Decidability of a Sound Set of Inference Rules for Computational Indistinguishability

Adrien Koutsos
LSV, CNRS, ENS Paris-Saclay, Université Paris-Saclay
Cachan, France
adrien.koutsos@lsv.fr

***Abstract*—Computational indistinguishability is a key property in cryptography and verification of security protocols. Current tools for proving it rely on cryptographic game transformations.**

We follow Bana and Comon's approach [1], [2], axiomatizing what an adversary cannot distinguish. We prove the decidability of a set of first-order axioms which are computationally sound, though incomplete, for protocols with a bounded number of sessions whose security is based on an IND-CCA$_2$ encryption scheme. Alternatively, our result can be viewed as the decidability of a family of cryptographic game transformations. Our proof relies on term rewriting and automated deduction techniques.

***Index Terms*—Security Protocols, Automated Deduction, Decision Procedure, Computational Indistinguishability**

I. Introduction

Designing security protocols is notoriously hard. For example, the TLS protocol used to secure most of the Internet connections was successfully attacked several times at the protocol level, e.g. the LogJam attack [3] or the TripleHandshake attack [4]. This shows that, even for high visibility protocols, and years after their design, attacks are still found.

Using formal methods to prove a security property is the best way to get a strong confidence. However, there is a difficulty, which is not present in standard program verification: we need not only to specify formally the program and the security property, but also the attacker. Several attacker models have been considered in the literature.

A popular attacker model, the *Dolev-Yao attacker*, grants the attacker the complete control of the network: he can intercept and re-route all messages. Besides, the adversary is allowed to modify messages using a fixed set of rules (e.g. given a cipher-text and its decryption key, he can retrieve the plain-text message). Formally, messages are terms in a term algebra and the rules are given through a set of rewrite rules. This model is very amenable to automatic verification of security properties. There are several automated tools, such as, e.g., ProVerif [5], Tamarin [6] and Deepsec [7].

Another attacker model, closer to a real world attacker, is the *computational attacker* model. This adversary also controls the network, but this model does not restrict the attacker to a fixed set of operations: the adversary can perform any probabilistic polynomial time computation. More formally, messages are bit-strings, random numbers are sampled uniformly among bit-strings in $\{0,1\}^\eta$ (where η is the *security parameter*) and the attacker is any probabilistic polynomial-time Turing machine (PPTM). This model offers stronger guarantees than the Dolev-Yao model (DY model), but formal proofs are harder to complete and more error-prone. There exist several formal verification tools in this model: for example, EasyCrypt [8] which relies on pRHL, and CryptoVerif [9] which performs game transformations. As expected, such tools are less automatic than the verification tools in the DY model. Moreover, the failure to find a proof in such tools, either because the proof search failed or did not terminate, or because the user could not manually find a proof, does not give any indication on the actual security of the protocol.

There is an alternative approach, the Bana-Comon model (BC model), introduced in [2]. In this model, we express the security of a protocol as the unsatisfiability of a set of formulas in first-order logic. The formulas contain the negation of the security property and *axioms*, which reflect implementation assumptions, such as functional correctness and cryptographic hypotheses on the security primitives. This method has several advantages over pRHL and game transformations. First, it is simpler, as there is no security game and no probabilities, only a first-order formula. Then, carrying out a proof of unsatisfiability in this logic entails the security of the protocol in the computational model. Finally, the absence of such a proof implies the existence of a model of the formula, i.e. an attack, albeit not necessarily a computational one; nonetheless, we know that the security of the protocol *cannot* be obtained without extra assumptions. Note that the Bana-Comon approach is only valid for protocols with a finite number of sessions (there is no unbounded replication). Since this is the model we use, we inherit this restriction.

There is another input to security proofs that we did not discuss yet: the class of security properties considered. Roughly, there are two categories. *Reachability* properties state that some bad state is unreachable. This includes, for example, authentication or (weak) secrecy. *Indistinguishability* properties state that an adversary cannot distinguish between the executions of two protocols. This allows for more complex properties, such as strong secrecy and unlinkability.

a) Deciding Security: When trying to prove a protocol, there are three possible outcomes: either we find a proof, which gives security guarantees corresponding to the attacker model; or we find an attack, meaning that the protocol is insecure; or the tool or the user (for interactive provers) could not carry out the proof and failed to find an attack. The latter case may happen for two different reasons. First, we could neither find a proof nor an attack because the proof method used is incomplete. In that case, we need either to make new assumptions and try again, or to use another proof technique. Second, the tool may not terminate on the protocol considered. This is problematic, as we do not know if we should continue waiting, and consume more resources and memory, or try another method.

This can be avoided for decidable classes of protocols and properties. Of course, such classes depend on both the attacker model and the security properties considered. We give here a non-exhaustive survey of such results. In the symbolic model, [10] shows decidability of secrecy (a reachability property) for a bounded number of sessions. In [11], the authors show the decidability of a secrecy property for *depth-bounded* protocols, with an unbounded number of sessions, using Well-Structured Transition Systems [12]. Chrétien et al [13] show the decidability of indistinguishability properties for a restricted class of protocols. E.g., they consider processes communicating on distinct channels and without else branches. The authors of [14] show the decidability of symbolic equivalence for a bounded number of sessions, but with conditional branching.

