \phi.$$

Moreover, in all cases above, let Δ *such that* $\Delta_1 \equiv_\alpha \Delta_0, \Delta$*, then* $\Delta_3 = \Delta_2, \Delta$*. In particular,* $\Delta_2 \geq \Delta_3$.

b) Soundness: We are now ready to show the soundness of our elaborator.

Proposition 4 (Soundness)**.**
The elaboration rules in Fig. 8 and Fig. 9, and the directed elaboration and re-elaboration rules in Fig. 10, are sound.

Proof. We prove this by induction on the typing derivation.

For elaboration and re-elaboration, each rule is immediately valid w.r.t. the semantics of the corresponding judgement. Directed elaboration is just elaboration followed by re-elaboration on the pattern.

Axioms rules.

$$\textsc{E.L.Ax}\ \frac{\Delta_0 = \mathbb{E}_0; \Gamma_0; \Lambda_0 \qquad (\mathsf{id}, \dot{\phi}) \in \Lambda_0}{\Delta_0 \vdash \mathsf{id} \rightsquigarrow \Delta_0 \vdash \mathsf{id} :_{\{\mathsf{id}\}} (\dot{\phi} \lessdot 0)} \qquad \textsc{E.G.Ax}\ \frac{\Delta_0 = \mathbb{E}_0; \Gamma_0; \Lambda_0 \qquad (\mathsf{id}, \tilde{\phi}) \in \Gamma_0}{\Delta_0 \vdash \mathsf{id} \rightsquigarrow \Delta_0 \vdash \mathsf{id} :_{\emptyset} \tilde{\phi}}$$

Localization rules.

$$\textsc{E.ByLoc}\ \frac{\Delta_0 \vdash (\mathsf{pt} \ \triangleleft\ (\_ \lessdot \_)) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_{\emptyset} (\dot{\phi} \lessdot \varepsilon)}{\Delta_0 \vdash [\mathsf{pt}] \rightsquigarrow \Delta_1 \vdash [\overline{\mathsf{pt}}] :_{\emptyset} [\dot{\phi}]_{\varepsilon}} \qquad \textsc{E.\%?}\ \frac{\Delta_0 \vdash (\mathsf{pt} \ \triangleleft\ (\_ \lessdot \_)) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_{I} (\dot{\phi} \lessdot \varepsilon)}{\Delta_0 \vdash \%? \ \mathsf{pt} \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_{I} (\dot{\phi} \lessdot \varepsilon)}$$

$$\textsc{E.ByGlob}\ \frac{\Delta_0 \vdash (\mathsf{pt} \ \triangleleft\ [\_]\_) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_{\emptyset} [\dot{\phi}]_{\varepsilon}}{\Delta_0 \vdash \% \ \mathsf{pt} \rightsquigarrow \Delta_1 \vdash \% \ \overline{\mathsf{pt}} :_{\emptyset} (\dot{\phi} \lessdot \varepsilon)}$$

Weakening rules.

$$\textsc{E.Weak}\ \frac{\Delta_0 \vdash (\mathsf{pt}_0 \ \triangleleft\ [\_ \leq \_]_0) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_0 :_{\emptyset} [\varepsilon \leq \bar{\varepsilon}]_0 \qquad \Delta_1 \vdash (\mathsf{pt}_1 \ \triangleleft\ (\_ \lessdot \varepsilon)) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_1 :_{I} (\dot{\phi} \lessdot \varepsilon)}{\begin{array}{l}\Delta_0 \vdash (\mathsf{pt}_1 \lessdot \bar{\varepsilon}(\mathsf{pt}_0)) \\ \quad \rightsquigarrow \Delta_2 \vdash (\overline{\mathsf{pt}}_1 \lessdot \bar{\varepsilon}(\overline{\mathsf{pt}}_0)) :_{I} (\dot{\phi} \lessdot \bar{\varepsilon})\end{array}}$$

$$\textsc{E.Weak.\_}\ \frac{\Delta_0 \vdash (\mathsf{pt} \ \triangleleft\ (\_ \lessdot \_)) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_{I} (\dot{\phi} \lessdot \varepsilon)}{\begin{array}{l}\Delta_0 \vdash (\mathsf{pt} \lessdot \bar{\varepsilon}(\_)) \\ \quad \rightsquigarrow \Delta_1, (\mathsf{id} : [\varepsilon \leq \bar{\varepsilon}]_0) \vdash (\overline{\mathsf{pt}} \lessdot \bar{\varepsilon}(\mathsf{id})) :_{I} (\dot{\phi} \lessdot \bar{\varepsilon})\end{array}}$$

Implication rules.

$$\textsc{E.}\dot{\Pi}\ \frac{\Delta_0, (\mathsf{id} : \dot{\phi}) \vdash (\mathsf{pt} \ \triangleleft\ (\_ \lessdot \_)) \rightsquigarrow \Delta_1, (\mathsf{id} : \dot{\phi}) \vdash \overline{\mathsf{pt}} :_{I} (\dot{\phi}_1 \lessdot \varepsilon)}{\begin{array}{l}\Delta_0 \vdash \dot{\Pi}(\mathsf{id} : \dot{\phi}_0).\ \mathsf{pt} \\ \quad \rightsquigarrow \Delta_1 \vdash \dot{\Pi}(\mathsf{id} : \dot{\phi}_0).\ \overline{\mathsf{pt}} :_{I \setminus \{\mathsf{id}\}} (\dot{\phi}_0 \dot{\Rightarrow} \dot{\phi}_1 \lessdot \varepsilon)\end{array}} \qquad \textsc{E.}\tilde{\Pi}\ \frac{\Delta_0, (\mathsf{id} : \tilde{\phi}_0) \vdash \mathsf{pt} \rightsquigarrow \Delta_1, (\mathsf{id} : \tilde{\phi}_0) \vdash \overline{\mathsf{pt}} :_{\emptyset} \tilde{\phi}_1}{\begin{array}{l}\Delta_0 \vdash \tilde{\Pi}(\mathsf{id} : \tilde{\phi}_0).\ \mathsf{pt} \\ \quad \rightsquigarrow \Delta_1 \vdash \tilde{\Pi}(\mathsf{id} : \tilde{\phi}_0).\ \overline{\mathsf{pt}} :_{\emptyset} \tilde{\phi}_0 \tilde{\Rightarrow} \tilde{\phi}_1\end{array}}$$

Universal quantification rules

In the rules below, $\forall(x : \tau).S$ is the set of hypotheses obtained from S by generalizing on x any hypothesis in which x appears freely. Further, if $\Delta = (\mathbb{E}; \Gamma; \Lambda)$, then $\forall x.\Delta = (\mathbb{E}; \forall x.\,\Gamma; \forall x.\,\Lambda)$ (note that we ignore $\mathbb{E}$, as generalizing there would be meaningless). Finally, the proof term $\mathsf{pt}_{\dot{\forall},\tilde{\forall}}$ is the proof term pt where the generalized hypotheses have been applied to x.

$$\textsc{E.L.}\lambda\ \frac{\begin{array}{c}\Delta_0, (x : \tau) \vdash \mathsf{pt} \rightsquigarrow \Delta_1, (x : \tau) \vdash \overline{\mathsf{pt}} :_{I} (\dot{\phi} \lessdot \varepsilon) \\ \tau \text{ is finite}\end{array}}{\begin{array}{l}\Delta_0 \vdash \lambda(x : \tau).\ \mathsf{pt} \\ \quad \rightsquigarrow \forall(x : \tau).\Delta_1 \vdash \dot{\lambda}(x : \tau).\ \overline{\mathsf{pt}}_{\dot{\forall},\tilde{\forall}} :_{I} (\dot{\forall}(x : \tau).\dot{\phi} \lessdot \mathsf{max}_x(\varepsilon))\end{array}} \qquad \textsc{E.}\dot{\lambda}\ \frac{\begin{array}{c}\Delta_0, (x : \tau) \vdash (\mathsf{pt} \ \triangleleft\ (\_ \lessdot \_)) \rightsquigarrow \Delta_1, (x : \tau) \vdash \overline{\mathsf{pt}} :_{I} (\dot{\phi} \lessdot \varepsilon) \\ \tau \text{ is finite}\end{array}}{\begin{array}{l}\Delta_0 \vdash \dot{\lambda}(x : \tau).\ \mathsf{pt} \\ \quad \rightsquigarrow \forall(x : \tau).\ \Delta_1 \vdash \dot{\lambda}(x : \tau).\ \overline{\mathsf{pt}}_{\dot{\forall},\tilde{\forall}} :_{I} (\dot{\forall}(x : \tau).\dot{\phi} \lessdot \mathsf{max}_x(\varepsilon))\end{array}}$$

$$\textsc{E.G.}\lambda\ \frac{\Delta_0, (x : \tau) \vdash \mathsf{pt} \rightsquigarrow \Delta_1, (x : \tau) \vdash \overline{\mathsf{pt}} :_{\emptyset} \tilde{\phi}}{\begin{array}{l}\Delta_0 \vdash \lambda(x : \tau).\ \mathsf{pt} \\ \quad \rightsquigarrow \forall(x : \tau).\ \Delta_1 \vdash \tilde{\lambda}(x : \tau).\ \overline{\mathsf{pt}}_{\dot{\forall},\tilde{\forall}} :_{\emptyset} \tilde{\forall}(x : \tau).\tilde{\phi}\end{array}} \qquad \textsc{E.}\tilde{\lambda}\ \frac{\Delta_0, (x : \tau) \vdash \mathsf{pt} \rightsquigarrow \Delta_1, (x : \tau) \vdash \overline{\mathsf{pt}} :_{\emptyset} \tilde{\phi}}{\begin{array}{l}\Delta_0 \vdash \tilde{\lambda}(x : \tau).\ \mathsf{pt} \\ \quad \rightsquigarrow \forall(x : \tau).\ \Delta_1 \vdash \tilde{\lambda}(x : \tau).\ \overline{\mathsf{pt}}_{\dot{\forall},\tilde{\forall}} :_{\emptyset} \tilde{\forall}(x : \tau).\phi\end{array}}$$

Figure 8: Basic, logic-specific, and constructor elaboration rules.

For example, we give the proof in the case of the rule E.I.G.$\llcorner\lrcorner$. By the induction hypothesis, we have that both those sequent are valid:

$$\Delta_0 \vdash \mathsf{pt}_0 \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_0 :_{\emptyset} \tilde{\phi}_0 \tilde{\Rightarrow} \tilde{\phi}$$

$$\Delta_1 \vdash (\mathsf{pt}_1 \ \triangleleft\ \tilde{\phi}_0) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_1 :_{\emptyset} \tilde{\phi}_0$$

Therefore, the typing judgements are well-typed:

$$\Delta_1 \vdash_{\emptyset} \overline{\mathsf{pt}}_0 : \tilde{\phi}_0 \tilde{\Rightarrow} \tilde{\phi} \tag{3}$$

$$\Delta_2 \vdash_{\emptyset} \overline{\mathsf{pt}}_1 : \tilde{\phi}_0 \tag{4}$$

By Lemma 1, we know that $\Delta_0 \geq \Delta_1 \geq \Delta_2$. Thus, we know that we can weaken the proof context in Eq. (3). Thus, the following typing judgement is also well-typed:

$$\Delta_2 \vdash_{\emptyset} \overline{\mathsf{pt}}_0 : \tilde{\phi}_0 \tilde{\Rightarrow} \tilde{\phi} \tag{5}$$

Since the proof contexts of Eq. (5) and Eq. (4) are identical, we can apply the typing rule Ty.I.$\tilde{\sqsubset}$, and we have that

$$\Delta_2 \vdash_{\emptyset} \overline{\mathsf{pt}}_0 \ \tilde{\sqsubset}\ \overline{\mathsf{pt}}_1 : \tilde{\phi}_0$$

is well-typed. Thus, the conclusion of E.I.G.$\llcorner\lrcorner$ is valid. □

C. Elaboration Strategy

The elaboration proof system we presented is not completely syntax directed, and there remains choices to be made during proof search. For example, assume that $\overline{\mathsf{pt}}$ prove $\Delta \vdash \tilde{\phi}_0 \tilde{\Rightarrow} \tilde{\phi}_1 \tilde{\Rightarrow} \tilde{\phi}_2$, then re-elaboration with the pattern $\_ \tilde{\Rightarrow} \_$ can yield any of two non-equivalent proof terms:

- the original proof term $\overline{\mathsf{pt}}$ proving $\Delta \vdash \tilde{\phi}_0 \tilde{\Rightarrow} \tilde{\phi}_1 \tilde{\Rightarrow} \tilde{\phi}_2$, by applying the rule RE.Conv;
- or $(\overline{\mathsf{pt}} \ \tilde{\sqsubset}\ H)$ proving $\Delta, \tilde{\phi}_0 \vdash \tilde{\phi}_1 \tilde{\Rightarrow} \tilde{\phi}_2$, by applying the rule RE.$[\cdot].\tilde{\Rightarrow}$ to discharge $\tilde{\phi}_0$ before applying RE.Conv.

Application rules: $\_\_$.

$$\textsc{E.I.L.}\_\_.\textsc{Right}\quad \frac{\Delta_0 \vdash \mathsf{pt}_0 \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_0 :_{I_0} (\dot{\phi}_0 \lessdot \varepsilon_0) \qquad \Delta_1 \vdash (\mathsf{pt}_1 \triangleleft (\dot{\phi}_0 \dot{\Rightarrow} \_ \lessdot \_)) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_1 :_{I_1} (\dot{\phi}_0 \dot{\Rightarrow} \dot{\phi} \lessdot \varepsilon_1)}{\Delta_0 \vdash \mathsf{pt}_1 \_\_ \mathsf{pt}_0 \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_1 \dot{\_\_} \overline{\mathsf{pt}}_0 :_{I_0 \cup I_1} (\dot{\phi} \lessdot \varepsilon_0 + \varepsilon_1)}$$

$$\textsc{E.I.L.}\_\_.\textsc{Left}\quad \frac{\Delta_0 \vdash \mathsf{pt}_1 \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_0} (\dot{\phi}_0 \dot{\Rightarrow} \dot{\phi} \lessdot \varepsilon_1) \qquad \Delta_1 \vdash (\mathsf{pt}_0 \triangleleft (\dot{\phi}_0 \lessdot \_)) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_0 :_{I_1} (\dot{\phi}_0 \lessdot \varepsilon_0)}{\Delta_0 \vdash \mathsf{pt}_1 \_\_ \mathsf{pt}_0 \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_1 \dot{\_\_} \overline{\mathsf{pt}}_0 :_{I_0 \cup I_1} (\dot{\phi} \lessdot \varepsilon_0 + \varepsilon_1)}$$

$$\textsc{E.I.G.}\_\_\quad \frac{\Delta_0 \vdash \mathsf{pt}_1 \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{\emptyset} \tilde{\phi}_0 \tilde{\Rightarrow} \tilde{\phi} \qquad \Delta_1 \vdash (\mathsf{pt}_0 \triangleleft \tilde{\phi}_0) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_0 :_{\emptyset} \tilde{\phi}_0}{\Delta_0 \vdash \mathsf{pt}_1 \_\_ \mathsf{pt}_0 \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_1 \tilde{\_\_} \overline{\mathsf{pt}}_0 :_{\emptyset} \tilde{\phi}}$$

Application rules: $\dot{\_\_}$ and $\tilde{\_\_}$.

$$\textsc{E.I.}\dot{\_\_}\quad \frac{\Delta_0 \vdash (\mathsf{pt}_0 \triangleleft (\_ \lessdot \_)) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_0 :_{I_0} (\dot{\phi}_0 \lessdot \varepsilon_0) \qquad \Delta_1 \vdash (\mathsf{pt}_1 \triangleleft (\dot{\phi}_0 \dot{\Rightarrow} \_ \lessdot \_)) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_1 :_{I_1} (\dot{\phi}_0 \dot{\Rightarrow} \dot{\phi} \lessdot \varepsilon_1)}{\Delta_0 \vdash \mathsf{pt}_1 \dot{\_\_} \mathsf{pt}_0 \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_1 \dot{\_\_} \overline{\mathsf{pt}}_0 :_{I_0 \cup I_1} (\dot{\phi} \lessdot \varepsilon_0 + \varepsilon_1)}$$

$$\textsc{E.I.}\tilde{\_\_}\quad \frac{\Delta_0 \vdash (\mathsf{pt}_1 \triangleleft \_ \tilde{\Rightarrow} \_) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{\emptyset} \tilde{\phi}_0 \tilde{\Rightarrow} \tilde{\phi} \qquad \Delta_1 \vdash (\mathsf{pt}_0 \triangleleft \tilde{\phi}_0) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_0 :_{\emptyset} \tilde{\phi}_0}{\Delta_0 \vdash \mathsf{pt}_1 \tilde{\_\_} \mathsf{pt}_0 \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 \tilde{\_\_} \overline{\mathsf{pt}}_0 :_{\emptyset} \tilde{\phi}}$$

Application rules with discharges.

$$\textsc{E.I.L.}\_\_.\_\quad \frac{\Delta_0 \vdash \mathsf{pt} \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_I (\dot{\phi}_0 \dot{\Rightarrow} \dot{\phi} \lessdot \varepsilon)}{\Delta_0 \vdash \mathsf{pt} \_\_ \_ \rightsquigarrow \Delta_1, (\mathsf{id} : \dot{\phi}_0) \vdash \overline{\mathsf{pt}} \dot{\_\_} \mathsf{id} :_{I \cup \mathsf{id}} (\dot{\phi} \lessdot \varepsilon)}$$

$$\textsc{E.I.G.}\_\_.\_\quad \frac{\Delta_0 \vdash \mathsf{pt} \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_{\emptyset} \tilde{\phi}_0 \tilde{\Rightarrow} \tilde{\phi}}{\Delta_0 \vdash \mathsf{pt} \_\_ \_ \rightsquigarrow \Delta_1, (\mathsf{id} : \tilde{\phi}_0) \vdash \overline{\mathsf{pt}} \tilde{\_\_} \mathsf{id} :_{\emptyset} \tilde{\phi}}$$

$$\textsc{E.I.}\dot{\_\_}.\_\quad \frac{\Delta_0 \vdash (\mathsf{pt} \triangleleft (\_ \dot{\Rightarrow} \_ \lessdot \_)) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_I (\dot{\phi}_0 \dot{\Rightarrow} \dot{\phi} \lessdot \varepsilon)}{\Delta_0 \vdash \mathsf{pt} \dot{\_\_} \_ \rightsquigarrow \Delta_1, (\mathsf{id} : \dot{\phi}_0) \vdash \overline{\mathsf{pt}} \dot{\_\_} \mathsf{id} :_{I \cup \mathsf{id}} (\dot{\phi} \lessdot \varepsilon)}$$

$$\textsc{E.I.}\tilde{\_\_}.\_\quad \frac{\Delta_0 \vdash (\mathsf{pt} \triangleleft \_ \tilde{\Rightarrow} \_) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_{\emptyset} \tilde{\phi}_0 \tilde{\Rightarrow} \tilde{\phi}}{\Delta_0 \vdash \mathsf{pt} \tilde{\_\_} \_ \rightsquigarrow \Delta_1, (\mathsf{id} : \tilde{\phi}_0) \vdash \overline{\mathsf{pt}} \tilde{\_\_} \mathsf{id} :_{\emptyset} \tilde{\phi}}$$