In the computational model, we are aware of only one direct result. In [15], the authors show the decidability of the security of a formula in the BC model, for *reachability properties*, for a bounded number of sessions. But there is an indirect way of getting decidability in the computational model, through a *computational soundness* theorem (e.g. [16]). A computational soundness theorem states that, for some given classes of protocols and properties, symbolic security implies computational security. These results usually make strong implementation assumptions (e.g. parsing assumptions, or the absence of dishonest keys), and require that the security primitives satisfy strong cryptographic hypothesis. By combining a decidability result in the symbolic model with a computational soundness theorem, which applies to the considered classes of protocols and properties (e.g. [17] for reachability properties, or [18] for indistinguishability properties), we obtain a decidability result in the computational model.

We discuss further related works later, in Section VIII.

b) Contributions: In this paper, we consider the BC model for indistinguishability properties [1]. This is a first-order logic in which we design a set of axioms Ax which includes, in particular, axioms for the IND-CCA$_2$ cryptographic assumption [19]. Given a protocol and a security property, we can build, using a folding technique described in [1], a ground atomic formula ψ expressing the security of the protocol. Showing the unsatisfiability of the conjunction of the axioms Ax and the negation of ψ entails the security of the protocol in the computational model, assuming that the encryption scheme is IND-CCA$_2$ secure.

Our main result is the decidability of the problem:

Input: A ground formula $\vec{u} \sim \vec{v}$.
Question: Is $\mathsf{Ax} \wedge \vec{u} \not\sim \vec{v}$ unsatisfiable?

That is, we show the decidability of a sound, though incomplete, axiomatization of computational indistinguishability.

All the formulas in Ax are Horn clauses, therefore to show the unsatisfiability of $\mathsf{Ax} \wedge \vec{u} \not\sim \vec{v}$ we use resolution with a negative strategy: we see axioms in Ax as inference rules and look for a derivation of the goal $\vec{u} \sim \vec{v}$. We prove the decidability of the corresponding satisfiability problem.

The main difficulty lies in dealing with equalities (defined through a term rewriting system R). First we show the completeness of an ordered strategy by commuting rule applications. This allows us to have only one rewriting modulo R at the beginning of the proof. We then bound the size of the terms after this rewriting as follows: we identify a class of proof cuts introducing arbitrary subterms; we give proof cut eliminations to remove them; and finally, we show that cut-free proofs are of bounded size w.r.t. the size of the conclusion.

c) Game Transformations: Our result can be reinterpreted as the decidability of the problem of determining whether there exits a sequence of game transformations [20], [21] that allows to prove the security of a protocol. Indeed, one can associate to every axiom in Ax either a cryptographic assumption or a game transformation.

Each unitary axiom in Ax (an atomic formula) corresponds to an instantiation of the IND-CCA$_2$ game. For instance, in the simpler case of IND-CPA security of an encryption $\{\_\}_{\mathsf{pk}}^{\mathsf{n}}$, no polynomial-time adversary can distinguish between two cipher-texts, even if it chooses the two corresponding plain-texts (here, n is the explicit encryption randomness). Initially, the public key pk is given to the adversary, who computes a pair of plain-texts $g(\mathsf{pk})$: g is interpreted as the adversary's computation. Then the two cipher-texts, corresponding to the encryptions of the first and second components of $g(\mathsf{pk})$, should be indistinguishable. This yields the unitary axiom:

$$\{\pi_1(g(\mathsf{pk}))\}_{\mathsf{pk}}^{\mathsf{n}} \sim \{\pi_2(g(\mathsf{pk}))\}_{\mathsf{pk}}^{\mathsf{n}}$$

Similarly, non-unitary axioms correspond to cryptographic game transformations. E.g., the function application axiom:

$$\vec{u} \sim \vec{v} \to f(\vec{u}) \sim f(\vec{v})$$

states that if no adversary can distinguish between the arguments of a function call, then no adversary can distinguish between the images. As for a cryptographic game transformation, the soundness of this axiom is shown by reduction. Given a winning adversary $\mathcal{A}$ against the conclusion $f(\vec{u}) \sim f(\vec{v})$, we build a winning adversary $\mathcal{B}$ against $\vec{u} \sim \vec{v}$: the adversary $\mathcal{B}$, on input $\vec{w}$ (which was sampled from $\vec{u}$ or $\vec{v}$), computes $f(\vec{w})$ and then gives the result to the distinguisher $\mathcal{A}$. The advantage of $\mathcal{B}$ against $\vec{u} \sim \vec{v}$ is then the advantage of $\mathcal{A}$ against $f(\vec{u}) \sim f(\vec{v})$, which is (by hypothesis) non negligible.

By interpreting every axiom in Ax as a cryptographic assumption or a game transformation, and the goal formula $\vec{u} \sim \vec{v}$ as the initial game, our result can be reformulated as showing the decidability of the following problem:

Input: An initial game $\vec{u} \sim \vec{v}$.
Question: Is there a sequence of game transformations in Ax showing that $\vec{u} \sim \vec{v}$ is secure?