Term application rules: $\_\_$.

$$\textsc{E.T.L.}\_\_\quad \frac{\mathbb{E} \vdash t : \tau \qquad \Delta_0 \vdash (\mathsf{pt} \triangleleft [\dot{\forall}(\_ : \tau).\_ ; (\dot{\forall}(\_ : \tau).\_ \lessdot \_)]) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_I (\dot{\forall}(x : \tau).\dot{\phi} \lessdot \varepsilon)}{\Delta_0 \vdash \mathsf{pt} \_\_ t \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} \dot{\_\_} t :_I (\dot{\phi}[x \mapsto t] \lessdot \varepsilon)}$$

$$\textsc{E.T.G.}\_\_\quad \frac{\mathbb{E} \vdash t : \tau \qquad \Delta_0 \vdash (\mathsf{pt} \triangleleft [\tilde{\forall}(\_ : \tau).\_ ; (\dot{\forall}(\_ : \tau).\_ \lessdot \_)]) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_{\emptyset} \tilde{\forall}(x : \tau).\tilde{\phi}}{\Delta_0 \vdash \mathsf{pt} \_\_ t \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} \tilde{\_\_} t :_{\emptyset} \tilde{\phi}[x \mapsto t]}$$

Term application rules: $\dot{\_\_}$ and $\tilde{\_\_}$.

$$\textsc{E.T.}\dot{\_\_}\quad \frac{\mathbb{E} \vdash t : \tau \qquad \Delta_0 \vdash (\mathsf{pt} \triangleleft (\dot{\forall}(\_ : \tau).\_ \lessdot \_)) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_I (\dot{\forall}(x : \tau).\dot{\phi} \lessdot \varepsilon)}{\Delta_0 \vdash \mathsf{pt} \dot{\_\_} t \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} \dot{\_\_} t :_I (\dot{\phi}[x \mapsto t] \lessdot \varepsilon)}$$

$$\textsc{E.T.}\tilde{\_\_}\quad \frac{\mathbb{E} \vdash t : \tau \qquad \Delta_0 \vdash (\mathsf{pt} \triangleleft \tilde{\forall}(\_ : \tau).\_) \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} :_{\emptyset} \tilde{\forall}(x : \tau).\tilde{\phi}}{\Delta_0 \vdash \mathsf{pt} \tilde{\_\_} t \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}} \tilde{\_\_} t :_{\emptyset} \tilde{\phi}[x \mapsto t]}$$

Figure 9: Applications elaboration rules.

To make elaboration effective and implement it, we must fix a proof search strategy. This strategy must decide which rule to apply for any given inputs, and for any of the three elaboration modes — (simple) elaboration, directed elaboration, and re-elaboration. It turns out that while the rules of (simple) elaboration are not syntax directed, any application order yield the same behavior (we will prove this later, in Proposition 6). Further, the rules of directed elaboration already resolved all possible sources of non-determinism. We are left with re-elaboration. Here, there is one possible source of non-determinism during a re-elaboration of $(\Delta \vdash (\mathsf{pt} : \phi \triangleleft P))$ that we must resolve: at all time, we may apply the RE.Conv rule to match ϕ with P; or we may apply another rule depending on the shape of the pattern P.

Our strategy applies the conversion rule eagerly: at each step during proof search, if the conclusion ϕ matches P, we apply RE.Conv and conclude. This strategy is both simple and natural, as it stops proof search as soon as possible. From now one, we always consider elaborations following this strategy.

This strategy does not yet yield a deterministic procedure, as there are some scenarios where multiple rules can be applied. Nonetheless, we will show that different rule choices yield identical outputs: thus, the choice of rule by an implementation is irrelevant. To characterize the fact that the remaining apparent non-determinism leaves the output unchanged, we show that any two derivations starting from the same input yield identical outputs.

Because of binders, this only holds up-to α-renaming of the variables and hypothesis identifiers in the output judgements. More precisely, we lift the notion of α equivalence from formulas, contexts and proofs terms to judgements as follows: two judgements $\Delta_0 \vdash_{I_0} \overline{\mathsf{pt}}_0 : \phi_0$ and $\Delta_1 \vdash_{I_1} \overline{\mathsf{pt}}_1 : \phi_1$ are

Directed elaboration rules.

$$\textsc{DE.RE}\quad \frac{\Delta_0 \vdash \mathsf{pt} \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1 \qquad \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1 \triangleleft P \twoheadrightarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2}{\Delta_0 \vdash (\mathsf{pt} \ \triangleleft \ P) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2}$$

$$\textsc{DE.Head}\quad \frac{\Delta_0 \vdash \mathsf{pt} \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1 \qquad \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1 \triangleleft P \twoheadrightarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2}{\Delta_0 \vdash (\mathsf{pt} \ \triangleleft \ P :: L) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2}$$

$$\textsc{DE.Tail}\quad \frac{\Delta_0 \vdash \mathsf{pt} \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1 \qquad \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1 \not\triangleleft P \qquad \Delta_0 \vdash (\mathsf{pt} \ \triangleleft \ L) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2}{\Delta_0 \vdash (\mathsf{pt} \ \triangleleft \ P :: L) \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2}$$

Re-elaboration: basic and congruence rules.

$$\textsc{RE.Conv}\quad \frac{\mathbb{E}_0 \vdash \phi \overset{?}{\leq} P}{\Delta_0 \vdash \overline{\mathsf{pt}} :_I \phi \triangleleft P \twoheadrightarrow \Delta_0 \vdash \overline{\mathsf{pt}} :_I \phi}$$

$$\textsc{RE.Congr.}\Rightarrow\quad \frac{\mathbb{E}_0 \vdash \tilde{\phi} \overset{?}{\leq} P \qquad \Delta_0, (\mathsf{id} : \tilde{\phi}) \vdash \overline{\mathsf{pt}}_0 \triangleq \mathsf{id} :_I \tilde{\phi}_0 \triangleleft P_0 \twoheadrightarrow \Delta_1, (\mathsf{id} : \tilde{\phi}) \vdash \overline{\mathsf{pt}}_1 :_I \tilde{\phi}_1}{\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_I \tilde{\phi} \Rightarrow \tilde{\phi}_0 \triangleleft P \Rightarrow P_0 \twoheadrightarrow \Delta_1 \vdash \tilde{\Pi}(\mathsf{id} : \tilde{\phi}).\ \overline{\mathsf{pt}}_1 :_I \tilde{\phi} \Rightarrow \tilde{\phi}_1}$$

$$\textsc{RE.Congr.}\tilde{\forall}\quad \frac{\Delta_0, (x : \tau) \vdash \overline{\mathsf{pt}}_0 \triangleq x :_I \tilde{\phi}_0 \triangleleft P_0 \twoheadrightarrow \Delta_1, (x : \tau) \vdash \overline{\mathsf{pt}}_1 :_I \tilde{\phi}_1}{\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_I \tilde{\forall}(x : \tau).\tilde{\phi}_0 \triangleleft \tilde{\forall}(x : \tau).P \twoheadrightarrow \forall(x : \tau).\Delta_1 \vdash \tilde{\lambda}(x : \tau).\ \overline{\mathsf{pt}}_1 :_I \tilde{\forall}(x : \tau).\tilde{\phi}_1}$$

Re-elaboration: locality rules.

$$\textsc{RE.ByLoc}\quad \frac{\Delta_0 \vdash \%\,\overline{\mathsf{pt}}_0 :_\emptyset (\dot{\phi}_0 \lessdot \varepsilon_0) \triangleleft (\dot{\phi}_1 \lessdot \varepsilon_1) \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_I (\dot{\phi}_2 \lessdot \varepsilon_2)}{\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_\emptyset [\dot{\phi}_0]_{\varepsilon_0} \triangleleft (\dot{\phi}_1 \lessdot \varepsilon_1) \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_I (\dot{\phi}_2 \lessdot \varepsilon_2)}$$

$$\textsc{RE.}\dot{\forall}.\tilde{\forall}\quad \frac{\Delta_0, (x : \tau) \vdash [\%\,\overline{\mathsf{pt}}_0 \triangleq x] :_\emptyset [\dot{\phi}]_\varepsilon \triangleleft \tilde{\phi} \twoheadrightarrow \Delta_1, (x : \tau) \vdash \overline{\mathsf{pt}}_1 :_\emptyset \tilde{\phi}_1}{\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_\emptyset [\dot{\forall}(x : \tau).\dot{\phi}]_\varepsilon \triangleleft \tilde{\forall}(x : \tau).\tilde{\phi} \twoheadrightarrow \forall(x : \tau).\Delta_1 \vdash \tilde{\lambda}x : \tau.\ \overline{\mathsf{pt}}_1 :_\emptyset \tilde{\forall}(x : \tau).\tilde{\phi}_1}$$

Re-elaboration: weakening rules. *(Recall that ε_0 is either a hole $\_$ or a term without hole.)*

$$\textsc{RE.Weak}\quad \frac{\Delta_0 = \mathbb{E}_0; \Gamma_0; \Lambda_0 \qquad \mathbb{E}_0 \vdash \dot{\phi} \overset{?}{\leq} \dot{\phi}_0 \qquad \mathbb{E}_0 \vdash \varepsilon \overset{?}{\not\leq} \varepsilon_0}{\Delta_0 \vdash \overline{\mathsf{pt}} :_I (\dot{\phi} \lessdot \varepsilon) \triangleleft (\dot{\phi}_0 \lessdot \varepsilon_0) \twoheadrightarrow \Delta_0, (\mathsf{id} : [\varepsilon \leq \varepsilon_0]_0) \vdash (\overline{\mathsf{pt}} \lessdot \varepsilon_0(\mathsf{id})) :_I (\dot{\phi} \lessdot \varepsilon_0)}$$

$$\textsc{RE.}[\cdot].\textsc{Weak}\quad \frac{\Delta_0 = \mathbb{E}_0; \Gamma_0; \Lambda_0 \qquad \mathbb{E}_0 \vdash \dot{\phi} \overset{?}{\leq} \dot{\phi}_0 \qquad \mathbb{E}_0 \vdash \varepsilon \overset{?}{\not\leq} \varepsilon_0}{\Delta_0 \vdash \overline{\mathsf{pt}} :_\emptyset [\dot{\phi}]_\varepsilon \triangleleft [\dot{\phi}_0]_{\varepsilon_0} \twoheadrightarrow \Delta_0, (\mathsf{id} : [\varepsilon \leq \varepsilon_0]_0) \vdash [(\%\,\overline{\mathsf{pt}} \lessdot \varepsilon_0(\mathsf{id}))] :_\emptyset [\dot{\phi}]_{\varepsilon_0}}$$

Re-elaboration: implication rules. *(Automatically introduce a global hypothesis in the proof context when looking for a local formula $\dot{\phi}_0$.)*

$$\textsc{RE.}[\cdot].\Rightarrow\quad \frac{\Delta_0, (\mathsf{id} : \tilde{\phi}_0) \vdash \overline{\mathsf{pt}}_0 \triangleq \mathsf{id} :_\emptyset \tilde{\phi}_1 \triangleleft [\dot{\phi}_0]_{\varepsilon_0} \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_\emptyset [\dot{\phi}_1]_{\varepsilon_1}}{\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_\emptyset \tilde{\phi}_0 \Rightarrow \tilde{\phi}_1 \triangleleft [\dot{\phi}_0]_{\varepsilon_0} \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_\emptyset [\dot{\phi}_1]_{\varepsilon_1}}$$

$$\textsc{RE.L.}\Rightarrow\quad \frac{\Delta_0, (\mathsf{id} : \tilde{\phi}_0) \vdash \overline{\mathsf{pt}}_0 \triangleq \mathsf{id} :_\emptyset \tilde{\phi}_1 \triangleleft (\dot{\phi}_0 \lessdot \varepsilon_0) \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_\emptyset (\dot{\phi}_1 \lessdot \varepsilon_1)}{\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \overline{\mathsf{pt}}_0 :_\emptyset \tilde{\phi}_0 \Rightarrow \tilde{\phi}_1 \triangleleft (\dot{\phi}_0 \lessdot \varepsilon_0) \twoheadrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_\emptyset (\dot{\phi}_1 \lessdot \varepsilon_1)}$$

Re-elaboration: forall rules. *(Automatically open a global universal quantification when looking for a local formula $\dot{\phi}_0$.)*

$$\textsc{RE.}[\cdot].\tilde{\forall}\quad \frac{\Delta_0, (x : \tau) \vdash \overline{\mathsf{pt}}_0 \triangleq x :_\emptyset \tilde{\phi}_0 \triangleleft [\dot{\phi}_0]_{\varepsilon_0} \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_\emptyset [\dot{\phi}_1]_{\varepsilon_1}}{\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_\emptyset \tilde{\forall}(x : \tau).\tilde{\phi}_0 \triangleleft [\dot{\phi}_0]_{\varepsilon_0} \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_\emptyset [\dot{\phi}_1]_{\varepsilon_1}}$$

$$\textsc{RE.L.}\tilde{\forall}\quad \frac{\Delta_0, (x : \tau) \vdash \overline{\mathsf{pt}}_0 \triangleq x :_\emptyset \tilde{\phi}_0 \triangleleft (\dot{\phi}_0 \lessdot \varepsilon_0) \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_I (\dot{\phi}_1 \lessdot \varepsilon_1)}{\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_\emptyset \tilde{\forall}(x : \tau).\tilde{\phi}_0 \triangleleft (\dot{\phi}_0 \lessdot \varepsilon_0) \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_I (\dot{\phi}_1 \lessdot \varepsilon_1)}$$

Figure 10: Directed elaboration and re-elaboration rules.

α-*equivalent*, when there exists a renaming σ of the variables and hypothesis identifiers in Δ_0 such that:

$$\Delta_0\sigma \equiv_\alpha \Delta_1 \qquad I_0\sigma \equiv_\alpha I_1 \qquad \overline{\mathsf{pt}}_0\sigma \equiv_\alpha \overline{\mathsf{pt}}_1 \qquad \phi_0\sigma \equiv_\alpha \phi_1.$$

We are now ready to prove that our strategy yields a deterministic procedure for all our elaboration modes. We first show that re-elaboration is deterministic, which can be done without considering the other two modes (Proposition 5). Then, we use this to show that elaboration is deterministic (Proposition 6). For our proofs to go through, we must generalize both results, and prove that α-equivalent inputs elaborates to α-equivalent outputs.

a) Re-elaboration: We start with the α-stability of re-elaboration. As a first useful observation, we note that a re-elaboration $(\Delta \vdash (\mathsf{pt} : \phi \triangleleft P))$ does not depend on the proof context Δ nor the proof term pt, in the sense that the *applicability* of a re-elaboration rule only depends on ϕ and P. This remark is limited to the rule's applicability: for example, the *output* of the rule — which contains the possibly modified proof context and proof term — depends on Δ and pt.

Proposition 5 (α-stability of re-elaboration)**.** *If the two typing judgements $\Delta_0 \vdash_{I_0} \overline{\mathsf{pt}}_0 : \phi_0$ and $\Delta_1 \vdash_{I_1} \overline{\mathsf{pt}}_1 : \phi_1$ are α-equivalent, the two pattern P_0 and P_1 are α-equivalent, and the following re-elaboration is derivable:*

$$\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_{I_0} \phi_0 \triangleleft P_0 \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2$$

then

1) $\Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1 \triangleleft P_1 \rightsquigarrow \Delta_3 \vdash \overline{\mathsf{pt}}_3 :_{I_3} \phi_3$ *is also derivable*
2) *the two outputted typing judgements are α-equivalent.*

Proof.
Proof of point 1:
We can check that all rules are α-stable, in the sense that a rule applies to a typing judgement iff it applies to an α-equivalent typing judgement, and it yields α-equivalent output typing judgements.

Therefore, if two typing judgements $\Delta_0 \vdash_{I_0} \overline{\mathsf{pt}}_0 : \phi_0$ and $\Delta_1 \vdash_{I_1} \overline{\mathsf{pt}}_1 : \phi_1$ are α-equivalent, and if the former can be re-elaborated, then we can replay its re-elaboration derivation to the latter judgement. (This is an easy induction on the former derivation.)

Proof of point 2:
In the previous point, we showed that a re-elaboration can be replayed on any α-equivalent typing judgement to obtain an α-equivalent output. Now, we prove that *any* two re-elaborations following our strategy and starting from α-equivalent inputs yield α-equivalent outputs.

We prove this by induction on the maximum of the size of the re-elaboration derivations of $\Delta_0 \vdash_{I_0} \overline{\mathsf{pt}}_0 : \phi_0$ and $\Delta_1 \vdash_{I_1} \overline{\mathsf{pt}}_1 : \phi_1$. First, we are going to show that the same rule must have applied to both sides. In a second time, we will prove that when this is the case, the outputs must be α-equivalent.

The same rule must apply: With the exception of RE.CONV, we first check that there is only one rule that can be applied at the top-level of both derivations. To show that, we list below, for each re-elaboration rule except conversion, the constraints a rule puts on the shape of the input formula ϕ and the pattern P. We describe our notation on an example: $(\cdot \dot{\Rightarrow} \cdot \triangleleft [\_]\_)$ means that ϕ must be of the form $\phi_1 \dot{\Rightarrow} \phi_2$, and that P must be of the form $[P_1]_{P_2}$ — the sake of readability, we denote formula's holes using $\cdot$, and use a different notation $\cdot$ for pattern's holes.