From this point of view, our result guarantees a kind of sub-formula property for the intermediate games appearing in the game transformation proof. We may only consider intermediate games that are in a finite set computable from the original protocol: the other games are provably unnecessary detours. To our knowledge, our result is the first showing the decidability of a class of game transformations.

d) Scope and Limitations: To achieve decidability, we had to remove or restrict some axioms. The most important restriction is arguably that we do not include the transitivity axiom. The transitivity axiom states that to show that $\vec{u} \sim \vec{v}$, it is sufficient to find a $\vec{w}$ such that $\vec{u} \sim \vec{w}$ and $\vec{w} \sim \vec{v}$. Obviously, this axiom is problematic for decidability, as the vector of term $\vec{w}$ must be guessed, and may be arbitrarily large. Therefore, instead of directly including transitivity, we push it inside the CCA2 axiom schema, by allowing instances of the CCA2 axiom to deal simultaneously with multiple keys and interleaved encryptions. Of course, this is at the cost of a more complex axiom. We do not know if our problem remains decidable when we include the transitivity axiom.

e) Applications: The BC indistinguishability model has been used to analyse RFID protocols [22], a key-wrapping API [23] and an e-voting protocol [24]. Ideally, we would like future case studies to be carried out automatically and machine checked. Because our procedure has a high complexity, it is unclear whether it can be used directly for this. Still, our procedure could be a building block in a tool doing an incomplete but faster heuristic exploration of the proof space.

CryptoVerif and EasyCrypt are based on game transformations, directly in the former and through the pRHL logic in the latter. Therefore, our result could be used to bring automation to these tools. Of course, both tools allow for more rules. Still, we could identify which game transformations or rules correspond to our axioms, and apply our result to obtain decidability for this subset of game transformations.

f) Outline: We introduce the logic and the axioms in Section II and III. We then state the main result in Section IV, and depict the difficulties of the proof. Finally we sketch the proof: in Section V we show the rule commutations and some cut eliminations; in Section VI we show a normal form for proofs and its properties; and in Section VII we give more cut eliminations and the decision procedure. We discuss in details the related works in Section VIII. Most of the proofs are in appendix.

II. The Logic

We recall here the logic introduced in [1]. In this logic, terms represent messages of the protocol sent over the network, including the adversary's inputs, which are specified using additional function symbols. Formulas are built using the usual Boolean connectives and FO quantifiers, and a single predicate, $\sim$, which stands for indistinguishability. The semantics of the logic is the usual first-order semantics, though we are particularly interested in computational models, in which terms are interpreted as PPTMs, and $\sim$ is interpreted as computational indistinguishability.

This logic is then used as follows: given a protocol and a security property, we can build (automatically) a single formula $\vec{u} \sim \vec{v}$ expressing the security of the protocol. We specify, through a (recursive) set of axioms, what the adversary *cannot* do. This yields a set of axioms Ax. We show that $\mathsf{Ax} \wedge \vec{u} \not\sim \vec{v}$ is unsatisfiable, and that the axioms Ax are valid in the computational model. We deduce from this the security of the protocol in the computational model.

A. Syntax

a) Terms: Terms are built upon a set of function symbols $\mathcal{F}$, a countable set of names $\mathcal{N}$ and a countable set of variables $\mathcal{X}$. This is a sorted logic with two sorts $\mathcal{S}_\mathsf{m}, \mathcal{S}_\mathsf{b}$, with $\mathcal{S}_\mathsf{b} \subseteq \mathcal{S}_\mathsf{m}$.

The set $\mathcal{F}$ of function symbols is composed of a countable set of adversarial function symbols $\mathcal{G}$ (representing the adversary computations), and the following function symbols: the pair $\langle \_, \_ \rangle$, projections π_1, π_2, public and private key generation $\mathsf{pk}(\_), \mathsf{sk}(\_)$, encryption with random seed $\{\_\}_\_^\_$, decryption $\mathsf{dec}(\_,\_)$, $\mathsf{if\_then\_else\_}$, true, false, zero $\mathbf{0}(\_)$ and equality check $\mathsf{eq}(\_,\_)$. We give their types below:

$$\begin{array}{llll} \langle \_, \_ \rangle, \mathsf{dec}(\_,\_) & : \mathcal{S}_\mathsf{m}^2 \to \mathcal{S}_\mathsf{m} & \mathsf{eq}(\_,\_) & : \mathcal{S}_\mathsf{m}^2 \to \mathcal{S}_\mathsf{b} \\ \pi_1, \pi_2, \mathbf{0}, \mathsf{pk}, \mathsf{sk} & : \mathcal{S}_\mathsf{m} \to \mathcal{S}_\mathsf{m} & \{\_\}_\_^\_ & : \mathcal{S}_\mathsf{m}^3 \to \mathcal{S}_\mathsf{m} \\ \mathsf{if\_then\_else\_} & : \mathcal{S}_\mathsf{b} \times \mathcal{S}_\mathsf{m}^2 \to \mathcal{S}_\mathsf{m} & \mathsf{true}, \mathsf{false} & : \to \mathcal{S}_\mathsf{b} \end{array}$$

Moreover all the names in $\mathcal{N}$ have sort $\mathcal{S}_\mathsf{m}$, and each variable in $\mathcal{X}$ comes with a sort. We let $\mathcal{F}_s$ be $\mathcal{F}$ without the $\mathsf{if\_then\_else\_}$ function symbol, and for any subset $\mathcal{S}$ of the union of $\mathcal{F}$, $\mathcal{N}$ and $\mathcal{X}$, we let $\mathcal{T}(\mathcal{S})$ be the set of terms built upon $\mathcal{S}$.

b) Formulas: For every integer n, we have one predicate symbol $\sim_n$ of arity $2n$, which represents equivalence between two vectors of terms of length n. Formulas are then obtained using the usual Boolean connectives and first-order quantifiers.

c) Semantics: We use the classical first-order logic semantics: every sort is interpreted by some domain, and function symbols and predicates are interpreted as, resp., functions of the appropriate domains and relations on these domains.

We focus on a particular class of such models, the *computational models*. We informally describe the properties of a computational model $\mathcal{M}_c$ (a full description is given in [1]):

- $\mathcal{S}_\mathsf{m}$ is interpreted as the set of probabilistic polynomial time Turing machines equipped with a working tape and two random tapes ρ_1, ρ_2 (one for the protocol random values, the other for the adversary random samplings). Moreover its input is of length η (the security parameter). $\mathcal{S}_\mathsf{b}$ is the restriction of $\mathcal{S}_\mathsf{m}$ to machines that return 0 or 1.
- A name $\mathsf{n} \in \mathcal{N}$ is interpreted as a machine that, on input of length η, extracts a word of length η from the first random tape ρ_1. Furthermore we require that different names extract disjoint parts of ρ_1.
- true, false, $\mathbf{0}(\_)$, $\mathsf{eq}(\_,\_)$, and $\mathsf{if\_then\_else\_}$ are interpreted as expected. For instance, $\mathsf{eq}(\_,\_)$ takes two machines M_1, M_2, and returns M such that on input w and random tapes ρ_1, ρ_2, M returns 1 if $M_1(w, \rho_1, \rho_2) = M_2(w, \rho_1, \rho_2)$ and 0 otherwise. The function symbol $\mathbf{0}$

is interpreted as the function that, on input of length l, returns the bit-string 0^l.

- A function symbol $g \in \mathcal{G}$ with n arguments is interpreted as a function $[\![g]\!]$ such that there is a polynomial-time Turing machine M_g such that for every machines $(m_i)_{i \leq n}$ in the interpretation domains, and for every inputs w, ρ_1, ρ_2:

$$[\![g]\!]\big((m_i)_{i \leq n}\big)(w, \rho_1, \rho_2) = M_g\big((m_i(w, \rho_1, \rho_2))_{i \leq n}, \rho_2\big)$$

Observe that M_g cannot access directly the tape ρ_1.

- Protocol function symbols are interpreted as deterministic polynomial-time Turing machine. Their interpretations will be restricted using *implementation axioms* later.
- The interpretation of function symbols is lifted to terms: given an assignment σ of the variables of a term t to elements of the appropriate domains, we write $[\![t]\!]^{\sigma}_{\eta,\rho_1,\rho_2}$ the interpretation of the term with respect to η, ρ_1, ρ_2. σ is omitted when empty. We also omit the other parameters when they are irrelevant.
- The predicate $\sim_n$ is interpreted as *computational indistinguishability* $\approx$, defined by $m_1, \dots, m_n \approx m'_1, \dots, m'_n$ iff for every PPTM $\mathcal{A}$ with random tape ρ_2:

$$\Big| \mathbf{Pr}(\rho_1, \rho_2 : \mathcal{A}((m_i(1^\eta, \rho_1, \rho_2))_{1 \leq i \leq n}, \rho_2) = 1) \; - \; \mathbf{Pr}(\rho_1, \rho_2 : \mathcal{A}((m'_i(1^\eta, \rho_1, \rho_2))_{1 \leq i \leq n}, \rho_2) = 1) \Big|$$

is negligible in η (a function is negligible if it is asymptotically smaller than the inverse of any polynomial). Moreover, for all ground terms u, v, we write $\mathcal{M}_c \models u \sim v$ when $[\![u]\!] \approx [\![v]\!]$ in $\mathcal{M}_c$.

Example 1. Let $\mathsf{n}_0, \mathsf{n}_1, \mathsf{n} \in \mathcal{N}$ and $g \in \mathcal{F}$ of arity zero. For every computational model $\mathcal{M}_c$:

$$\mathcal{M}_c \models \mathsf{if}\ g()\ \mathsf{then}\ \mathsf{n}_0\ \mathsf{else}\ \mathsf{n}_1 \sim \mathsf{n}$$

Indeed, the term on the left represents the message obtained by letting the adversary choose a branch, and then sampling from n_0 or n_1 accordingly. This is semantically equivalent to directly performing a random sampling, as done on the right.