Rule	Formula		Pattern
RE.CONGR.$\dot{\Rightarrow}$	$\cdot \dot{\Rightarrow} \cdot$	$\triangleleft$	$\_ \dot{\Rightarrow} \_$
RE.CONGR.$\tilde{\forall}$	$\tilde{\forall}(\cdot).\cdot$	$\triangleleft$	$\tilde{\forall}(\_).\_$
RE.BYLOC	$[\,\cdot\,].$	$\triangleleft$	$(\_ \lessdot \_)$
RE.$\dot{\forall}.\tilde{\forall}$	$[\dot{\forall}(\cdot).\,\cdot\,].$	$\triangleleft$	$\tilde{\forall}(\_).\_$
RE.WEAK	$(\cdot \lessdot \cdot)$	$\triangleleft$	$(\_ \lessdot \_)$
RE.$[\cdot]$.WEAK	$[\,\cdot\,].$	$\triangleleft$	$[\_]\_$
RE.$[\cdot].\dot{\Rightarrow}$	$\cdot \dot{\Rightarrow} \cdot$	$\triangleleft$	$[\_]\_$
RE.L.$\dot{\Rightarrow}$	$\cdot \dot{\Rightarrow} \cdot$	$\triangleleft$	$(\_ \lessdot \_)$
RE.$[\cdot].\tilde{\forall}$	$\tilde{\forall}(\cdot).\cdot$	$\triangleleft$	$[\_]\_$
RE.L.$\tilde{\forall}$	$\tilde{\forall}(\cdot).\cdot$	$\triangleleft$	$(\_ \lessdot \_)$

We observe that there is not overlap possible: given a formula ϕ and a pattern P, only a single of the rule above may apply.

The constraints above are stable by α-renaming: thus, if both derivations do not start with conversion, the same rule must apply on both sides.

Further, if RE.CONV is applicable to one input, then it is applicable to the other (because matching is stable by α-renaming). Because our strategy eagerly applies conversion, this means that the same rule must apply to both inputs.

Outputs are α-equivalent: It remains to show that the outputs must be α-equivalent. If we applied RE.CONV, we have the α-equivalence form the hypothesis directly. Otherwise, we get it form the induction hypothesis and using the fact that for all rules that are different form conversion, if the premises are α-equivalent, then the conclusion are also α-equivalent. □

b) Elaboration and directed elaboration: We now show the α-stability of elaboration and directed elaboration. We start with a technical lemma on the re-elaboration of proof term that yield local formula. To state this lemma, we generalize the notion of α-equivalence w.r.t. some proof context: we say that $(\Delta_0, \mathsf{obj}_0^1, \ldots, \mathsf{obj}_0^n)$ and $(\Delta_1, \mathsf{obj}_1^1, \ldots, \mathsf{obj}_1^n)$ are α-equivalent when there exists a renaming σ of the variables and hypothesis identifiers in Δ_0 such that

$$\Delta_0\sigma \equiv_\alpha \Delta_1 \qquad \mathsf{obj}_0^i\sigma \equiv_\alpha \mathsf{obj}_0^i \quad \text{for any } i$$

For example, $(\Delta_0, \varepsilon_0, \overline{\mathsf{pt}}_0)$ and $(\Delta_1, \varepsilon_1, \overline{\mathsf{pt}}_1)$ are α-equivalent if

$$\Delta_0\sigma \equiv_\alpha \Delta_1 \qquad \varepsilon_0\sigma \equiv_\alpha \varepsilon_1 \qquad \overline{\mathsf{pt}}_0\sigma \equiv_\alpha \overline{\mathsf{pt}}_1$$

Lemma 3 (Characterization of local re-elaboration)**.** *If*

$$\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_{I_0} (\dot{\phi}_0 \lessdot \varepsilon_0) \triangleleft P \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1$$

then there exists $\dot{\phi}_1$ and ε_1 such as $\phi_1 = (\dot{\phi}_1 \lessdot \varepsilon_1)$ and $\dot{\phi}_0$ is α-equivalent to $\dot{\phi}_1$.

Moreover, when P is of the form $(\dot{\phi} \lessdot \_)$, we also have that $\Delta_0 \vdash_{I_0} \overline{\mathsf{pt}}_0 : (\dot{\phi}_0 \lessdot \varepsilon_0)$ and $\Delta_1 \vdash_{I_1} \overline{\mathsf{pt}}_1 : \phi_1$ are α-equivalent.

Proof. For the first part, we check that only the rules RE.CONV and RE.WEAK apply. The former leaves the conclusion unchanged, while the later only changes the bound: in both cases, we have the wanted result.

For the second part of the lemma, we observe that the additional constraint that P is of the form $(\dot{\phi} \lessdot \_)$ prevents that the rule RE.WEAK applies. Thus, only RE.CONV may apply, which trivially yields the wanted result. □

Proposition 6 (α-stability of the elaboration)**.**
If Δ_0, pt_0 and Δ_1, pt_1 are α-equivalent, and the following elaboration is derivable:

$$\Delta_0 \vdash \mathsf{pt}_0 \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_0} \phi_2$$

then

1) *$\Delta_1 \vdash \mathsf{pt}_1 \rightsquigarrow \Delta_3 \vdash \overline{\mathsf{pt}}_3 :_{I_1} \phi_3$ is also derivable*
2) *the two underlining typing judgement are α-equivalent.*

Proof.
Proof of point 1:
The proof goes as for point 1 of Proposition 5. We observe that all rules are α-stable (i.e. a rule applies iff it applies to an α-equivalent proof term, and both applications yield α-equivalent outputs), and prove by induction over the elaboration derivation of pt_0 that pt_1 can be elaborated to an α-equivalent output.

Proof of point 2:
It remains to show that *any* elaboration of pt_1 yields an α-equivalent output, which takes more work. We prove this by induction on maximum of the size of the elaboration derivations of pt_0 and pt_1.

We proceed by case analysis on the last rule applied in the elaboration of pt_0. We classify all the elaboration rules in three groups (I, II, and III), according to the kind of argument we use to prove our result.

I. Unique proof term shape

E.BYLOC E.%? E.BYGLOB E.WEAK

E.WEAK._ E.$\dot{\Pi}$ E.$\tilde{\Pi}$ E.$\dot{\lambda}$ E.$\tilde{\lambda}$ E.I.$\dot{\sqcup}$

E.I.$\tilde{\sqcup}$ E.I.$\dot{\sqcup}$._ E.I.$\tilde{\sqcup}$._ E.T.$\dot{\sqcup}$ E.T.$\tilde{\sqcup}$

This group contains the rules that uniquely characterize the shape of the input proof term, meaning that if a rule of this group applies to some proof term pt, then no other rule may apply to pt. For example, the rule E.I.$\dot{\sqcup}$._ is the only rule that applies to a proof term of the form pt $\dot{\sqcup}$ _.

Thus, if the last rule applied to pt_0 belongs to this group, the same rule must apply to pt_1. Further, as all rules are α-stable (as mentioned in point 1 of this proof), we know that pt_0 and pt_1 elaborates to α-equivalent outputs.

II. Overlap between two rules

(E.L.AX, E.G.AX) (E.L.λ, E.G.λ)

(E.T.L.$\sqcup$, E.T.G.$\sqcup$) (E.I.L.$\sqcup$._, E.I.G.$\sqcup$._)

Those pairs of rules may apply to the same input proof term when considering the shape of the input proof term alone. For example, the rules E.L.λ and E.G.λ, may both apply to a proof term of the shape $\lambda(x : \tau)$. pt. Thus, we need an additional argument to ensure that only one of these rules may apply to two α-equivalent proof term.

For the pair (E.L.AX, E.G.AX), this is easy, as the identifier id may appear only once in the proof context, and α-renaming preserves the kind of id (i.e. whether id appears in Γ or Λ).

For the remaining rules, we observe that for any pair $(\mathsf{R}_0, \mathsf{R}_1)$ of rules in the group II (excluding the axioms rules that we already dealt with), both rules have an elaboration premise with identical inputs but incompatible outputs, in the sense that one rule, say R_i, has an elaboration premise whose conclusion is local while the other rule R_{1-i} has an elaboration premise with identical inputs by a global conclusion. By the induction hypothesis, since both premises have α-equivalent inputs, they must have α-equivalent outputs. Since a local and global formulas are not α-equivalent, we deduce that the same rule must apply to pt_0 and pt_1.

III. Triple rule overlap

Those three rule can overlap each others:

E.I.L.$\sqcup$.RIGHT E.I.L.$\sqcup$.LEFT E.I.G.$\sqcup$

We will show that the property still hold, for all cases:

- *Overlap between* E.I.L.$\sqcup$.LEFT *and* E.I.G.$\sqcup$*:* The proof is identical as in group II.

For the others two cases, we use Lemma 3.

- *Overlap between* E.I.L.$\sqcup$.RIGHT *and* E.I.G.$\sqcup$:
  Cannot happens thanks to the lemma. Indeed, if this was possible for a proof term $\mathsf{pt} = \mathsf{pt}_2 \sqcup \mathsf{pt}_3$.
  Then since the rule E.I.L.$\sqcup$.RIGHT apply, we have a derivation of $\Delta_3 \vdash \mathsf{pt}_3 \rightsquigarrow \Delta_4 \vdash \overline{\mathsf{pt}}_4 :_{I_4} (\dot{\phi}_4 \lessdot \varepsilon_4)$.
  And with the derivation of E.I.G.$\sqcup$ (since the rule DE.RE is the only one that apply for directed elaboration), we get the derivations

$$\Delta_3 \vdash \mathsf{pt}_3 \rightsquigarrow \Delta_5 \vdash \overline{\mathsf{pt}}_5 :_{I_5} \phi_5$$
$$\Delta_5 \vdash \overline{\mathsf{pt}}_5 :_{I_5} \phi_5 \lhd \tilde{\phi}_0 \twoheadrightarrow \Delta_6 \vdash \overline{\mathsf{pt}}_6 :_{I_6} \tilde{\phi}_6 \qquad (6)$$

  Moreover, by the induction hypothesis, we have that $(\dot{\phi}_4 \lessdot \varepsilon_4)$ is α-equivalent to ϕ_5: thus ϕ_5 is local and cannot be re-elaborated to a global formula (Lemma 3), contradicting the derivation in Eq. (6).
  Thus both rules may not overlap. We conclude by a reasoning similar to case I.
- *Overlap between* E.I.L.$\sqcup$.RIGHT *and* E.I.L.$\sqcup$.LEFT: Here, the overlap can really happens and lead to two different proof trees. However, those proof trees yield the same formula up to α-renaming. Indeed, assuming that $\mathsf{pt}_0 = \mathsf{pt}^1_{\mathsf{R}} \sqcup \mathsf{pt}^0_{\mathsf{R}}$ is elaborated using the rule E.I.L.$\sqcup$.RIGHT, and $\mathsf{pt}_1 = \mathsf{pt}^1_{\mathsf{L}} \sqcup \mathsf{pt}^0_{\mathsf{L}}$ is elaborated using the rule E.I.L.$\sqcup$.LEFT. Then we have that:
  1) $\mathsf{pt}^0_{\mathsf{R}}$ and $\mathsf{pt}^0_{\mathsf{L}}$ are α-equivalent.
  2) $\mathsf{pt}^1_{\mathsf{R}}$ and $\mathsf{pt}^1_{\mathsf{L}}$ are α-equivalent.

3) $\Delta_0 \vdash \mathsf{pt}^0_{\mathsf{R}} \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} (\dot\phi_2 \lessdot \varepsilon_2)$ is derivable.
4) we can derive:

$$\Delta_2 \vdash \mathsf{pt}^1_{\mathsf{R}} \rightsquigarrow \Delta_4 \vdash (\overline{\mathsf{pt}}_4 \triangleleft (\dot\phi_2 \dot\Rightarrow \_ \lessdot \_)) :_{I_4} (\dot\phi_2 \dot\Rightarrow \dot\phi_4 \lessdot \varepsilon_4)$$

5) $\Delta_1 \vdash \mathsf{pt}^1_{\mathsf{L}} \rightsquigarrow \Delta_3 \vdash \overline{\mathsf{pt}}_3 :_{I_3} (\dot\phi_3 \dot\Rightarrow \dot\phi_5 \lessdot \varepsilon_3)$ is derivable.
6) $\Delta_3 \vdash \mathsf{pt}^0_{\mathsf{L}} \rightsquigarrow \Delta_5 \vdash (\overline{\mathsf{pt}}_5 \triangleleft (\dot\phi_3 \lessdot \_)) :_{I_5} (\dot\phi_3 \lessdot \varepsilon_5)$ is derivable.

Which with similar reasoning to unfold the directed elaboration that in the previous case, and using Lemma 3, we get that $(\Delta_4, \overline{\mathsf{pt}}_2, \overline{\mathsf{pt}}_4, (\dot\phi_2 \lessdot \varepsilon_2), (\dot\phi_2 \dot\Rightarrow \dot\phi_4 \lessdot \varepsilon_4))$ is α-equivalent to $(\Delta_5, \overline{\mathsf{pt}}_5, \overline{\mathsf{pt}}_3, (\dot\phi_3 \lessdot \varepsilon_5), (\dot\phi_3 \dot\Rightarrow \dot\phi_5 \lessdot \varepsilon_3))$. This give us the α-equivalence of the wanted typing judgements. □

D. Predictability

Soundness is not the only property we expect of an elaboration procedure. Notably, we expect that elaboration fills the gaps in a proof term, and does not produces a new proof term out of the blue. For example, in the context

$$\mathsf{H}_0 : \dot\phi_0, \quad \mathsf{G} : \dot\phi_0 \dot\Rightarrow \dot\phi_1, \quad \mathsf{H}_2 : \dot\phi_2$$

we do not expect the elaboration of $(\mathsf{G}\ \mathsf{H}_0)$ to yield $\dot\phi_2$, even though this would be sound. We expect elaboration to be *predictable*. We capture the notion of predictability by over-approximating the impact that elaboration can have on the input proof term.

To that end, define in Fig. 11 a rewriting relation $\overset{+}{\rightsquigarrow}$ such that any elaboration $\overline{\mathsf{pt}}_1$ of a proof term $\overline{\mathsf{pt}}_0$ verifies $\overline{\mathsf{pt}}_0 \overset{+}{\rightsquigarrow} \overline{\mathsf{pt}}_1$. Intuitively, this relation ensures that $\overline{\mathsf{pt}}_1$ can be obtained from $\overline{\mathsf{pt}}_0$ through a number of successive operations. The description of these operations provides a precise characterization of the modifications that elaboration may make to a proof term. Roughly, these operations consists in adding missing annotations, adding or removing locality constructs, and inserting applications and generalization.

We now describe $\overset{+}{\rightsquigarrow}$ in more detail. A first group of operations add locality annotations, e.g., $\overset{+}{\rightsquigarrow}.\tilde\lambda$ replaces $\lambda(x : \tau).\ \mathsf{pt}$ with $\tilde\lambda(x : \tau).\ \mathsf{pt}$. The rules $\overset{+}{\rightsquigarrow}.\%?$ and $\overset{+}{\rightsquigarrow}.\%$ respectively add or remove locality modalities. The rule $\overset{+}{\rightsquigarrow}.[\,\cdot\,].\dot\forall$ inserts a proof term application below a $[\,\cdot\,]$ modality by exploiting the $[\dot\forall.\ \cdot\ ]/\tilde\forall.[\,\cdot\,]$ commutations. Then, we have rules weakening the bound of, respectively, a local formula ($\overset{+}{\rightsquigarrow}.\textsc{Weak}$) and a global $[\,\cdot\,].$ formula ($\overset{+}{\rightsquigarrow}.[\,\cdot\,].\textsc{Weak}$). We have rules inserting proof term applications ($\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde\subset.\mathsf{H}$) or term applications ($\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde\subset.\mathsf{T}$ and $\overset{+}{\rightsquigarrow}.\mathrm{I}.\dot\subset.\mathsf{T}$), and rules inserting term ($\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde\lambda$) or proof term generalizations ($\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde\Pi$). The rules $\overset{+}{\rightsquigarrow}.\textsc{Weak.Name}._$ and $\overset{+}{\rightsquigarrow}.\textsc{App.Name}._$ replace a proof term hole by an internal identifier. Finally, we have the identity rule $\overset{+}{\rightsquigarrow}.\textsc{Id}$.

Remark 2. *If we see the proof terms as their syntax tree (where the annotation $\cdot$ and $\sim$ are symbols of order 1), then if we have $\mathsf{pt}_0 \overset{+}{\rightsquigarrow} \mathsf{pt}_1$, then pt_0 is a minor of pt_1 in the sense of graph theory — H is a minor of G if it can be obtained from G by deleting or contracting edges, and by deleting vertices. This interpretation matches the intuition that elaboration should fill the gap in the user-provided proof term.*

We prove that our procedure ensures that the elaboration $\overline{\mathsf{pt}}_1$ of a proof term $\overline{\mathsf{pt}}_0$ is such that $\overline{\mathsf{pt}}_0 \overset{+}{\rightsquigarrow} \overline{\mathsf{pt}}_1$. We first show that this holds for re-elaboration (Proposition 7) in isolation, which we then use to show that elaboration is predictable (Proposition 8).

Proposition 7 (Predictability of re-elaboration)**.**
If there exists a derivation of

$$\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_{I_0} \phi_0 \triangleleft P \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1$$

then $\overline{\mathsf{pt}}_0 \overset{+}{\rightsquigarrow} \overline{\mathsf{pt}}_1$.

Proof. The proof is by induction on the size of the derivation of

$$\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_{I_0} \phi_0 \triangleleft P \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1.$$

We do a case-disjunction on the last re-elaboration rule applied. There are two cases.

- *Case 1:* If this rule does not have a re-elaboration premise, we prove directly that $\overline{\mathsf{pt}}_0 \overset{+}{\rightsquigarrow} \overline{\mathsf{pt}}_1$ with one of the rule in Fig. 11.
- *Case 2:* If this rule has one re-elaboration premise

$$\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P' \twoheadrightarrow \Delta_3 \vdash \overline{\mathsf{pt}}_3 :_{I_3} \phi_3,$$

we prove the wanted result it in multiple steps using the fact that the $\overset{+}{\rightsquigarrow}$ relation is transitive:
  - $\overline{\mathsf{pt}}_0 \overset{+}{\rightsquigarrow} \overline{\mathsf{pt}}_2$ through one of the rule in Fig. 11;
  - $\overline{\mathsf{pt}}_2 \overset{+}{\rightsquigarrow} \overline{\mathsf{pt}}_3$ by the induction hypothesis;
  - $\overline{\mathsf{pt}}_3 \overset{+}{\rightsquigarrow} \overline{\mathsf{pt}}_1$ through one of the rule in Fig. 11.