III. Axioms

We present the axioms Ax, which are of two kinds:

- *structural axioms* represent properties that hold in every computational model. This includes axioms such as the symmetry of $\sim$, or properties of the $\mathsf{if\_then\_else\_}$.
- *implementation axioms* reflect implementation assumptions, such as the functional correctness of the pair and projections (e.g. $\pi_1(\langle u\,,\,v\rangle) = u$), or cryptographic assumptions on the security primitives (e.g. IND-CCA$_2$).

All our axioms Ax are universally quantified Horn clauses. To show the unsatisfiability of $\mathsf{Ax} \wedge \vec{u} \not\sim \vec{v}$, we use resolution with a negative strategy (which is complete, see [25]). As all axioms are Horn clauses, a proof by resolution with a negative strategy can be seen as a proof tree where each node is indexed by the axiom of Ax used at this resolution step. Hence, axioms will be given as inference rules (where variables are implicitly universally quantified).

A. Equality and Structural Axioms

Some notation conventions: we use $\vec{u}$ to denote a vector of terms; and we use an infix notation for $\sim$, writing $\vec{u} \sim \vec{v}$ when $\vec{u}$ and $\vec{v}$ are of the same length.

The equality and structural axioms we present here already appeared in the literature [1], [22], [26], sometimes with slightly different formulations.

a) Equality: Computational indistinguishability is an equivalence relation (i.e. reflexive, symmetric and transitive). But we can observe that it is not a congruence. E.g. take a computational model $\mathcal{M}_c$, we know that two names n and n' are indistinguishable (since they are interpreted as independent uniform random sampling in $\{0,1\}^\eta$), and n is indistinguishable from itself. Therefore:

$$\mathcal{M}_c \models \mathsf{n} \sim \mathsf{n}' \quad \text{and} \quad \mathcal{M}_c \models \mathsf{n} \sim \mathsf{n}$$

But there is a simple PPTM that can distinguish between $\langle \mathsf{n}\,,\,\mathsf{n}\rangle$ and $\langle \mathsf{n}'\,,\,\mathsf{n}\rangle$: simply test whether the two arguments are equal, if so return 1 and otherwise return 0. Then, with overwhelming probability, this machine will guess from which distribution its input was sampled from.

Even though $\sim$ is not a congruence, we can get a congruence from it: if $\mathsf{eq}(s,t) \sim \mathsf{true}$ holds in all models then, using the semantics of $\mathsf{eq}(\_,\_)$, in every computational model $\mathcal{M}_c$, $[\![s]\!]$ and $[\![t]\!]$ are identical except for a negligible number of samplings. Hence we can replace any occurrence of s by t in a formula without changing its semantics with respect to computational indistinguishability.

We use this in our logic as follows: we let $s = t$ be a shorthand for $\mathsf{eq}(s,t) \sim \mathsf{true}$, and we introduce a set of equalities R (given in Fig. 1) and its congruence closure $=_R$. We split R in four sub-parts: R_1 contains the functional correctness assumptions on the pair and encryption; R_2 and R_3 contain, respectively, the homomorphism properties and simplification rules of the $\mathsf{if\_then\_else\_}$; and R_4 allows to change the order in which conditional tests are performed.

We then introduce a recursive set of rules:

$$\frac{\vec{u}, t \sim \vec{v}}{\vec{u}, s \sim \vec{v}}\ R \qquad (s, t \text{ ground terms with } s =_R t)$$

By orienting R_1, R_2, R_3 from left to right, and carefully choosing an orientation for the ground instances of R_4, we obtain a recursive term rewriting system $\rightarrow_R$. We have the following theorem (proven in Appendix I):

Theorem 1. *The TRS $\rightarrow_R$ is convergent on ground terms.*

b) Structural Axioms: We now give an informal description of the axioms given in Fig. 2. We describe in details the case study axiom CS, which is the most complicated one. It states that in order to show that:

$$\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v \sim \mathsf{if}\ b'\ \mathsf{then}\ u'\ \mathsf{else}\ v'$$

it is sufficient to show that the then branches and the else branches are indistinguishable, *when giving to the adversary the value of the conditional* (i.e. b on the left and b' on the right). We can do better, by considering simultaneously several

$$R_1 \begin{cases} \pi_i(\langle x_1, x_2\rangle) = x_i \qquad\qquad \mathsf{eq}(x,x) = \mathsf{true} \\ \mathsf{dec}(\{x\}^z_{\mathsf{pk}(y)}, \mathsf{sk}(y)) = x \end{cases}$$

$$R_2 \begin{cases} f(\vec{u}, \text{if } b \text{ then } x \text{ else } y, \vec{v}) = \\ \quad \text{if } b \text{ then } f(\vec{u}, x, \vec{v}) \text{ else } f(\vec{u}, y, \vec{v}) \qquad (f \in \mathcal{F}_s) \\ \text{if (if } b \text{ then } a \text{ else } c) \text{ then } x \text{ else } y = \\ \quad \text{if } b \text{ then (if } a \text{ then } x \text{ else } y) \text{ else (if } c \text{ then } x \text{ else } y) \end{cases}$$

$$R_3 \begin{cases} \text{if } b \text{ then } x \text{ else } x = x \\ \text{if true then } x \text{ else } y = x \qquad \text{if false then } x \text{ else } y = y \\ \text{if } b \text{ then (if } b \text{ then } x \text{ else } y) \text{ else } z = \text{if } b \text{ then } x \text{ else } z \\ \text{if } b \text{ then } x \text{ else (if } b \text{ then } y \text{ else } z) = \text{if } b \text{ then } x \text{ else } z \end{cases}$$

$$R_4 \begin{cases} \text{if } b \text{ then (if } a \text{ then } x \text{ else } y) \text{ else } z = \\ \quad \text{if } a \text{ then (if } b \text{ then } x \text{ else } z) \text{ else (if } b \text{ then } y \text{ else } z) \\ \text{if } b \text{ then } x \text{ else (if } a \text{ then } y \text{ else } z) = \\ \quad \text{if } a \text{ then (if } b \text{ then } x \text{ else } y) \text{ else (if } b \text{ then } x \text{ else } z) \end{cases}$$

Fig. 1. $R = R_1 \cup R_2 \cup R_3 \cup R_4$

terms starting with the same conditional. We also allow some terms $\vec{w}$ and $\vec{w}'$ on the left and right to stay untouched:

$$\frac{\vec{w}, b, (u_i)_i \sim \vec{w}', b', (u'_i)_i \qquad \vec{w}, b, (v_i)_i \sim \vec{w}', b', (v'_i)_i}{\vec{w}, (\text{if } b \text{ then } u_i \text{ else } v_i)_i \sim \vec{w}', (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_i}$$

This is the only axiom with more than one premise. Furthermore we assume that b, b' do not contain conditionals.

We quickly describe the other structural axioms: Perm allows to change the terms order, using the same permutation on both sides of $\sim$; Restr is a strengthening axiom; R allows to replace a term s by any R-equal term t; the function application axiom FA states that to prove that two images are indistinguishable, it is sufficient to show that the arguments are indistinguishable (we restrict this axiom to the case where f is in $\mathcal{F} \setminus \{\mathbf{0}\}$); Sym states that indistinguishability is symmetrical; and Dup states that giving twice the same value to an adversary is equivalent to giving it only once. All the above axioms are computationally valid.

Proposition 1. *The axioms given in Fig. 2 are valid in any computational model in which the functional correctness assumptions R_1 on pairs and encryptions hold.*

Proof. The proof can be found in [1]. ∎

c) Restrictions: As mentioned earlier, we restricted some axioms to achieve decidability. For example, the CS and FA axioms presented above are weaker than the corresponding axioms in [1]: in the CS axiom, we forbid the terms b and b' from containing conditionals; and we do not allow FA applications on the $\mathbf{0}$ function symbols. These are technical restrictions which are used in the proof, but might be unnecessary.

B. Cryptographic Assumptions

We now show how cryptographic assumptions are translated into unitary axioms. In the computational model, the security of a cryptographic primitive is expressed through a game between a challenger and an attacker (which is a PPTM) that tries to break the primitive.

We present here the IND-CCA$_2$ game (for Indistinguishability against Chosen Ciphertexts Attacks, see [19]). First, the challenger computes a public/private key pair $(\mathsf{pk}(\mathsf{n}), \mathsf{sk}(\mathsf{n}))$ (using a nonce n of length η uniformly sampled), and sends $\mathsf{pk}(\mathsf{n})$ to the attacker. The adversary then has access to two oracles: i) a left-right oracle $\mathcal{O}^b_{\mathsf{LR}}(\mathsf{n})$ that takes two messages m_0, m_1 as input and returns $\{m_b\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}$, where b is an internal bit uniformly sampled at the beginning by the challenger and n_r is a fresh nonce; ii) a decryption oracle $\mathcal{O}_{\mathsf{dec}}(\mathsf{n})$ that, given m, returns $\mathsf{dec}(m, \mathsf{sk}(\mathsf{n}))$ if m was not the result of a previous $\mathcal{O}_{\mathsf{LR}}$ oracle query, and length of m zeros otherwise. Remark that the two oracles have a shared memory. For simplicity, we omit the length constraints of these oracles (we give them in Appendix II). The advantage $\mathrm{Adv}^{\mathsf{CCA2}}_{\mathcal{A}}(\eta)$ of an adversary $\mathcal{A}$ against this game is the probability for $\mathcal{A}$ to guess the bit b:

$$\Big| \, \mathbf{Pr}\big(\mathsf{n} : \mathcal{A}^{\mathcal{O}^1_{\mathsf{LR}}(\mathsf{n}), \mathcal{O}_{\mathsf{dec}}(\mathsf{n})}(1^\eta) = 1\big) - \mathbf{Pr}\big(\mathsf{n} : \mathcal{A}^{\mathcal{O}^0_{\mathsf{LR}}(\mathsf{n}), \mathcal{O}_{\mathsf{dec}}(\mathsf{n})}(1^\eta) = 1\big) \, \Big|$$