We give below, for each re-elaboration rule of Fig. 10, the rule (*case 1*) or rules (*case 2*) of $\overset{+}{\rightsquigarrow}$ we use to conclude the proof:

- RE.Conv: $\overset{+}{\rightsquigarrow}.\textsc{Id}$
- RE.Congr.$\dot\Rightarrow$: $\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde\subset.\mathsf{H}$ and $\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde\Pi$
- RE.Congr.$\tilde\forall$: $\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde\subset.\mathsf{T}$ and $\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde\lambda$
- RE.ByLoc: $\overset{+}{\rightsquigarrow}.\%$
- RE.$\dot\forall.\tilde\forall$: $\overset{+}{\rightsquigarrow}.[\,\cdot\,].\dot\forall$ and $\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde\lambda$
- RE.Weak: $\overset{+}{\rightsquigarrow}.\textsc{Weak}$
- RE.[·].Weak: $\overset{+}{\rightsquigarrow}.[\,\cdot\,].\textsc{Weak}$
- RE.[·].$\dot\Rightarrow$: $\overset{+}{\rightsquigarrow}.\mathrm{I}.\dot\subset.\mathsf{H}$
- RE.L.$\dot\Rightarrow$: $\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde\subset.\mathsf{H}$
- RE.[·].$\tilde\forall$: $\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde\subset.\mathsf{T}$
- RE.L.$\tilde\forall$: $\overset{+}{\rightsquigarrow}.\mathrm{I}.\dot\subset.\mathsf{T}$ □

Proposition 8 (Predictability of elaboration)**.**
If there exists a derivation of

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \overline{\mathsf{pt}}_0 \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_I \phi_1$$

then $\overline{\mathsf{pt}}_0 \overset{+}{\rightsquigarrow} \overline{\mathsf{pt}}_1$.

Insert locality annotations.

$$\overset{+}{\rightsquigarrow}.\tilde{\lambda} : \lambda(x : \tau).\ \mathsf{pt} \overset{+}{\rightsquigarrow} \tilde{\lambda}(x : \tau).\ \mathsf{pt} \qquad \overset{+}{\rightsquigarrow}.\dot{\lambda} : \lambda(x : \tau).\ \mathsf{pt} \overset{+}{\rightsquigarrow} \dot{\lambda}(x : \tau).\ \mathsf{pt} \qquad \overset{+}{\rightsquigarrow}.\tilde{\sqsubset} : \mathsf{pt} \sqsubset \mathsf{pt}' \overset{+}{\rightsquigarrow} \mathsf{pt} \sqsubset \mathsf{pt}'$$

$$\overset{+}{\rightsquigarrow}.\dot{\sqsubset} : \mathsf{pt} \sqsubset \mathsf{pt}' \overset{+}{\rightsquigarrow} \mathsf{pt} \dot{\sqsubset} \mathsf{pt}' \qquad \overset{+}{\rightsquigarrow}.\tilde{\sqsubset} : \mathsf{pt} \sqsubset x \overset{+}{\rightsquigarrow} \mathsf{pt} \sqsubset x \qquad \overset{+}{\rightsquigarrow}.\dot{\sqsubset} : \mathsf{pt} \sqsubset x \overset{+}{\rightsquigarrow} \mathsf{pt} \dot{\sqsubset} x$$

Insert/remove locality modalities.

$$\overset{+}{\rightsquigarrow}.\%? : \%?\,\mathsf{pt} \overset{+}{\rightsquigarrow} \mathsf{pt} \qquad \overset{+}{\rightsquigarrow}.\% : \mathsf{pt} \overset{+}{\rightsquigarrow} \%\,\mathsf{pt}$$

Insert a proof-term application below a $[\,\cdot\,]$ modality

$$\overset{+}{\rightsquigarrow}.[\,\cdot\,].\dot{\forall} : \mathsf{pt} \overset{+}{\rightsquigarrow} [\%\,\mathsf{pt} \dot{\sqsubset} x]$$

Bound weakening.

$$\overset{+}{\rightsquigarrow}.\textsc{Weak} : \mathsf{pt} \overset{+}{\rightsquigarrow} (\mathsf{pt} \lessdot \varepsilon(H)) \qquad \overset{+}{\rightsquigarrow}.[\,\cdot\,].\textsc{Weak} : \mathsf{pt} \overset{+}{\rightsquigarrow} [(\%\,\mathsf{pt} \lessdot \varepsilon(H))]$$

Insert applications.

$$\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde{\sqsubset}.\mathrm{H} : \mathsf{pt} \overset{+}{\rightsquigarrow} \mathsf{pt} \dot{\sqsubset} \mathsf{id} \qquad \overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde{\sqsubset}.\mathrm{T} : \mathsf{pt} \overset{+}{\rightsquigarrow} \mathsf{pt} \dot{\sqsubset} x \qquad \overset{+}{\rightsquigarrow}.\mathrm{I}.\dot{\sqsubset}.\mathrm{T} : \mathsf{pt} \overset{+}{\rightsquigarrow} \mathsf{pt} \dot{\sqsubset} x$$

Insert generalizations.

$$\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde{\lambda} : \mathsf{pt} \overset{+}{\rightsquigarrow} \tilde{\lambda}(x : \tau).\ \mathsf{pt} \qquad \overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde{\Pi} : \mathsf{pt} \overset{+}{\rightsquigarrow} \tilde{\Pi}(\mathsf{id} : \tilde{\phi}).\ \mathsf{pt}$$

Name proof term holes.

$$\overset{+}{\rightsquigarrow}.\textsc{Weak.Name}._ : (\mathsf{pt} \lessdot \varepsilon(\_)) \overset{+}{\rightsquigarrow} (\mathsf{pt} \lessdot \varepsilon(H)) \qquad \overset{+}{\rightsquigarrow}.\textsc{App.Name}._ : \mathsf{pt} \circledcirc \_ \overset{+}{\rightsquigarrow} \mathsf{pt} \circledcirc H$$

Other.

$$\overset{+}{\rightsquigarrow}.\textsc{Id} : \mathsf{pt} \overset{+}{\rightsquigarrow} \mathsf{pt}$$

Figure 11: Rules for the over-approximation relation used to show that elaboration is predictable.

Proof. The proof is by induction on the size of the derivation, and is similar to the proof of Proposition 7, with some additional complexity.

First, many rules have a directed elaboration premise. This is easily to deal with with since directed elaboration is elaboration followed by re-elaboration: we use the induction hypothesis for the former and Proposition 7 for the latter. If directed elaboration targets a list of patterns instead of a single pattern, we apply the induction hypothesis on the selected pattern.

Second, some rules have more that one elaboration premises. When that happens, distinct premises are on independent part of the original proof term. Thus, we use the induction hypothesis for each part separately, and then use the fact that $\overset{+}{\rightsquigarrow}$ is a rewriting relation, and is thus compositional.

Third, some rule change the head constructor of the proof term to add locality information. We deal with those rules using using the rules inserting locating annotations (e.g., $\overset{+}{\rightsquigarrow}.\tilde{\lambda}$), inserting or removing locality modalities, etc.

Finally, the biggest difficulty comes form the universal quantification rules (E.L.λ, E.$\dot{\lambda}$, E.G.λ, and E.$\tilde{\lambda}$) where the proof term is modified to account the generalization that happens in the context. This require to use the rules $\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde{\sqsubset}.\mathrm{T}$, and $\overset{+}{\rightsquigarrow}.\mathrm{I}.\dot{\sqsubset}.\mathrm{T}$ to insert the necessary term applications to fill the gap between $\overline{\mathsf{pt}}$ and $\overline{\mathsf{pt}}_{\dot{\forall},\tilde{\forall}}$ (c.f. the universal quantification rules of Fig. 8).

For each rule elaboration rules (Fig. 8, Fig. 9), we give the $\overset{+}{\rightsquigarrow}$ rule from Fig. 11 we use (excluding cases that only need the induction hypothesis):

- E.L.Ax : $\overset{+}{\rightsquigarrow}.\textsc{Id}$
- E.G.Ax : $\overset{+}{\rightsquigarrow}.\textsc{Id}$
- E.%? : $\overset{+}{\rightsquigarrow}.\%?$
- E.Weak._ : $\overset{+}{\rightsquigarrow}.\textsc{Weak.Name}._$
- E.L.λ : $\overset{+}{\rightsquigarrow}.\dot{\lambda}$, $\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde{\sqsubset}.\mathrm{T}$ and $\overset{+}{\rightsquigarrow}.\mathrm{I}.\dot{\sqsubset}.\mathrm{T}$
- E.$\dot{\lambda}$: $\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde{\sqsubset}.\mathrm{T}$ and $\overset{+}{\rightsquigarrow}.\mathrm{I}.\dot{\sqsubset}.\mathrm{T}$
- E.G.λ : $\overset{+}{\rightsquigarrow}.\tilde{\lambda}$, $\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde{\sqsubset}.\mathrm{T}$ and $\overset{+}{\rightsquigarrow}.\mathrm{I}.\dot{\sqsubset}.\mathrm{T}$
- E.$\tilde{\lambda}$: $\overset{+}{\rightsquigarrow}.\mathrm{I}.\tilde{\sqsubset}.\mathrm{T}$ and $\overset{+}{\rightsquigarrow}.\mathrm{I}.\dot{\sqsubset}.\mathrm{T}$
- E.I.L.$\sqsubset$.Right : $\overset{+}{\rightsquigarrow}.\dot{\sqsubset}$
- E.I.L.$\sqsubset$.Left : $\overset{+}{\rightsquigarrow}.\dot{\sqsubset}$
- E.I.G.$\sqsubset$: $\overset{+}{\rightsquigarrow}.\dot{\sqsubset}$
- E.I.L.$\sqsubset$._ : $\overset{+}{\rightsquigarrow}.\dot{\sqsubset}$ and $\overset{+}{\rightsquigarrow}.\textsc{App.Name}._$
- E.I.G.$\sqsubset$._ : $\overset{+}{\rightsquigarrow}.\tilde{\sqsubset}$ and $\overset{+}{\rightsquigarrow}.\textsc{App.Name}._$
- E.I.$\dot{\sqsubset}$._ : $\overset{+}{\rightsquigarrow}.\textsc{App.Name}._$
- E.I.$\tilde{\sqsubset}$._ : $\overset{+}{\rightsquigarrow}.\textsc{App.Name}._$
- E.T.L.$\sqsubset$: $\overset{+}{\rightsquigarrow}.\dot{\sqsubset}$
- E.T.G.$\sqsubset$: $\overset{+}{\rightsquigarrow}.\tilde{\sqsubset}$ □

E. Completeness of the Elaboration Procedure

Elaboration should fill the gaps left by the user in a proof term. If a proof term has no gaps, then it should be left as-is, where a proof term without gaps is a well-typed proof term.

Proposition 9 (Relative completeness of elaboration for annotated proof term)**.**
Let $\overline{\mathsf{pt}}$ *be an annotated proof term such that*

$$\mathbb{E}; \Gamma; \Lambda \vdash \overline{\mathsf{pt}} : \phi$$

is derivable. Then, there exists I *such that we can derive*

$$\mathbb{E}; \Gamma; \Lambda \vdash \overline{\mathsf{pt}} \rightsquigarrow \mathbb{E}; \Gamma; \Lambda \vdash \overline{\mathsf{pt}} :_I \phi.$$

Proof notes. We exploit the one-to-one correspondence between the typing and elaboration rules. □

Therefore, we have a form of completeness, similar to the one found in [37]: if there exists a proof term that prove a formula (this is purely declarative since our typing system is not an algorithmic one), then there exists a proof term that elaborate to that formula – Proposition 9 actually ensures that this is the same proof term.

Corollary 1 (Completeness for annotated proof terms)**.**
The proof system for the elaboration of annotated proof term is complete, i.e., if $\overline{\mathsf{pt}}$ *is a proof term such that either*

$$\mathbb{E}; \Gamma; \Lambda \vdash \overline{\mathsf{pt}} : \phi \qquad \textit{or} \qquad \mathbb{E}; \Gamma \vdash \overline{\mathsf{pt}} : \phi$$

is derivable then there exists pt_s *and* I *such that we can derive*

$$\mathbb{E}; \Gamma; (\Lambda) \vdash \mathsf{pt}_s \rightsquigarrow \mathbb{E}; \Gamma; (\Lambda) \vdash \overline{\mathsf{pt}} :_I \phi.$$

Proof. This is a direct corollary of Proposition 9. □

Appendix C
Preliminaries for the Erasability Theorem

A. Erasure

Property 1 (Basic properties)**.**

- *The relation* $\xrightarrow{\spadesuit}$ *is terminating.*
- *The relation* $\xrightarrow{\spadesuit}$ *is confluent.*

Proof. Termination is immediate as each erasure reduce the number of weakening by one. As there are no critical pair, the system is locally confluent, and thus confluent by Newman's lemma. □

B. Syntactic Implication Properties

We define syntactic implication relation in Fig. 12.

1) Technical lemmas: We observe that the typing environment on both sides of $\Rrightarrow$ is identical.

Lemma 4 (Typing environments and $\Rrightarrow$)**.**
If $\Delta_0 \vdash \phi_0 \Rrightarrow \Delta_1 \vdash \phi_1$ *then* Δ_0 *and* Δ_1 *have the same typing environment:*

$$\Delta_0 = \mathbb{E}; \Gamma_0; \Lambda_0 \qquad \Delta_1 = \mathbb{E}; \Gamma_1; \Lambda_1$$

Proof. By induction over the derivation of $\Rrightarrow$ and case analysis over the last rule applied (recall that rules are in Fig. 12), looking at rules that change the typing environments. □

Proposition 10 ($\Rrightarrow$ is locality-preserving)**.** *if* $\Delta_0 \vdash \phi_0 \Rrightarrow \Delta_1 \vdash \phi_1$ *and* ϕ_0 *is local (resp. global) then* ϕ_1 *is local (resp. global).*

Moreover,

- *if* $\phi_1 = (\dot{\phi} \lessdot \varepsilon_1)$*, then* $\phi_0 = (\dot{\phi} \lessdot \varepsilon_0)$
- *if* $\phi_0 = (\dot{\phi} \lessdot \varepsilon_0)$*, then* $\phi_1 = (\dot{\phi} \lessdot \varepsilon_1)$
- *if* $\phi_1 = [\dot{\phi}]_{\varepsilon_1}$*, then* $\phi_0 = [\dot{\phi}]_{\varepsilon_0}$
- *if* $\phi_0 = [\dot{\phi}]_{\varepsilon_0}$*, then* $\phi_1 = [\dot{\phi}]_{\varepsilon_1}$

Proposition 11 (SI.Weak is left-compatible with $\Rrightarrow$)**.** *if we have that*

$$\Delta_0 \vdash (\dot{\phi}_0 \lessdot \varepsilon_0) \Rrightarrow \Delta_1 \vdash (\dot{\phi}_1 \lessdot \varepsilon_1)$$

with $\mathbb{E}_1 \vdash \varepsilon_0 \overset{?}{\not\leq} \varepsilon_1$ *then*

$$\Delta_0, (\mathsf{id} : [\varepsilon_0 \leq \varepsilon_1]_0) \vdash (\dot{\phi}_0 \lessdot \varepsilon_1) \Rrightarrow \Delta_1 \vdash (\dot{\phi}_1 \lessdot \varepsilon_1)$$

2) Simulation:

Proposition 12 (Simulation of re-elaboration w.r.t. equivalence)**.**
If we have that

$$\mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \overline{\mathsf{pt}}_2 \Rrightarrow \mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \overline{\mathsf{pt}}_0$$

and that

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \overline{\mathsf{pt}}_0 :_{I_0} \phi_0 \triangleleft P \twoheadrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1$$

are derivable and that P *is without* $\tilde{\forall}$ *or* $\dot{\forall}$ *and well-defined in* $\mathbb{E}_2$ *then*

- $\mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \twoheadrightarrow \mathbb{E}_3; \Gamma_3; \Lambda_3 \vdash \overline{\mathsf{pt}}_3 :_{I_3} \phi_3$ *is also derivable*
- $\mathbb{E}_3; \Gamma_3; \Lambda_3 \vdash \overline{\mathsf{pt}}_3 \Rrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1$.

Proof. See Appendix D. □

3) Pattern generalization:

Definition 5 (Order on pattern)**.** Given two pattern P_0, P_1, we say that P_0 is greater (written down $\geq_{\mathbb{E}}$) than P_1 in context $\mathbb{E}$, when there exists $\sigma : \mathtt{Vars}_{\mathtt{Meta}}(P_0) \mapsto \mathtt{Terms}(\mathtt{Vars}(\mathbb{E}), \mathtt{Vars}_{\mathtt{Meta}}(P_1))$ such that $P_0\sigma = P_1$.

Lemma 5 (Directed elaboration and order on pattern)**.**
If $P_1 \geq P_2$ *and we can derive*

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash (\mathsf{pt} \triangleleft P_1) \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_I \phi_1$$

then we can derive

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash (\mathsf{pt} \triangleleft P_2) \rightsquigarrow \mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \overline{\mathsf{pt}}_2 :_I \phi_2$$

and $\mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \phi_1 \Rrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \phi_2$.