An encryption scheme is IND-CCA$_2$ if the advantage $\mathrm{Adv}^{\mathsf{CCA2}}_{\mathcal{A}}(\eta)$ of any adversary $\mathcal{A}$ is negligible in η. The IND-CCA$_1$ game is the restriction of this game where the adversary cannot call $\mathcal{O}_{\mathsf{dec}}$ after having called $\mathcal{O}_{\mathsf{LR}}$. An encryption scheme is IND-CCA$_1$ if $\mathrm{Adv}^{\mathsf{CCA1}}_{\mathcal{A}}(\eta)$ is negligible for any adversary $\mathcal{A}$.

a) CCA1 Axiom: Before introducing the CCA2 axioms, we recall informally the CCA1 axioms from [1]. First, we define a syntactic property on secret keys used as a side-condition of the CCA1 axioms:

Definition 1. *For every ground term t, we say that a secret key $\mathsf{sk}(\mathsf{n})$ appears only in* decryption position *in t if it appears only in subterms of t of the form $\mathsf{dec}(\_, \mathsf{sk}(\mathsf{n}))$.*

We now define the CCA1 axioms:

Definition 2. CCA1 *is the computable set of unitary axioms:*

$$\vec{w}, t[\{u\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}] \sim \vec{w}, t[\{v\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}]$$

where: n_r does not appear in $t, u, v, \vec{w}$; n appears only in $\mathsf{pk}(\mathsf{n})$ or $\mathsf{sk}(\mathsf{n})$ in $t, u, v, \vec{w}$; $\mathsf{sk}(\mathsf{n})$ does not appear in $t, \vec{w}$; $\mathsf{sk}(\mathsf{n})$ appears only in decryption position in u, v; and the terms u and v are always of the same length.

Proposition 2. CCA1 *is valid in every computational model where the encryption scheme interpretation is* IND-CCA$_1$.

Proof. (*sketch*) The proof is a reduction that, given a PPTM $\mathcal{A}$ that can distinguish between $\vec{w}, t[\{u\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}]$ and $\vec{w}, t[\{v\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}]$, builds a winning adversary against the IND-CCA$_1$ game.

We define the adversary. First, it computes $[\![u]\!]$ and $[\![v]\!]$, calling the decryption oracle if necessary. It then sends them to the challenger who answers c, which is either $[\![\{u\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}]\!]$ or $[\![\{v\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}]\!]$. Observe that we need the freshness hypothesis on n_r as it is drawn by the challenger and the adversary cannot sample it. Using c, the adversary computes $[\![t[c]]\!]$, which it can do because the secret key does not appear in t, and then returns the bit $\mathcal{A}([\![t[c]]\!])$. The advantage of the adversary is exactly the advantage of $\mathcal{A}$, which we assumed non-negligible, hence the adversary wins the game. ∎

$$\frac{u_{\pi(1)},\dots,u_{\pi(n)} \sim v_{\pi(1)},\dots,v_{\pi(n)}}{u_1,\dots,u_n \sim v_1,\dots,v_n}\ \mathsf{Perm} \qquad \frac{\vec{u},t \sim \vec{v},t'}{\vec{u} \sim \vec{v}}\ \mathsf{Restr} \qquad \text{for any } s =_R t,\ \frac{\vec{u},t \sim \vec{v}}{\vec{u},s \sim \vec{v}}\ R \qquad \frac{\vec{u}_1,\vec{v}_1 \sim \vec{u}_2,\vec{v}_2}{f(\vec{u}_1),\vec{v}_1 \sim f(\vec{u}_2),\vec{v}_2}\ \mathsf{FA}$$

$$\frac{\vec{u},t \sim \vec{v},t'}{\vec{u},t,t \sim \vec{v},t',t'}\ \mathsf{Dup} \qquad \frac{\vec{v} \sim \vec{u}}{\vec{u} \sim \vec{v}}\ \mathsf{Sym} \qquad \text{for any } b,b' \in \mathcal{T}(\mathcal{F}_s,\mathcal{N}),\ \frac{\vec{w},b,(u_i)_i \sim \vec{w}',b',(u'_i)_i \qquad \vec{w},b,(v_i)_i \sim \vec{w}',b',(v'_i)_i}{\vec{w},(\text{if } b \text{ then } u_i \text{ else } v_i)_i \sim \vec{w}',(\text{if } b' \text{ then } u'_i \text{ else } v'_i)_i}\ \mathsf{CS}$$

Conventions: π is a permutation of $\{1,\dots,n\}$ and $f \in \mathcal{F} \setminus \{\mathbf{0}\}$.

Fig. 2. The Axioms Struct-Ax.