Syntactic Implication rules.

$$\textsc{SI.Weak}\ \frac{}{\Delta \vdash (\dot\phi \lessdot \varepsilon) \Rrightarrow \Delta, (\mathsf{id} : [\varepsilon \le \varepsilon_0]_0) \vdash (\dot\phi \lessdot \varepsilon_0)}$$

$$\textsc{SI.Refl}\ \frac{\begin{array}{c}\Delta_0 = \mathbb{E}; \Gamma, \Gamma_{\le,0}; \Lambda \\ \Delta_1 = \mathbb{E}; \Gamma, \Gamma_{\le,1}; \Lambda \\ \Gamma_{\le,0}, \Gamma_{\le,1} = ([\varepsilon_i \le \varepsilon_i']_0)_i\end{array}}{\Delta_0 \vdash \phi \Rrightarrow \Delta_1 \vdash \phi}$$

$$\textsc{SI.Trans}\ \frac{\Delta_0 \vdash \phi_0 \Rrightarrow \Delta_1 \vdash \phi_1 \qquad \Delta_1 \vdash \phi_1 \Rrightarrow \Delta_2 \vdash \phi_2}{\Delta_0 \vdash \phi_0 \Rrightarrow \Delta_2 \vdash \phi_2}$$

$$\textsc{SI.Context}\ \frac{\Delta_0 \vdash \phi_0 \Rrightarrow \Delta_1 \vdash \phi_1}{\Delta_0, \Delta \vdash \phi_0 \Rrightarrow \Delta_1, \Delta \vdash \phi_1}$$

$$\textsc{SI.Contract}\ \frac{}{\begin{array}{l}\Delta, (\mathsf{id} : [\varepsilon_0 \le \varepsilon_1]_0) \vdash \phi \\ \Rrightarrow \Delta, (\mathsf{id}_0 : [\varepsilon_0 \le \varepsilon]_0), (\mathsf{id}_1 : [\varepsilon \le \varepsilon_1]_0) \vdash \phi\end{array}}$$

$$\textsc{SI.Contract.+}\ \frac{}{\begin{array}{l}\Delta, (\mathsf{id} : [\varepsilon_0 + \varepsilon_1 \le \varepsilon_2 + \varepsilon_3]_0) \vdash \phi \\ \Rrightarrow \Delta, (\mathsf{id}_0 : [\varepsilon_0 \le \varepsilon_2]_0), (\mathsf{id}_1 : [\varepsilon_1 \le \varepsilon_3]_0) \vdash \phi\end{array}}$$

$$\textsc{SI.ByLoc}\ \frac{\Delta_0 \vdash (\dot\phi \lessdot \varepsilon_0) \Rrightarrow \Delta_1 \vdash (\dot\phi \lessdot \varepsilon_1)}{\Delta_0 \vdash [\dot\phi]_{\varepsilon_0} \Rrightarrow \Delta_1 \vdash [\dot\phi]_{\varepsilon_1}}$$

$$\textsc{SI.ByGlob}\ \frac{\Delta_0 \vdash [\dot\phi]_{\varepsilon_0} \Rrightarrow \Delta_1 \vdash [\dot\phi]_{\varepsilon_1}}{\Delta_0 \vdash (\dot\phi \lessdot \varepsilon_0) \Rrightarrow \Delta_1 \vdash (\dot\phi \lessdot \varepsilon_1)}$$

$$\textsc{SI.Destr.}\dot\Rightarrow\textsc{.-}\ \frac{\Delta_0 \vdash (\dot\phi_0 \dot\Rightarrow \dot\phi_1 \lessdot \varepsilon_0) \Rrightarrow \Delta_1 \vdash (\dot\phi_0 \dot\Rightarrow \dot\phi_1 \lessdot \varepsilon_1)}{\Delta_0, (\mathsf{id} : \dot\phi_0) \vdash (\dot\phi_1 \lessdot \varepsilon_0) \Rrightarrow \Delta_1, (\mathsf{id} : \dot\phi_0) \vdash (\dot\phi_1 \lessdot \varepsilon_1)}$$

$$\textsc{SI.Destr.}\tilde\Rightarrow\textsc{.-}\ \frac{\Delta_0 \vdash \tilde\phi \tilde\Rightarrow \tilde\phi_0 \Rrightarrow \Delta_1 \vdash \tilde\phi \tilde\Rightarrow \tilde\phi_1}{\Delta_0, (\mathsf{id} : \tilde\phi) \vdash \tilde\phi_0 \Rrightarrow \Delta_1, (\mathsf{id} : \tilde\phi) \vdash \tilde\phi_1}$$

$$\textsc{SI.Destr.}\dot\Rightarrow\ \frac{\begin{array}{c}\Delta_0 \vdash (\dot\phi_0 \lessdot \varepsilon_0) \Rrightarrow \Delta_1 \vdash (\dot\phi_0 \lessdot \varepsilon_1) \\ \Delta_0 \vdash (\dot\phi_0 \dot\Rightarrow \dot\phi_1 \lessdot \varepsilon_2) \Rrightarrow \Delta_1 \vdash (\dot\phi_0 \dot\Rightarrow \dot\phi_1 \lessdot \varepsilon_3)\end{array}}{\Delta_0 \vdash (\dot\phi_1 \lessdot \varepsilon_0 + \varepsilon_2) \Rrightarrow \Delta_1 \vdash (\dot\phi_1 \lessdot \varepsilon_1 + \varepsilon_3)}$$

$$\textsc{SI.Destr.}\tilde\Rightarrow\ \frac{\begin{array}{cc}\Delta_0 = \mathbb{E}_0; \Gamma_0; \Lambda_0 & \Delta_1 = \mathbb{E}_1; \Gamma_1; \Lambda_1 \\ \mathbb{E}_0; \Gamma_0 \vdash \tilde\phi_2 & \mathbb{E}_1; \Gamma_1 \vdash \tilde\phi_2 \\ \multicolumn{2}{c}{\Delta_0 \vdash \tilde\phi_2 \tilde\Rightarrow \tilde\phi_0 \Rrightarrow \Delta_1 \vdash \tilde\phi_2 \tilde\Rightarrow \tilde\phi_1}\end{array}}{\Delta_0 \vdash \tilde\phi_0 \Rrightarrow \Delta_1 \vdash \tilde\phi_1}$$

$$\textsc{SI.Intro.}\dot\Rightarrow\ \frac{\Delta_0, (\mathsf{id} : \dot\phi) \vdash (\dot\phi_0 \lessdot \varepsilon_0) \Rrightarrow \Delta_1, (\mathsf{id} : \dot\phi) \vdash (\dot\phi_0 \lessdot \varepsilon_1)}{\Delta_0 \vdash (\dot\phi \dot\Rightarrow \dot\phi_0 \lessdot \varepsilon_0) \Rrightarrow \Delta_1 \vdash (\dot\phi \dot\Rightarrow \dot\phi_0 \lessdot \varepsilon_1)}$$

$$\textsc{SI.Intro.}\tilde\Rightarrow\ \frac{\Delta_0, (\mathsf{id} : \tilde\phi) \vdash \tilde\phi_0 \Rrightarrow \Delta_1, (\mathsf{id} : \tilde\phi) \vdash \tilde\phi_1}{\Delta_0 \vdash \tilde\phi \tilde\Rightarrow \tilde\phi_0 \Rrightarrow \Delta_1 \vdash \tilde\phi \tilde\Rightarrow \tilde\phi_1}$$

$$\textsc{SI.Congr.Weak}\ \frac{\begin{array}{cc}\Delta_0 = \mathbb{E}_0; \Gamma_0; \Lambda_0 & \Delta_1 = \mathbb{E}_1; \Gamma_1; \Lambda_1 \\ \mathbb{E}_0; \Gamma_0 \vdash [\varepsilon_0 \le \varepsilon]_0 & \mathbb{E}_1; \Gamma_1 \vdash [\varepsilon_1 \le \varepsilon]_0 \\ \multicolumn{2}{c}{\Delta_0 \vdash (\dot\phi \lessdot \varepsilon_0) \Rrightarrow \Delta_1 \vdash (\dot\phi \lessdot \varepsilon_1)}\end{array}}{\Delta_0 \vdash (\dot\phi \lessdot \varepsilon) \Rrightarrow \Delta_1 \vdash (\dot\phi \lessdot \varepsilon)}$$

$$\textsc{SI.Congr.Weak1}\ \frac{\begin{array}{cc}\Delta_1 = \mathbb{E}_1; \Gamma_1; \Lambda_1 & \mathbb{E}_1; \Gamma_1 \vdash [\varepsilon_0 \le \varepsilon]_0 \\ \multicolumn{2}{c}{\Delta_0 \vdash (\dot\phi \lessdot \varepsilon_0) \Rrightarrow \Delta_1 \vdash (\dot\phi \lessdot \varepsilon_0)}\end{array}}{\Delta_0 \vdash (\dot\phi \lessdot \varepsilon_0) \Rrightarrow \Delta_1 \vdash (\dot\phi \lessdot \varepsilon)}$$

$$\textsc{SI.Congr.Weak.-}\ \frac{\Delta_0 \vdash (\dot\phi_0 \lessdot \varepsilon_0) \Rrightarrow \Delta_1 \vdash (\dot\phi_1 \lessdot \varepsilon_1)}{\begin{array}{l}\Delta_0, (\mathsf{id} : [\varepsilon_0 \le \varepsilon]_0) \vdash (\dot\phi \lessdot \varepsilon) \\ \Rrightarrow \Delta_1, (\mathsf{id} : [\varepsilon_1 \le \varepsilon]_0) \vdash (\dot\phi \lessdot \varepsilon)\end{array}}$$

Figure 12: Rules for the syntactic implication relations.

C. Footprint

Proposition 13 (Footprint preservation)**.**
Let $\mathtt{LID}(\mathsf{pt})$ *be the set of free* local *identifiers in the proof-term* pt*. Then, all elaboration sub-procedures preserve* $\mathtt{LID}$*, i.e.,* $I = \mathtt{LID}(\mathsf{pt}) = \mathtt{LID}(\overline{\mathsf{pt}})$ *if* pt *and* $\overline{\mathsf{pt}}$ *are such that one of the following holds:*

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt} \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}} :_I \phi$$

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash (\mathsf{pt} \ \triangleleft\ P) \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}} :_I \phi$$

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt} :_I \phi_0 \triangleleft P \rightsquigarrow\!\!\!\!\Rightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_I \phi_1$$

Proof. This immediately follows from the rules. □

Proposition 14 (Footprint minimality)**.**
If we can derive

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt} \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}} :_{I_0} \phi$$
$$\mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash_{I_1} \overline{\mathsf{pt}} : \phi$$

then $I_0 \subseteq I_1$.

Proof. By Proposition 13, the free local identifiers in $\overline{\mathsf{pt}}$ must be at least that of pt, i.e., $\mathtt{LID}(\mathsf{pt}) = \mathtt{LID}(\overline{\mathsf{pt}}) = I_0$.

We conclude by observing that in a proof term typing derivation

$$\mathbb{E}; \Gamma; \Lambda \vdash_I \overline{\mathsf{pt}} : \phi,$$

the footprint I must contain at-least all free local identifiers of $\overline{\mathsf{pt}}$. □

Lemma 6 (Footprint and erasure)**.**
If $\mathsf{pt} \xrightarrow{\spadesuit} \mathsf{pt}_0$ *and we can derive one of the following:*

- $\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt} \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}} :_I \phi$
- $\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash (\mathsf{pt} \ \triangleleft\ P) \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}} :_I \phi$
- $\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt} :_I \phi_0 \triangleleft P \rightsquigarrow\!\!\!\!\Rightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}} :_I \phi_1$

then $I = \mathtt{LID}(\mathsf{pt}) = \mathtt{LID}(\mathsf{pt}_0)$.

Proof. Erasure does not change the *local* identifiers in a valid elaboration. □

Appendix D
Proof of the simulation lemma

In that section, we present the proof of Proposition 12. This proof is by induction on the derivation of

$$\Delta_2 \vdash \phi_2 \Rrightarrow \Delta_0 \vdash \phi_0$$

In order to do this proof, we do a case analysis on the both the head rule used to derive that

$$\Delta_2 \vdash \phi_2 \Rrightarrow \Delta_0 \vdash \phi_0$$

and the one use the derive

$$\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_{I_0} \phi_0 \triangleleft P \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1.$$

We list all cases in Fig. 13, where cases that cannot happens are marked by ✗. For example, for the case RE.[·].WEAK and SI.INTRO.⇛, the re-elaboration rule require that ϕ_0 has the form $[\,\cdot\,]$., while the syntactic implication rule require that ϕ_0 has the form $(\cdot \dot{\Rightarrow} \cdot \lessdot \cdot)$: therefore this case is impossible.

Case S-1

In that case, we consider the rule RE.CONV when ϕ_0 is a local formula. Then, ϕ_2 is also a local formula with a different bound (Proposition 10). Therefore, we have that:

1) $\phi_0 = (\dot{\phi}_0 \lessdot \varepsilon_0)$
2) $\Delta_1 = \Delta_0$, $\overline{\mathsf{pt}}_1 = \overline{\mathsf{pt}}_0$ and $\phi_1 = \phi_0$
3) $\phi_2 = (\dot{\phi}_2 \lessdot \varepsilon_2)$
4) $\mathbb{E}_0 \vdash \phi_0 \overset{?}{\leq} P$
5) $\Delta_0 \vdash \phi_0 \equiv_\alpha \Delta_2 \vdash \phi_2$

Since $\mathbb{E}_0 \vdash \phi_0 \overset{?}{\leq} P$, we have that $P = \_$ or $P = (\dot{\phi}_P \lessdot \varepsilon_P)$ with $\varepsilon_P = \_$ or $\varepsilon_P \equiv_\alpha \varepsilon_0$ (since the bound position of pattern are either fully specify or a inference pattern).

Subcase S-1.1: $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\leq} P$

This cover the case $P = \_$ and the case $P = (\dot{\phi} P \lessdot \_)$.[1] Therefore, RE.CONV also apply to $\overline{\mathsf{pt}}_2$ and give

$$\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2.$$

And we have that

$$\Delta_2 \vdash \phi_2 \Rrightarrow \Delta_1 \vdash \phi_1.$$

Subcase S-1.2: $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\not\leq} P$

Therefore, we have that $P = (\dot{\phi}_P \lessdot \varepsilon_P)$ with $\varepsilon_P \equiv_\alpha \varepsilon_0$ Since we have that $\mathbb{E}_0 \vdash \phi_0 = (\dot{\phi} \lessdot \varepsilon_0) \overset{?}{\leq} P$, We have that $\mathbb{E}_0 = \mathbb{E}_2 \vdash \dot{\phi} \overset{?}{\leq} P$ ($\mathbb{E}_0 = \mathbb{E}_2$ by point 3). We also have that $\mathbb{E}_2 \vdash \varepsilon_2 \overset{?}{\not\leq} \varepsilon_P$ (otherwise, we will have $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\leq} P$). Therefore, we can apply RE.WEAK and get that

$$\begin{aligned}&\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \\ &\rightsquigarrow \Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_P]_0) \vdash (\overline{\mathsf{pt}}_2 \lessdot \varepsilon_P(\mathsf{id})) :_{I_2} (\dot{\phi} \lessdot \varepsilon_P)^\cdot\end{aligned}$$

And we have that:

$$\begin{aligned}&\Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_0]_0) \vdash (\dot{\phi}_2 \lessdot \varepsilon_0) \\ &\Rrightarrow \Delta_0 \vdash \phi_0\end{aligned} \qquad \text{(by Proposition 11)}$$

and

$$\begin{aligned}&\Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_P]_0) \vdash (\dot{\phi}_2 \lessdot \varepsilon_P) \\ &\Rrightarrow \Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_0]_0) \vdash (\dot{\phi}_2 \lessdot \varepsilon_0)\end{aligned} \qquad \text{(by SI.REFL)}$$

[1] It also cover the particular case where $P = (\dot{\phi} P \lessdot \varepsilon_P)$ where $\varepsilon_P \equiv_\alpha \varepsilon_0 \equiv_\alpha \varepsilon_2$.

Therefore, by SI.TRANS, we have:

$$\Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_P]_0) \vdash (\dot{\phi}_2 \lessdot \varepsilon_P) \Rrightarrow \Delta_0 \vdash \phi_0.$$

Case S-2

In that case, we consider the rule RE.WEAK. (We don't need to do multiples cases with the various syntactic implication rules.) Therefore, by the fact that RE.WEAK apply and also the Proposition 10 we have that:

1) $\phi_0 = (\dot{\phi}_0 \lessdot \varepsilon_0)$
2) $P = (\dot{\phi}_P \lessdot \varepsilon_P)$
3) $\Delta_1 = \Delta_0, (\mathsf{id} : [\varepsilon_0 \leq \varepsilon_P]_0)$
4) $\phi_1 = (\dot{\phi}_0 \lessdot \varepsilon_1)$
5) $\phi_2 = (\dot{\phi}_2 \lessdot \varepsilon_2)$
6) $\mathbb{E}_0 \vdash \dot{\phi}_0 \overset{?}{\leq} \dot{\phi}_P$
7) $\mathbb{E}_0 \vdash \varepsilon_0 \overset{?}{\not\leq} \varepsilon_P$

Subcase S-2.1: $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\leq} P$

Therefore, RE.CONV also apply to $\overline{\mathsf{pt}}_2$ and give

$$\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2.$$

And we have that:

$$\Delta_2 \vdash \phi_2 \Rrightarrow \Delta_0 \vdash \phi_0 \qquad \text{(by hypothesis)}$$

and

$$\Delta_0 \vdash \phi_0 \Rrightarrow \Delta_1 \vdash \phi_1 \qquad \text{(by SI.WEAK)}$$

Therefore, by SI.TRANS, we have:

$$\Delta_2 \vdash \phi_2 \Rrightarrow \Delta_1 \vdash \phi_1.$$

Subcase S-2.2: $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\not\leq} P$

Therefore, we have that $P = (\dot{\phi}_P \lessdot \varepsilon_P)$ with $\varepsilon_P \equiv_\alpha \varepsilon_0$ Since we have that $\mathbb{E}_0 \vdash \phi_0 = (\dot{\phi} \lessdot \varepsilon_0) \overset{?}{\leq} P$, We have that $\mathbb{E}_2 \vdash \dot{\phi} \overset{?}{\leq} P$ by Lemma 4 We also have that $\mathbb{E}_2 \vdash \varepsilon_2 \overset{?}{\not\leq} \varepsilon_P$ (otherwise, we will have $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\leq} P$). Therefore, we can apply RE.WEAK and get that

$$\begin{aligned}&\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \\ &\rightsquigarrow \Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_P]_0) \vdash (\overline{\mathsf{pt}}_2 \lessdot \varepsilon_P(\mathsf{id})) :_{I_2} (\dot{\phi}_2 \lessdot \varepsilon_P)^\cdot\end{aligned}$$

and by SI.CONGR.WEAK._ and the hypothesis $\Delta_2 \vdash \phi_2 \Rrightarrow \Delta_0 \vdash \phi_0$, we have that

$$\Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_P]_0) \vdash (\dot{\phi}_2 \lessdot \varepsilon_P) \Rrightarrow \Delta_1 \vdash \phi_1.$$

Case S-3

In that case, we consider the rule RE.[·].WEAK. (We don't need to do multiples cases with the various syntactic implication rules.) Therefore, by the fact that RE.[·].WEAK apply and also the Proposition 10 we have that:

1) $\phi_0 = [\dot{\phi}_0]_{\varepsilon_0}$
2) $P = [\dot{\phi}_P]_{\varepsilon_P}$
3) $\Delta_1 = \Delta_0, (\mathsf{id} : [\varepsilon_0 \leq \varepsilon_P]_0)$
4) $\phi_1 = [\dot{\phi}_0]_{\varepsilon_1}$
5) $\phi_2 = [\dot{\phi}_2]_{\varepsilon_2}$

	RE.CONV	RE.WEAK	RE.[·].WEAK	RE.[·].⇛	RE.L.⇛	RE.[·].∀̇	RE.L.∀̇	RE.∀̇.∀̇	RE.BYLOC	RE.CONGR.⇛	RE.CONGR.∀̇
SI.WEAK	S-1	S-2	✗	✗	✗	✗	✗	✗	✗	✗	✗
SI.REFL	S-4	S-2, S-4	S-3, S-4	S-4	S-4	S-4	S-4	✗	S-4	S-4	✗
SI.TRANS	S-5	S-2, S-5	S-3, S-5	S-5	S-5	S-5	S-5	✗	S-5	S-5	✗
SI.CONTEXT	S-6	S-2, S-6	S-3, S-6	S-6	S-6	S-6	S-6	✗	S-6	S-6	✗
SI.CONTRACT	S-7	S-2, S-7	S-3, S-7	S-7	S-7	S-7	S-7	✗	S-7	S-7	✗
SI.CONTRACT.+	S-8	S-2, S-8	S-3, S-8	S-8	S-8	S-8	S-8	✗	S-8	S-8	✗
SI.BYLOC	S-9	✗	S-3	✗	✗	✗	✗	✗	S-11	✗	✗
SI.BYGLOB	S-1	S-2	✗	✗	✗	✗	✗	✗	✗	✗	✗
SI.DESTR.⇛._	S-1	S-2	✗	✗	✗	✗	✗	✗	✗	✗	✗
SI.DESTR.⇛._	S-12	✗	S-3, S-12	S-12	S-12	S-12	S-12	✗	S-12	S-12	✗
SI.DESTR.⇛	S-1	S-2	✗	✗	✗	✗	✗	✗	✗	✗	✗
SI.DESTR.⇛	S-13	✗	S-3, S-13	S-13	S-13	S-13	S-13	✗	S-13	S-13	✗
SI.INTRO.⇛	S-1	S-2	✗	✗	✗	✗	✗	✗	✗	✗	✗
SI.INTRO.⇛	S-14	✗	✗	S-15	S-16	✗	✗	✗	✗	S-17	✗
SI.CONGR.WEAK	S-1	S-2	✗	✗	✗	✗	✗	✗	✗	✗	✗
SI.CONGR.WEAK._	S-1	S-2	✗	✗	✗	✗	✗	✗	✗	✗	✗

Impossible cases are marked by ✗. Otherwise, we indicate the proof case where a particular pair of a syntactic implication and reelaboration rules are dealt with. In the PDF document, cases are hyperlinks.

Figure 13: Table of possible cases for the simulation proof (Appendix D).

6) $\mathbb{E}_0 \vdash \dot{\phi}_0 \overset{?}{\leq} \dot{\phi}_P$

7) $\mathbb{E}_0 \vdash \varepsilon_0 \overset{?}{\not\leq} \varepsilon_P$

Subcase S-3.1: $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\leq} P$

Therefore, RE.CONV also apply to $\overline{\mathsf{pt}}_2$ and give

$$\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2.$$

And we have that:

$$\Delta_2 \vdash \phi_2 \Rrightarrow \Delta_0 \vdash \phi_0 \qquad \text{(by hypothesis)}$$

and

$$\Delta_0 \vdash \phi_0 \Rrightarrow \Delta_1 \vdash \phi_1 \qquad \text{(by SI.WEAK)}$$

Therefore, by SI.TRANS, we have:

$$\Delta_2 \vdash \phi_2 \Rrightarrow \Delta_1 \vdash \phi_1.$$

Subcase S-3.2: $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\not\leq} P$

Therefore, we have that $P = [\dot{\phi}_P]_{\varepsilon_P}$ with $\varepsilon_P \equiv_\alpha \varepsilon_0$ Since we have that $\mathbb{E}_0 \vdash \phi_0 = (\dot{\phi} \lessdot \varepsilon_0) \overset{?}{\leq} P$, We have that $\mathbb{E}_2 \vdash \dot{\phi} \overset{?}{\leq} P$ by Lemma 4 We also have that $\mathbb{E}_2 \vdash \varepsilon_2 \overset{?}{\not\leq} \varepsilon_P$ (otherwise, we will have $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\leq} P$). Therefore, we can apply RE.WEAK and get that

$$\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \rightsquigarrow \Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_P]_0) \vdash (\overline{\mathsf{pt}}_2 \lessdot \varepsilon_P(\mathsf{id})) :_{I_2} [\dot{\phi}_2]_{\varepsilon_P}.$$

and by SI.CONGR.WEAK._ and the hypothesis $\Delta_2 \vdash \phi_2 \Rrightarrow \Delta_0 \vdash \phi_0$, we have that

$$\Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_P]_0) \vdash [\dot{\phi}_2]_{\varepsilon_P} \Rrightarrow \Delta_1 \vdash \phi_1.$$

Case S-4

In case, we can deal with syntactic implication derived using the head rule SI.REFL (Therefore, we have that $\phi_2 = \phi_0$). By the opening remark of B-C0a (re-elaboration does depends only on the formula and the pattern), we have the following judgement is derivable:

$$\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \rightsquigarrow \Delta_3 \vdash \overline{\mathsf{pt}}_3 :_{I_3} \phi_3.$$

We also have that $\Delta_1 = \Delta_0, \Delta$, also $\Delta_3 = \Delta_2, \Delta$ and $\phi_3 = \phi_1$. Therefore, we can apply SI.REFL and get that $\Delta_3 \vdash \phi_3 \Rrightarrow \Delta_1 \vdash \phi_1$.

Case S-5

In case, we can deal with syntactic implication derived using the head rule SI.TRANS. Since SI.TRANS apply, there exists $\Delta, \overline{\mathsf{pt}}, \phi$ such that:

- $\Delta_2 \vdash \phi_2 \Rrightarrow \Delta \vdash \phi$;
- $\Delta \vdash \phi \Rrightarrow \Delta_0 \vdash \phi_0$.

Therefore, by the induction hypothesis on $\Delta \vdash \phi \Rrightarrow \Delta_0 \vdash \phi_0$, we have that there exists $\Delta^1, \overline{\mathsf{pt}}^1, \phi^1, I^1, I$ such that:

- $\Delta \vdash \overline{\mathsf{pt}} :_I \phi \triangleleft P \rightsquigarrow \Delta^1 \vdash \overline{\mathsf{pt}}^1 :_{I^1} \phi^1$ is derivable;
- $\Delta^1 \vdash \phi^1 \Rrightarrow \Delta_1 \vdash \phi_1$ is derivable.

With this additional derivability of the re-elaboration of $\overline{\mathsf{pt}}$, we can apply the induction hypothesis on $\Delta_2 \vdash \phi_2 \Rrightarrow \Delta \vdash \phi$, to get that:

- $\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \rightsquigarrow \Delta_3 \vdash \overline{\mathsf{pt}}_3 :_{I_3} \phi_3$ is derivable;
- $\Delta_3 \vdash \phi_3 \Rrightarrow \Delta^1 \vdash \phi^1$ is derivable.

And we get $\Delta_3 \vdash \phi_3 \Rrightarrow \Delta_1 \vdash \phi_1$ by SI.TRANS.

Case S-6

In case, we can deal with syntactic implication derived using the head rule SI.CONTEXT. Therefore, we have that $\Delta^2 \vdash \phi_2 \Rrightarrow \Delta^0 \vdash \phi_0$ with $\Delta_2 = \Delta^2, \Delta$ and $\Delta_0 = \Delta^0, \Delta$

By the opening remark of B-C0a (re-elaboration does depends only on the formula and the pattern) and that there exists $\overline{\mathsf{pt}}^0$ is well-formed in Δ^0, we have the following judgement is derivable:

$$\Delta^0 \vdash \overline{\mathsf{pt}}^0 :_{I_0} \phi_0 \triangleleft P \rightsquigarrow \Delta^1 \vdash \overline{\mathsf{pt}}^1 :_{I_1} \phi_1$$

with $\Delta_1 = \Delta^1, \Delta$

Therefore by the induction hypothesis, we have that

- $\Delta^2 \vdash \overline{\mathsf{pt}}^2 :_{I_2} \phi_2 \triangleleft P \rightsquigarrow \Delta^3 \vdash \overline{\mathsf{pt}}^3 :_{I_3} \phi_3$
- $\Delta^3 \vdash \phi_3 \Rrightarrow \Delta^1 \vdash \phi_1$

By the same remark as before we have that $\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_3} \phi_2 \triangleleft P \rightsquigarrow \Delta_3 \vdash \overline{\mathsf{pt}}_3 :_{I_3} \phi_3$ with $\Delta_3 = \Delta^3, \Delta$

Finally, by SI.CONTEXT, $\Delta_3 \vdash \phi_3 \Rrightarrow \Delta_2 \vdash \phi_1$.

Case S-7

In case, we can deal with syntactic implication derived using the head rule SI.CONTRACT. Since SI.CONTRACT apply, there exists $\varepsilon^0, \varepsilon^1, \varepsilon^2, \Delta$, such that:

- $\Delta_0 = \Delta, (\mathsf{id}_0 : [\varepsilon^0 \leq \varepsilon^1]_0), (\mathsf{id}_1 : [\varepsilon^1 \leq \varepsilon^2]_0)$
- $\Delta_2 = \Delta, (\mathsf{id} : [\varepsilon^0 \leq \varepsilon^2]_0)$
- $\phi_0 = \phi_2$

By the opening remark of B-C0a (re-elaboration does not depends on the proof-term nor the context), we have that there exists Δ':

$$\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \rightsquigarrow \Delta_3 \vdash \overline{\mathsf{pt}}_3 :_{I_3} \phi_3$$
$$\Delta_3 = \Delta_2, \Delta'; \Delta_1 = \Delta_0; \Delta'$$
$$\text{and}$$
$$\phi_3 = \phi_1$$

Finally, by SI.CONTRACT, we get that $\Delta_3 \vdash \phi_3 \Rrightarrow \Delta_1 \vdash \phi_1$.

Case S-8

In case, we can deal with syntactic implication derived using the head rule SI.CONTRACT.+. Since SI.CONTRACT.+ apply, there exists $\varepsilon^0, \varepsilon^1, \varepsilon^2, \varepsilon^3, \Delta$, such that:

- $\Delta_0 = \Delta, (\mathsf{id}_0 : [\varepsilon^0 \leq \varepsilon^1]_0), (\mathsf{id}_1 : [\varepsilon^2 \leq \varepsilon^3]_0)$
- $\Delta_2 = \Delta, (\mathsf{id} : [\varepsilon^0 + \varepsilon^2 \leq \varepsilon^1 + \varepsilon^3]_0)$
- $\phi_0 = \phi_2$

By the opening remark of B-C0a (re-elaboration does not depends on the proof-term nor the context), we have that there exists Δ':

$$\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \rightsquigarrow \Delta_3 \vdash \overline{\mathsf{pt}}_3 :_{I_3} \phi_3$$
$$\Delta_3 = \Delta_2, \Delta'; \Delta_1 = \Delta_0; \Delta'$$
$$\text{and}$$
$$\phi_3 = \phi_1$$

Finally, by SI.CONTRACT.+, we get that $\Delta_3 \vdash \phi_3 \Rrightarrow \Delta_1 \vdash \phi_1$.

Case S-9

In case, we can deal with syntactic implication derived using the head rule SI.BYLOC and the re-elaboration derived using the head rule RE.CONV.

Since SI.BYLOC and RE.CONV apply, we have that:

- $\Delta_0 = \Delta_1, \overline{\mathsf{pt}}_0 = \overline{\mathsf{pt}}_1, \phi_0 = \phi_1$
- $\phi_0 = [\dot{\phi}_0]_{\varepsilon_0}$
- $\phi_2 = [\dot{\phi}_2]_{\varepsilon_2}$
- $\dot{\phi}_0 = \dot{\phi}_2$ (by 10)

There is two possible cases depending on if ϕ_2 match P.

Subcase S-9.1: $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\leq} P$

RE.CONV also apply to $\overline{\mathsf{pt}}_2$ and give

$$\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2.$$

And we have that

$$\Delta_2 \vdash \phi_2 \Rrightarrow \Delta_1 \vdash \phi_1.$$

Subcase S-9.2: $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\not\leq} P$

Therefore, we have that $P = [\dot{\phi}_P]_{\varepsilon_P}$ with $\varepsilon_P \equiv_\alpha \varepsilon_0$ Since we have that $\mathbb{E}_0 \vdash \phi_0 = [\dot{\phi}_2]_{\varepsilon_0} \overset{?}{\leq} P$, We have that $\mathbb{E}_0 = \mathbb{E}_2 \vdash \dot{\phi} \overset{?}{\leq} P$ by Lemma 4. We also have that $\mathbb{E}_2 \vdash \varepsilon_2 \overset{?}{\not\leq} \varepsilon_P$ (otherwise, we will have $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\leq} P$). Therefore, we can apply RE.[·].WEAK and get that

$$\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \rightsquigarrow \Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_P]_0) \vdash (\overline{\mathsf{pt}}_2 \lessdot \varepsilon_P(\mathsf{id})) :_{I_2} [\dot{\phi}]_{\varepsilon_P}.$$

And we have that:

$$\Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_0]_0) \vdash (\dot{\phi}_2 \lessdot \varepsilon_0) \Rrightarrow \Delta_0 \vdash (\dot{\phi}_0 \lessdot \varepsilon_0) \quad \text{(by Proposition 11)}$$

and

$$\Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_P]_0) \vdash [\dot{\phi}_2]_{\varepsilon_P} \Rrightarrow \Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_0]_0) \vdash [\dot{\phi}_2]_{\varepsilon_0} \quad \text{(by SI.REFL)}$$

Therefore, by SI.BYLOC and SI.TRANS, we have:

$$\Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_P]_0) \vdash [\dot{\phi}_2]_{\varepsilon_P} \Rrightarrow \Delta_0 \vdash \phi_0.$$

Case S-10

In case, we can deal with syntactic implication derived using the head rule SI.BYLOC and the re-elaboration derived using the head rule RE.[·].WEAK.

Since SI.BYLOC and RE.[·].WEAK apply, we have that:

- $\Delta_1 = \Delta_0, (H : [\varepsilon_0 \leq \varepsilon_P]_0)$
- $\phi_1 = [\dot{\phi}_0]_{\varepsilon_P}$
- $\phi_0 = [\dot{\phi}_0]_{\varepsilon_0}$
- $\phi_2 = [\dot{\phi}_2]_{\varepsilon_2}$
- $\dot{\phi}_0 = \dot{\phi}_2$ (by Proposition 10)
- $P = [\dot{\phi}_P]_{\varepsilon_P}$ with ε_P without holes

There is two possible cases depending on if ϕ_2 match P.

Subcase S-10.1: $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\leq} P$

RE.CONV apply to $\overline{\mathsf{pt}}_2$ and give

$$\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \rightsquigarrow \Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2.$$

And by SI.WEAK, SI.BYLOC and SI.TRANS, we have that

$$\Delta_2 \vdash \phi_2 \Rrightarrow \Delta_1 \vdash \phi_1.$$

Subcase S-10.2: $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\not\leq} P$

Since we have that $\mathbb{E}_0 \vdash \phi_0 = [\dot{\phi}_2]_{\varepsilon_0} \overset{?}{\leq} P$, We have that $\mathbb{E}_2 \vdash \dot{\phi}_2 \overset{?}{\leq} P$ by Lemma 4 We also have that $\mathbb{E}_2 \vdash \varepsilon_2 \overset{?}{\not\leq} \varepsilon_P$ (otherwise, we will have $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\leq} P$). Therefore, we can apply RE.[·].WEAK and get that

$$\Delta_2 \vdash \overline{\mathsf{pt}}_2 :_{I_2} \phi_2 \triangleleft P \rightsquigarrow \Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_P]_0) \vdash (\overline{\mathsf{pt}}_2 \lessdot \varepsilon_P(\mathsf{id})) :_{I_2} [\dot{\phi}]_{\varepsilon_P}.$$

And we have that:

$$\Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_0]_0) \vdash (\dot{\phi}_2 \lessdot \varepsilon_P) \Rrightarrow \Delta_1 \vdash (\dot{\phi}_0 \lessdot \varepsilon_P) \quad \text{by Proposition 11}$$

Therefore, by SI.ByLoc and SI.Trans, we have:

$$\Delta_2, (\mathsf{id} : [\varepsilon_2 \leq \varepsilon_P]_0) \vdash [\dot{\phi}_2]_{\varepsilon_P} \Rrightarrow \Delta_1 \vdash \phi_1.$$

Case S-11

In case, we can deal with syntactic implication derived using the head rule SI.ByLoc and the re-elaboration derived using the head rule RE.ByLoc.

Since SI.ByLoc and RE.ByLoc apply, we have that:

- $\phi_1 = (\dot{\phi}_1 \lessdot \varepsilon_1)$
- $\phi_0 = [\dot{\phi}_0]_{\varepsilon_0}$
- $\phi_2 = [\dot{\phi}_2]_{\varepsilon_2}$
- $\dot{\phi}_0 = \dot{\phi}_2$ (by Proposition 10)
- $P = (\dot{\phi}_P \lessdot \varepsilon_P)$
- $\Delta_0 \vdash \% \, \overline{\mathsf{pt}}_0 :_{I_0} \phi_0 \triangleleft P \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1$
- $\Delta_2 \vdash (\dot{\phi}_2 \lessdot \varepsilon_2) \Rrightarrow \Delta_0 \vdash (\dot{\phi}_0 \lessdot \varepsilon_0)$

Therefore by induction hypothesis, we have that:

- $\Delta_2 \vdash \% \, \overline{\mathsf{pt}}_2 :_{I_2} (\dot{\phi}_2 \lessdot \varepsilon_2) \triangleleft P \rightsquigarrow \Delta_3 \vdash \overline{\mathsf{pt}}_3 :_{I_3} \phi_3$
- $\Delta_3 \vdash \phi_3 \Rrightarrow \Delta_1 \vdash \phi_1$

and we can conclude by a simple application of RE.ByLoc on $\overline{\mathsf{pt}}_2$.

Case S-12

In case, we can deal with syntactic implication derived using the head rule SI.Destr.$\tilde{\Rightarrow}$._ Since this rule apply; we have that:

- $\Delta_0 = \Delta^0, (H : \tilde{\phi})$
- $\Delta_2 = \Delta^2, (H : \tilde{\phi})$
- $\Delta^2 \vdash \tilde{\phi} \tilde{\Rightarrow} \phi_0 \Rrightarrow \Delta^0 \vdash \tilde{\phi} \tilde{\Rightarrow} \phi_2$
- $\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_{I_0} \phi_0 \triangleleft P \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1$

Therefore, we also have that

$$\Delta^0 \vdash \overline{\mathsf{pt}}^0 :_{I_0} \tilde{\phi} \tilde{\Rightarrow} \phi_0 \triangleleft \tilde{\phi} \tilde{\Rightarrow} P \rightsquigarrow \Delta^1 \vdash \overline{\mathsf{pt}}^1 :_{I_1} \tilde{\phi} \tilde{\Rightarrow} \phi_1$$

with $\Delta_1 = \Delta^1, (H : \tilde{\phi})$ by either RE.Conv or RE.Congr.$\tilde{\Rightarrow}$.

By the induction hypothesis, we get that:

- $\Delta^2 \vdash \overline{\mathsf{pt}}^2 :_{I_2} \tilde{\phi} \tilde{\Rightarrow} \phi_2 \triangleleft \tilde{\phi} \tilde{\Rightarrow} P \rightsquigarrow \Delta^3 \vdash \overline{\mathsf{pt}}^3 :_{I_3} \tilde{\phi} \tilde{\Rightarrow} \phi_3$ with $\Delta_3 = \Delta^3, (H : \tilde{\phi})$ by either RE.Conv or RE.Congr.$\tilde{\Rightarrow}$
- $\Delta^3 \vdash \tilde{\phi} \tilde{\Rightarrow} \phi_3 \Rrightarrow \Delta^1 \vdash \tilde{\phi} \tilde{\Rightarrow} \phi_1$

Then, by SI.Destr.$\tilde{\Rightarrow}$._, we get: $\Delta_3 \vdash \phi_3 \Rrightarrow \Delta_1 \vdash \phi_1$ In order to get the derivation of the re-elaboration for $\overline{\mathsf{pt}}_2$, we need to do a case analysis on the rule used to get the re-elaboration of $\overline{\mathsf{pt}}^2$. (RE.Conv (subcase S-12.1) or RE.Congr.$\tilde{\Rightarrow}$ (subcase S-12.2)).

Subcase S-12.1

Since $\mathbb{E}_0 \vdash \tilde{\phi} \tilde{\Rightarrow} \phi_2 \overset{?}{\leq} \tilde{\phi} \tilde{\Rightarrow} P$, we also have that $\mathbb{E}_0 \vdash \phi_2 \overset{?}{\leq} P$. Therefore, we can apply RE.Conv to derive the re-elaboration of $\overline{\mathsf{pt}}_2$.

Subcase S-12.2

The derivation of the re-elaboration of $\overline{\mathsf{pt}}_2$ is a premise of RE.Congr.$\tilde{\Rightarrow}$.

Case S-13

In case, we can deal with syntactic implication derived using the head rule SI.Destr.$\tilde{\Rightarrow}$ Since this rule apply; we have that:

- $\Delta_2 \vdash \tilde{\phi} \tilde{\Rightarrow} \phi_0 \Rrightarrow \Delta_0 \vdash \tilde{\phi} \tilde{\Rightarrow} \phi_2$;
- $\Delta_0 \vdash \overline{\mathsf{pt}}_0 :_{I_0} \phi_0 \triangleleft P \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}_1 :_{I_1} \phi_1$.

Therefore, we also have that

$$\Delta_0 \vdash \overline{\mathsf{pt}}^0 :_{I_0} \tilde{\phi} \tilde{\Rightarrow} \phi_0 \triangleleft \tilde{\phi} \tilde{\Rightarrow} P \rightsquigarrow \Delta_1 \vdash \overline{\mathsf{pt}}^1 :_{I_1} \tilde{\phi} \tilde{\Rightarrow} \phi_1$$

by either RE.Conv or RE.Congr.$\tilde{\Rightarrow}$.

By the induction hypothesis, we get that:

- $\Delta_2 \vdash \overline{\mathsf{pt}}^2 :_{I_2} \tilde{\phi} \tilde{\Rightarrow} \phi_2 \triangleleft \tilde{\phi} \tilde{\Rightarrow} P \rightsquigarrow \Delta_3 \vdash \overline{\mathsf{pt}}^3 :_{I_3} \tilde{\phi} \tilde{\Rightarrow} \phi_3$; by either RE.Conv or RE.Congr.$\tilde{\Rightarrow}$;
- $\Delta_3 \vdash \tilde{\phi} \tilde{\Rightarrow} \phi_3 \Rrightarrow \Delta_1 \vdash \tilde{\phi} \tilde{\Rightarrow} \phi_1$.

Then, by SI.Destr.$\tilde{\Rightarrow}$, we get: $\Delta_3 \vdash \phi_3 \Rrightarrow \Delta_1 \vdash \phi_1$ In order to get the derivation of the re-elaboration for $\overline{\mathsf{pt}}_2$, we need to do a case analysis on the rule used to get the re-elaboration of $\overline{\mathsf{pt}}^2$. (RE.Conv (subcase S-13.1) or RE.Congr.$\tilde{\Rightarrow}$ (Subcase S-13.2)).

Subcase S-13.1

Since $\mathbb{E}_0 \vdash \tilde{\phi} \tilde{\Rightarrow} \phi_2 \overset{?}{\leq} \tilde{\phi} \tilde{\Rightarrow} P$, we also have that $\mathbb{E}_0 \vdash \phi_2 \overset{?}{\leq} P$. Therefore, we can apply RE.Conv to derive the re-elaboration of $\overline{\mathsf{pt}}_2$.

Subcase S-13.2

The derivation of the re-elaboration of $\overline{\mathsf{pt}}_2$ is a premise of RE.Congr.$\tilde{\Rightarrow}$.

Case S-14

In case, we can deal with syntactic implication derived using the head rule SI.Intro.$\tilde{\Rightarrow}$, when the re-elaboration of ϕ_0 is done by RE.Conv Therefore, we have that:

1) $\phi_0 = \tilde{\phi} \tilde{\Rightarrow} \tilde{\phi}_0$
2) $\Delta_1 = \Delta_0$, $\overline{\mathsf{pt}}_1 = \overline{\mathsf{pt}}_0$ and $\phi_1 = \phi_0$
3) $\phi_2 = \tilde{\phi} \tilde{\Rightarrow} \tilde{\phi}_2$
4) $\mathbb{E}_0 \vdash \phi_0 \overset{?}{\leq} P$
5) $\Delta_2, (H : \tilde{\phi}) \vdash \tilde{\phi}_2 \Rrightarrow \Delta_0, (H : \tilde{\phi}) \vdash \tilde{\phi}_0$

Since $\mathbb{E}_0 \vdash \phi_0 \overset{?}{\leq} P$, we have that $P = \_$ or $P = \tilde{\phi}_{P,0} \tilde{\Rightarrow} \tilde{\phi}_{P,1}$.

Subcase S-14.1: $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\leq} P$

Then, we can apply RE.Conv on ϕ_2, and get that $\Delta_3 = \Delta_2$, $\overline{\mathsf{pt}}_3 = \overline{\mathsf{pt}}_2$ and $\phi_3 = \phi_2$ So, we also have that $\Delta_3 \vdash \phi_3 \Rrightarrow \Delta_1 \vdash \phi_1$ by the hypothesis.

Subcase S-14.2: $\mathbb{E}_2 \vdash \phi_2 \overset{?}{\not\leq} P$

Then, we have that:

- $\mathbb{E}_2 \vdash \tilde{\phi} \overset{?}{\leq} \tilde{\phi}_{P,0}$
- $\mathbb{E}_2 \vdash \tilde{\phi}_2 \overset{?}{\not\leq} \tilde{\phi}_{P,1}$.

Given that we also have $\mathbb{E}_0 \vdash \tilde{\phi}_0 \overset{?}{\leq} \tilde{\phi}_{P,1}$, we also have that $\Delta_{\Delta_0,(H:\tilde{\phi})} \vdash \overline{\mathsf{pt}}^0 :_{I_0} \tilde{\phi}_0 \triangleleft \tilde{\phi}_{P,1} \rightsquigarrow \Delta_{\Delta_1,(H:\tilde{\phi})} \vdash \overline{\mathsf{pt}}^1 :_{I_1} \tilde{\phi}_0$. Therefore, we can apply the induction hypothesis to get that $\Delta_{\Delta_2,(H:\tilde{\phi})} \vdash \overline{\mathsf{pt}}^2 :_{I_2} \tilde{\phi}_2 \triangleleft \tilde{\phi}_{P,1} \rightsquigarrow \Delta_{\Delta_3,(H:\tilde{\phi})} \vdash \overline{\mathsf{pt}}^3 :_{I_3} \tilde{\phi}_3$

,with $\Delta_3, (H : \tilde{\phi}) \vdash \tilde{\phi}_3 \Rrightarrow \Delta_1, (H : \tilde{\phi}) \vdash \tilde{\phi}_0$ And we can conclude by apply SI.INTRO.$\tilde{\Rightarrow}$ with the previous syntactic implication judgement to get the wanted syntactic implication derivation. Similarly, we use RE.CONGR.$\tilde{\Rightarrow}$ with the previous re-elaboration judgement to get the wanted re-elaboration derivation.

Case S-15

In case, we can deal with syntactic implication derived using the head rule SI.INTRO.$\tilde{\Rightarrow}$, when the re-elaboration of ϕ_0 is done by RE.$[\cdot]$.$\tilde{\Rightarrow}$ Therefore, we have that:

1) $\phi_0 = \tilde{\phi} \tilde{\Rightarrow} \tilde{\phi}_0$
2) $\phi_2 = \dot{\phi} \tilde{\Rightarrow} \tilde{\phi}_2$
3) $P = [\dot{\phi}_P]_{\varepsilon_P}$
4) $\Delta_2, (H : \tilde{\phi}) \vdash \tilde{\phi}_2 \Rrightarrow \Delta_0, (H : \tilde{\phi}) \vdash \tilde{\phi}_0$
5) $\Delta_0, (H : \tilde{\phi}) \vdash \overline{\mathsf{pt}}^0 :_{I_0} \tilde{\phi}_0 \triangleleft P \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}^1 :_{I_1} \phi_1$

By the induction hypothesis (we can apply it thanks to the last two point of the previous enumeration), we get that:

- $\Delta_3 \vdash \phi_3 \Rrightarrow \Delta_1 \vdash \phi_1$
- $\Delta_2, (H : \tilde{\phi}) \vdash \overline{\mathsf{pt}}^2 :_{I_2} \tilde{\phi}_2 \triangleleft P \twoheadrightarrow \Delta_3 \vdash \overline{\mathsf{pt}}^3 :_{I_3} \phi_3$

And we can get the wanted elaborations with an application of the RE.$[\cdot]$.$\tilde{\Rightarrow}$ rule, with the premise being the previous re elaboration judgement.

Case S-16

In case, we can deal with syntactic implication derived using the head rule SI.INTRO.$\tilde{\Rightarrow}$, when the elaborations of ϕ_0 is done by RE.L.$\tilde{\Rightarrow}$ Therefore, we have that:

1) $\phi_0 = \tilde{\phi} \tilde{\Rightarrow} \tilde{\phi}_0$
2) $\phi_2 = \dot{\phi} \tilde{\Rightarrow} \tilde{\phi}_2$
3) $P = (\dot{\phi}_P \lessdot \varepsilon_P)$
4) $\Delta_2, (H : \tilde{\phi}) \vdash \tilde{\phi}_2 \Rrightarrow \Delta_0, (H : \tilde{\phi}) \vdash \tilde{\phi}_0$
5) $\Delta_0, (H : \tilde{\phi}) \vdash \overline{\mathsf{pt}}^0 :_{I_0} \tilde{\phi}_0 \triangleleft P \twoheadrightarrow \Delta_1 \vdash \overline{\mathsf{pt}}^1 :_{I_1} \phi_1$

By the induction hypothesis (we can apply it thanks to the last two point of the previous enumeration), we get that:

- $\Delta_3 \vdash \phi_3 \Rrightarrow \Delta_1 \vdash \phi_1$
- $\Delta_2, (H : \tilde{\phi}) \vdash \overline{\mathsf{pt}}^2 :_{I_2} \tilde{\phi}_2 \triangleleft P \twoheadrightarrow \Delta_3 \vdash \overline{\mathsf{pt}}^3 :_{I_3} \phi_3$

And we can get the wanted re elaboration with an application of the RE.L.$\tilde{\Rightarrow}$ rule, with the premise being the previous elaborations judgement.

Case S-17

In case, we can deal with syntactic implication derived using the head rule SI.INTRO.$\tilde{\Rightarrow}$, when the elaborations of ϕ_0 is done by RE.CONGR.$\tilde{\Rightarrow}$ Therefore, we have that:

1) $\phi_0 = \tilde{\phi} \tilde{\Rightarrow} \tilde{\phi}_0$
2) $\phi_2 = \dot{\phi} \tilde{\Rightarrow} \tilde{\phi}_2$
3) $P = \tilde{\phi}_{P,0} \tilde{\Rightarrow} \tilde{\phi}_{P,1}$
4) $\mathbb{E}_0 \vdash \tilde{\phi} \overset{?}{\leq} \tilde{\phi}_{P,0}$
5) $\Delta_2, (H : \tilde{\phi}) \vdash \tilde{\phi}_2 \Rrightarrow \Delta_0, (H : \tilde{\phi}) \vdash \tilde{\phi}_0$
6) $\Delta_0, (H : \tilde{\phi}) \vdash \overline{\mathsf{pt}}^0 :_{I_0} \tilde{\phi}_0 \triangleleft P \twoheadrightarrow \Delta_1, (H : \tilde{\phi}) \vdash \overline{\mathsf{pt}}^1 :_{I_1} \tilde{\phi}_1$

By the induction hypothesis (we can apply it thanks to the last two point of the previous enumeration), we get that:

- $\Delta_3, (H : \tilde{\phi}) \vdash \tilde{\phi}_3 \Rrightarrow \Delta_1, (H : \tilde{\phi}) \vdash \tilde{\phi}_1$
- $\Delta_2, (H : \tilde{\phi}) \vdash \overline{\mathsf{pt}}^2 :_{I_2} \tilde{\phi}_2 \triangleleft P \twoheadrightarrow \Delta_3, (H : \tilde{\phi}) \vdash \overline{\mathsf{pt}}^3 :_{I_3} \tilde{\phi}_3$

we also have that $\mathbb{E}_2 \vdash \tilde{\phi} \overset{?}{\leq} \tilde{\phi}_{P,0}$ (since $\mathbb{E}_2 \geq \mathbb{E}_0$) Therefore, we can get the wanted re elaboration with an application of the RE.CONGR.$\tilde{\Rightarrow}$ rule, with the premise being the previous elaborations judgement. We also have the wanted syntactic implication with an application of the SI.INTRO.$\tilde{\Rightarrow}$ rule, with the premise being the previous syntactic implication judgement.

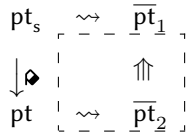
APPENDIX E
PROOF OF THE ERASABILITY THEOREM

We now recall and prove Theorem 1.

Theorem 1 (Erasability)**.** *If* pt_s *is a proof term such that* $\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt}_s \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_I \phi_1$ *is derivable and* pt_s *contains no term applications and* λ*s, then for any* pt *such that* $\mathsf{pt}_s \xrightarrow{\spadesuit} \mathsf{pt}$ *we have*

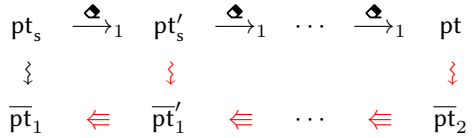
1) $\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt} \rightsquigarrow \mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \overline{\mathsf{pt}}_2 :_I \phi_2$ *is derivable;*
2) $\mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \phi_2 \Rrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \phi_1$.

This can be depicted as follows:



where the dotted box indicates the relations to be proved.

First, it is sufficient to prove that this hold for a single erasure step $\xrightarrow{\spadesuit}_1$, as can be seen on the following diagram:



where all relations on the top row and the left-most $\rightsquigarrow$ relation are by assumptions, and the other relations are proven (we also color proven relations in red).

It remains to show that Theorem 1 holds for a single erasure step $\xrightarrow{\spadesuit}_1$. We proceed by induction on the size of the smallest elaboration derivation of pt_s. To that end, we start by defining the two property we will prove by induction

Property $P_{\rightsquigarrow}(\underline{n}) :=$
for any $\mathbb{E}_0, \mathbb{E}_1, \Gamma_0, \Gamma_1, \Lambda_0, \Lambda_1, \mathsf{pt}_0, \overline{\mathsf{pt}}_1, \phi_1$ such that the smallest derivation of

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt}_0 \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_I \phi_1$$

is of depth at most $\underline{n}$, we have that for any pt_2 such that $\mathsf{pt}_0 \xrightarrow{\spadesuit}_1 \mathsf{pt}_2$, there exists $\mathbb{E}_2, \Gamma_2, \Lambda_2, \overline{\mathsf{pt}}_2, \phi_2$ such that

1) $\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt}_2 \rightsquigarrow \mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \overline{\mathsf{pt}}_2 :_I \phi_2$ is derivable;
2) $\mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \dot{\phi}_2 \Rrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \phi_1$.

Property $P_{\rightsquigarrow_P}(\underline{n}) :=$
for any $\mathbb{E}_0, \mathbb{E}_1, \Gamma_0, \Gamma_1, \Lambda_0, \Lambda_1, \mathsf{pt}_0, \overline{\mathsf{pt}}_1, \phi_1, P$ such that the smallest derivation of

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash (\mathsf{pt}_0 \ \triangleleft \ P) \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_I \phi_1$$

is of depth at most $\underline{n}$, we have that for any pt_2 such that $\mathsf{pt}_0 \xrightarrow{\spadesuit}_1 \mathsf{pt}_2$, there exists $\mathbb{E}_2, \Gamma_2, \Lambda_2, \overline{\mathsf{pt}}_2, \phi_2$ such that

1) $\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash (\mathsf{pt}_2 \triangleleft P) \rightsquigarrow \mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \overline{\mathsf{pt}}_2 :_I \phi_2$ is derivable;
2) $\mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \dot{\phi}_2 \Rrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \dot{\phi}_1$.

Induction: We prove $P_{\rightsquigarrow}(n)$ and $P_{\rightsquigarrow_P}(n)$ by induction on n.

Base cases : Any proof term that can be elaborated with at most 1 step are the identifiers proof terms, that can not be erased, that end the base case.

Induction case for $P_{\rightsquigarrow_P}(n)$: Take a derivation

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash (\mathsf{pt}_0 \triangleleft P) \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_I \phi_1$$

of minimal depth at most $\underline{n}$, and pt_2 such that $\mathsf{pt}_0 \xrightarrow{\spadesuit}_1 \mathsf{pt}_2$. Since the only rule that can be apply to derive $\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash (\mathsf{pt}_0 \triangleleft P) \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_I \phi_1$ is DE.RE, we have that the following partial proof tree exist (the footprint is the same, see Proposition 13).[2]

$$\dfrac{\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt}_0 \rightsquigarrow \mathbb{E}_{0.5}; \Gamma_{0.5}; \Lambda_{0.5} \vdash \overline{\mathsf{pt}}_{0.5} :_I \phi_{0.5} \quad \mathbb{E}_{0.5}; \Gamma_{0.5}; \Lambda_{0.5} \vdash \overline{\mathsf{pt}}_{0.5} :_I \phi_{0.5} \triangleleft P \Rrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_I \phi_1}{\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash (\mathsf{pt}_0 \triangleleft P) \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_I \phi_1}\ \text{DE.RE}$$

Therefore by $P_{\rightsquigarrow}(n-1)$, since

1) $\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt}_0 \rightsquigarrow \mathbb{E}_{0.5}; \Gamma_{0.5}; \Lambda_{0.5} \vdash \overline{\mathsf{pt}}_{0.5} :_I \phi_{0.5}$ is derivable with depth at most $n-1$;
2) and $\mathsf{pt}_0 \xrightarrow{\spadesuit}_1 \mathsf{pt}_1$;

we have that there exists $\mathbb{E}_{2.5}, \Gamma_{2.5}, \Lambda_{2.5}, \overline{\mathsf{pt}}_{2.5}$ such that

1) $\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt}_2 \rightsquigarrow \mathbb{E}_{2.5}; \Gamma_{2.5}; \Lambda_{2.5} \vdash \overline{\mathsf{pt}}_{2.5} :_I \phi_{2.5}$ is derivable;
2) $\mathbb{E}_{2.5}; \Gamma_{2.5}; \Lambda_{2.5} \vdash \phi_{2.5} \Rrightarrow \mathbb{E}_{0.5}; \Gamma_{0.5}; \Lambda_{0.5} \vdash \phi_{0.5}$.

And also by Proposition 12 since

1) $\mathbb{E}_{0.5}; \Gamma_{0.5}; \Lambda_{0.5} \vdash \overline{\mathsf{pt}}_{0.5} :_I \phi_{0.5} \triangleleft P \Rrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_I \phi_1$ is derivable;
2) and $\mathbb{E}_{2.5}; \Gamma_{2.5}; \Lambda_{2.5} \vdash \phi_{2.5} \Rrightarrow \mathbb{E}_{0.5}; \Gamma_{0.5}; \Lambda_{0.5} \vdash \phi_{0.5}$;

we have that there exists $\mathbb{E}_2, \Gamma_2, \Lambda_2, \overline{\mathsf{pt}}_2, \phi_2$ such that

1) $\mathbb{E}_{2.5}; \Gamma_{2.5}; \Lambda_{2.5} \vdash \overline{\mathsf{pt}}_{2.5} :_I \phi_{2.5} \triangleleft P \Rrightarrow \mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \overline{\mathsf{pt}}_2 :_I \phi_2$ is derivable;
2) $\mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \phi_2 \Rrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \phi_1$.

Item 2) above is one of the two property we must establish. To conclude, we observe that this derivation is valid:

$$\dfrac{\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt}_2 \rightsquigarrow \mathbb{E}_{2.5}; \Gamma_{2.5}; \Lambda_{2.5} \vdash \overline{\mathsf{pt}}_{2.5} :_I \phi_{2.5} \quad \mathbb{E}_{2.5}; \Gamma_{2.5}; \Lambda_{2.5} \vdash \overline{\mathsf{pt}}_{2.5} :_I \phi_{2.5} \triangleleft P \Rrightarrow \mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \overline{\mathsf{pt}}_2 :_I \phi_2}{\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash (\mathsf{pt}_2 \triangleleft P) \rightsquigarrow \mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \overline{\mathsf{pt}}_2 :_I \phi_2}\ \text{DE.RE}$$

Induction case for $P_{\rightsquigarrow}(n)$: Take a derivation

$$\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash \mathsf{pt}_0 \rightsquigarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \overline{\mathsf{pt}}_1 :_I \phi_1$$

[2]We introduce some new elements in the proof tree, those are also supposed to exists given the existence of the proof tree, this will be done at every case.

of minimal depth at most $\underline{n}$, and pt_2 such that $\mathsf{pt}_0 \xrightarrow{\spadesuit}_1 \mathsf{pt}_2$. We will perform a case analysis on the erasure, and if it is at the head of the term or not.

Most cases are of the same form, we show here a generic proof in order to give the general ideas for all those cases.

$$\dfrac{F((\mathsf{pt}_0^i)_{i\in[\![1;n]\!]}) \xrightarrow{\spadesuit}_1 F((\mathsf{pt}_2^i)_{i\in[\![1;n]\!]}),}{\exists!\ j, \mathsf{pt}_0^j \xrightarrow{\spadesuit}_1 \mathsf{pt}_2^j, \forall k \neq j, \mathsf{pt}_0^k = \mathsf{pt}_2^k}$$

Here the partial derivation proof usually look like this:

$$\dfrac{\begin{array}{l}\text{E.F}\\ \mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash (\mathsf{pt}_0^1 \triangleleft P^1)\\ \quad \rightsquigarrow \mathbb{E}_{0.1}; \Gamma_{0.1}; \Lambda_{0.1} \vdash \overline{\mathsf{pt}}_1^1 :_{I_1} \phi_1^1\\ \quad\quad \mathbb{E}_{0.(j-1)}; \Gamma_{0.(j-1)}; \Lambda_{0.(j-1)} \vdash (\mathsf{pt}_0^j \triangleleft P^j)\\ \quad\quad \rightsquigarrow \mathbb{E}_{0.j}; \Gamma_{0.j}; \Lambda_{0.j} \vdash \overline{\mathsf{pt}}_1^j :_{I_j} \phi_1^j\\ \quad\quad\quad \mathbb{E}_{0.(n-1)}; \Gamma_{0.(n-1)}; \Lambda_{0.(n-1)} \vdash (\mathsf{pt}_0^n \triangleleft P^n)\\ \quad\quad\quad \rightsquigarrow \mathbb{E}_{0.n}; \Gamma_{0.n}; \Lambda_{0.n} \vdash \overline{\mathsf{pt}}_1^n :_{I_n} \phi_1^n\end{array}}{\begin{array}{l}\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash F(\mathsf{pt}_0^i)\\ \quad \rightsquigarrow \mathbb{E}_{0.n(=1)}; \Gamma_{0.n(=1)}; \Lambda_{0.n(=1)} \vdash \hat{F}(\overline{\mathsf{pt}}_1^i) :_{\cup I_i} \phi_1\end{array}}$$

Here we can apply either $P_{\rightsquigarrow}(n)$ or $P_{\rightsquigarrow_P}(n)$ to pt_0^j and pt_2^j — we use one or the other depending on whether there is a pattern or not. Indeed

1) $\mathbb{E}_{0.(j-1)}; \Gamma_{0.(j-1)}; \Lambda_{0.(j-1)} \vdash (\mathsf{pt}_0^j \triangleleft P^j) \rightsquigarrow \mathbb{E}_{0.j}; \Gamma_{0.j}; \Lambda_{0.j} \vdash \overline{\mathsf{pt}}_1^j :_{I_j} \phi_1^j$ is derivable;
2) and $\mathsf{pt}_0^j \xrightarrow{\spadesuit}_1 \mathsf{pt}_2^j$.

So we have

1) $\mathbb{E}_{0.(j-1)}; \Gamma_{0.(j-1)}; \Lambda_{0.(j-1)} \vdash (\mathsf{pt}_2^j \triangleleft P^j) \rightsquigarrow \mathbb{E}_{2.j}; \Gamma_{2.j}; \Lambda_{2.j} \vdash \overline{\mathsf{pt}}_2^j :_{I_j} \phi_2^j$ is derivable;
2) and $\mathbb{E}_{2.j}; \Gamma_{2.j}; \Lambda_{2.j} \vdash \phi_2^j \Rrightarrow \mathbb{E}_{0.j}; \Gamma_{0.j}; \Lambda_{0.j} \vdash \phi_0^j$.

Therefore by Lemma 2 and Lemma 1, we also have for all $k \in \{j+1, \ldots, n\}$:

1) $\mathbb{E}_{2.(k-1)}; \Gamma_{2.(k-1)}; \Lambda_{2.(k-1)} \vdash (\mathsf{pt}_1^k \triangleleft P^k) \rightsquigarrow \mathbb{E}_{2.k}; \Gamma_{2.k}; \Lambda_{2.k} \vdash \overline{\mathsf{pt}}_1^k :_{I_k} \phi_1^k$ is derivable;
2) $\mathbb{E}_{2.k}, \Gamma_{2.k}, \Lambda_{2.k}$ is more general $\mathbb{E}_{0.k}, \Gamma_{0.k}, \Lambda_{0.k}$.

Given the previous two items, we have $\mathbb{E}_{2.k}; \Gamma_{2.k}; \Lambda_{2.k} \vdash \phi_0^k \Rrightarrow \mathbb{E}_{0.k}; \Gamma_{0.k}; \Lambda_{0.k} \vdash \phi_0^k$ by the rule SI.CONTEXT.

Finally, the following proof is valid:

$$\dfrac{\begin{array}{l}\text{E.F}\\ \mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash (\mathsf{pt}_2^1(=\mathsf{pt}_0^1) \triangleleft P^1)\\ \quad \rightsquigarrow \mathbb{E}_{2.1(=0.1)}; \Gamma_{2.1(=0.1)}; \Lambda_{2.1(=0.1)} \vdash \overline{\mathsf{pt}}_2^1(=\overline{\mathsf{pt}}_1^1) :_{I_1} \phi_2^1(=\phi_1^1)\\ \quad\quad \mathbb{E}_{2.(j-1)}; \Gamma_{2.(j-1)}; \Lambda_{2.(j-1)} \vdash (\mathsf{pt}_2^j \triangleleft P^j)\\ \quad\quad \rightsquigarrow \mathbb{E}_{2.j}; \Gamma_{2.j}; \Lambda_{2.j} \vdash \overline{\mathsf{pt}}_2^j :_{I_j} \phi_2^j\\ \quad\quad\quad \mathbb{E}_{2.(n-1)}; \Gamma_{2.(n-1)}; \Lambda_{2.(n-1)} \vdash (\mathsf{pt}_2^n(=\mathsf{pt}_0^n) \triangleleft P^n)\\ \quad\quad\quad \rightsquigarrow \mathbb{E}_{2.n}; \Gamma_{2.n}; \Lambda_{2.n} \vdash \overline{\mathsf{pt}}_2^n(=\overline{\mathsf{pt}}_1^n) :_{I_n} \phi_2^n(=\phi_2^n)\end{array}}{\begin{array}{l}\mathbb{E}_0; \Gamma_0; \Lambda_0 \vdash F(\mathsf{pt}_2^i)\\ \quad \rightsquigarrow \mathbb{E}_{2.n(=2)}; \Gamma_{2.n(=2)}; \Lambda_{2.n(=2)} \vdash \hat{F}(\overline{\mathsf{pt}}_2^i) :_{\cup I_i} \phi_2\end{array}}$$

We are left to prove that

$$\mathbb{E}_2; \Gamma_2; \Lambda_2 \vdash \phi_2 \Rrightarrow \mathbb{E}_1; \Gamma_1; \Lambda_1 \vdash \phi_1$$

which is done by applying the appropriate $\Rrightarrow$ rule, given that we have for any $k \in \{1, \ldots, n\}$, $\mathbb{E}_{2.k}; \Gamma_{2.k}; \Lambda_{2.k} \vdash \phi_2^k \Rrightarrow \mathbb{E}_{0.k}; \Gamma_{0.k}; \Lambda_{0.k} \vdash \phi_0^k$. This end the proof of this case.

$\mathsf{pt}_0 = (\mathsf{pt}_0^0 \lessdot \varepsilon(\_)) \xrightarrow{\spadesuit}_1 \mathsf{pt}_2 = \%?\,\mathsf{pt}_0^0$: Here, pt_0 is derivable (in the relevant context), and the only rule that be apply here is E.WEAK._, so the derivation look like this:

$$\dfrac{\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash (\mathsf{pt}_0^0 \lhd (\_ \lessdot \_)) \rightsquigarrow \mathbb{E}_{0.5};\Gamma_{0.5};\Lambda_{0.5} \vdash \overline{\mathsf{pt}}_1^0 :_I (\dot{\phi}_0{}^0 \lessdot \varepsilon_0^0)}{\begin{array}{l}\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash (\mathsf{pt}_0^0 \lessdot \varepsilon(\_)) \\ \quad \rightsquigarrow \mathbb{E}_{0.5};\Gamma_{0.5},(H : [\varepsilon_0^0 \le \varepsilon]_0);\Lambda_{0.5} \vdash (\overline{\mathsf{pt}}_1^0 \lessdot \varepsilon(H)) :_I (\dot{\phi}_0{}^0 \lessdot \varepsilon)\end{array}}\ \text{E.WEAK.\_}$$

Therefore, the following derivation is valid:

$$\dfrac{\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash (\mathsf{pt}_0^0 \lhd (\_ \lessdot \_)) \rightsquigarrow \mathbb{E}_{0.5};\Gamma_{0.5};\Lambda_{0.5} \vdash \overline{\mathsf{pt}}_1^0 :_I (\dot{\phi}_0{}^0 \lessdot \varepsilon_0^0)}{\begin{array}{l}\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash \%?\,\mathsf{pt}_0^0 \\ \quad \rightsquigarrow \mathbb{E}_{0.5};\Gamma_{0.5};\Lambda_{0.5} \vdash \overline{\mathsf{pt}}_1^0 :_I (\dot{\phi}_0{}^0 \lessdot \varepsilon_0^0)\end{array}}\ \text{E.\%?}$$

And we have to show that

$$\begin{array}{l}\mathbb{E}_{0.5};\Gamma_{0.5};\Lambda_{0.5} \vdash (\dot{\phi}_0{}^0 \lessdot \varepsilon_0^0) \\ \quad \Rrightarrow \mathbb{E}_{0.5};\Gamma_{0.5},(H : [\varepsilon_0^0 \le \varepsilon]_0);\Lambda_{0.5} \vdash (\dot{\phi}_0{}^0 \lessdot \varepsilon)\end{array}$$

which is exactly SI.WEAK.

$\mathsf{pt}_0 = (\mathsf{pt}_0^0 \lessdot \varepsilon(\mathsf{pt}_0^1)) \xrightarrow{\spadesuit}_1 \mathsf{pt}_2 = \%?\,\mathsf{pt}_0^0$: Here, pt_0 is derivable (in the relevant context), and the only rule that be apply here is E.WEAK, so the derivation look like this:

$$\dfrac{\begin{array}{l}\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash (\mathsf{pt}_0^1 \lhd [\_ \le \varepsilon]_0) \\ \quad \rightsquigarrow \mathbb{E}_{0.5};\Gamma_{0.5};\Lambda_{0.5} \vdash \overline{\mathsf{pt}}_1^1 :_\emptyset [\varepsilon_0^0 \le \varepsilon]_0 \\ \mathbb{E}_{0.5};\Gamma_{0.5};\Lambda_{0.5} \vdash (\mathsf{pt}_0^0 \lhd (\_ \lessdot \varepsilon_0^0)) \\ \quad \rightsquigarrow \mathbb{E}_1;\Gamma_1;\Lambda_1 \vdash \overline{\mathsf{pt}}_1^0 :_{I_0} (\dot{\phi}_0{}^0 \lessdot \varepsilon_0^0)\end{array}}{\begin{array}{l}\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash (\mathsf{pt}_0^0 \lessdot \varepsilon(\mathsf{pt}_0^1)) \\ \quad \rightsquigarrow \mathbb{E}_1;\Gamma_1;\Lambda_1 \vdash (\overline{\mathsf{pt}}_1^0 \lessdot \varepsilon(\overline{\mathsf{pt}}_1^1)) :_{I_0} (\dot{\phi}_0{}^0 \lessdot \varepsilon)\end{array}}\ \text{E.WEAK}$$

Since we have that $\mathbb{E}_0, \Gamma_0, \Lambda_0$ is more general than $\mathbb{E}_{0.5}, \Gamma_{0.5}, \Lambda_{0.5}$, then by Lemma 2, we have that

$$\begin{array}{l}\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash (\mathsf{pt}_0^0 \lhd (\_ \lessdot \varepsilon_0^0)) \\ \quad \rightsquigarrow \mathbb{E}_{1'};\Gamma_{1'};\Lambda_{1'} \vdash \overline{\mathsf{pt}}_1^0 :_{I_0} (\dot{\phi}_0{}^0 \lessdot \varepsilon_0^0)\end{array}$$

is derivable with $\mathbb{E}_{1'};\Gamma_{1'};\Lambda_{1'} \vdash (\dot{\phi}_0{}^0 \lessdot \varepsilon_0^0) \Rrightarrow \mathbb{E}_1;\Gamma_1;\Lambda_1 \vdash (\dot{\phi}_0{}^0 \lessdot \varepsilon_0^0)$. Then since $(\_ \lessdot \_)$ is more generic than $(\_ \lessdot \varepsilon_0^0)$, by Lemma 5, we also have that

$$\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash (\mathsf{pt}_0^0 \lhd (\_ \lessdot \_)) \rightsquigarrow \mathbb{E}_2;\Gamma_2;\Lambda_2 \vdash \overline{\mathsf{pt}}_2 :_\emptyset (\dot{\phi}_2 \lessdot \varepsilon_2)$$

is derivable with $\mathbb{E}_2;\Gamma_2;\Lambda_2 \vdash (\dot{\phi}_2 \lessdot \varepsilon_2) \Rrightarrow \mathbb{E}_{1'};\Gamma_{1'};\Lambda_{1'} \vdash (\dot{\phi}_0{}^0 \lessdot \varepsilon_0^0)$. Therefore, by transitivity of $\Rrightarrow$ (SI.TRANS), we have $\mathbb{E}_2;\Gamma_2;\Lambda_2 \vdash (\dot{\phi}_2 \lessdot \varepsilon_2) \Rrightarrow \mathbb{E}_1;\Gamma_1;\Lambda_1 \vdash (\dot{\phi}_0{}^0 \lessdot \varepsilon_0^0)$ And the following derivation is valid (the footprint is dealt with Proposition 13):

$$\dfrac{\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash (\mathsf{pt}_0^0 \lhd (\_ \lessdot \_)) \rightsquigarrow \mathbb{E}_2;\Gamma_2;\Lambda_2 \vdash \overline{\mathsf{pt}}_2 :_{I_0} (\dot{\phi}_2 \lessdot \varepsilon_2)}{\mathbb{E}_0;\Gamma_0;\Lambda_0 \vdash \%?\,\mathsf{pt}_0^0 \rightsquigarrow \mathbb{E}_2;\Gamma_2;\Lambda_2 \vdash \overline{\mathsf{pt}}_2 :_{I_0} (\dot{\phi}_2 \lessdot \varepsilon_2)}\ \text{E.\%?}$$

This concludes the proof.