Remark 1. In the CCA1 axiom, we did not specify how we ensure that u and v are always of the same length. Since the length of a term depends on implementation details (e.g. how the pair $\langle\_\,,\,\_\rangle$ implemented), we let the user supply implementation assumptions, but require that these assumptions satisfy some properties (this is necessary to get decidability). To simplify the presentation, we omit all length constraints for now. We describe them later, in Section II-B.

b) CCA2 *Axiom:* To extend this axiom to the IND-CCA$_2$ game, we need to deal with calls to the decryption oracle performed after some calls to the left-right oracle. For example, consider the case where one call (u,v) was made. Let $\alpha \equiv \{u\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}$ and $\alpha' \equiv \{v\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}$ (where $\equiv$ denotes syntactic equality) be the result of the call on, respectively, the left and the right. A naive first try could be to state that decryptions are indistinguishable. That is, if we let $s \equiv t[\alpha]$ and $s' \equiv t[\alpha']$, then $\mathsf{dec}(s,\mathsf{sk(n)}) \sim \mathsf{dec}(s',\mathsf{sk(n)})$. But this is not valid: for example, take $u \equiv 0, v \equiv 1$, $t \equiv g([])$ (where $[]$ is a hole variable). Then the adversary can, by interpreting g as the identity function, obtain a term semantically equal to 0 on the left and 1 on the right. This allows him to distinguish between the left and right cases.

We prevent this by adding a guard checking that we are not decrypting α on the left (resp. α' on the right): if not, we return the decryption $\mathsf{dec}(s,\mathsf{sk(n)})$ (resp. $\mathsf{dec}(s',\mathsf{sk(n)})$) asked for, otherwise we return a dummy message $\mathbf{0}(\mathsf{dec}(s,\mathsf{sk(n)}))$ (resp. $\mathbf{0}(\mathsf{dec}(s',\mathsf{sk(n)}))$).

Definition 3. CCA2^s *is the (recursive) set of unitary axioms:*

$$\begin{aligned}
&\vec{w},t[\alpha],\ \textit{if } \mathit{eq}(s,\alpha) \textit{ then } \mathbf{0}(\mathit{dec}(t[\alpha],\mathit{sk}(n))) \\
&\qquad\qquad\qquad \textit{else } \mathit{dec}(t[\alpha],\mathit{sk}(n)) \\
\sim\ &\vec{w},t[\alpha'],\ \textit{if } \mathit{eq}(s',\alpha') \textit{ then } \mathbf{0}(\mathit{dec}(t[\alpha'],\mathit{sk}(n))) \\
&\qquad\qquad\qquad \textit{else } \mathit{dec}(t[\alpha'],\mathit{sk}(n))
\end{aligned}$$

under the side-conditions of Definition 2.

This axiom is valid whenever the encryption is IND-CCA$_2$.

Proposition 3. CCA2^s *is valid in every computational model where the encryption scheme interpretation is* IND-CCA$_2$.

This construction can be generalized to any number of calls to the left-right oracle, by adding a guard for each call, and to any number of keys. We refer the reader to Appendix II, where we define formally the general CCA2 axioms. [1] Still, a few comments: we use extra syntactic side-conditions to remove superfluous guards; we allow for α-renaming of names; we restrict t to be without if_then_else_ and $\mathbf{0}$; the axioms allow for an arbitrary number of public/private key pairs to be used simultaneously; and finally, an instance of the axiom can contain any number of interleaved left-right and decryption oracles calls.

Remark 2. The last point is what allows us to avoid transitivity in proofs. For example, consider four encryptions, two of them (α and γ) using the public key $\mathsf{pk(n)}$, and the other two (β and δ) using the public key $\mathsf{pk(n')}$:

$$\alpha \equiv \{A\}^{\mathsf{n}_0}_{\mathsf{pk(n)}} \quad \beta \equiv \{B\}^{\mathsf{n}_1}_{\mathsf{pk(n')}} \quad \gamma \equiv \{C\}^{\mathsf{n}_0}_{\mathsf{pk(n)}} \quad \delta \equiv \{D\}^{\mathsf{n}_1}_{\mathsf{pk(n')}}$$

Then the following formula is a valid instance of the CCA2 axioms on, simultaneously, keys $\mathsf{pk(n)}$ and $\mathsf{pk(n')}$:

$$\frac{}{\alpha,\beta \sim \gamma,\delta}\ \mathsf{CCA2}(\mathsf{pk(n)},\mathsf{pk(n')})$$

However, proving the above formula using CCA2 only on one key at a time, as in [1], requires transitivity:

$$\frac{\dfrac{}{\alpha,\beta \sim \alpha,\delta}\ \mathsf{CCA2}(\mathsf{pk(n')}) \qquad \dfrac{}{\alpha,\delta \sim \gamma,\delta}\ \mathsf{CCA2}(\mathsf{pk(n)})}{\alpha,\beta \sim \gamma,\delta}$$

C. *Comments and Examples*

Our set of axioms is not complete w.r.t. the computational interpretation semantics. Indeed, being so would mean axiomatizing exactly which distributions (computable in polynomial time) can be distinguished by PPTMs, which is unrealistic and would lead to undecidability. E.g., if we completely axiomatized IND-CCA$_2$, then showing the satisfiability of our set of axioms would show the existence of IND-CCA$_2$ functions, which is an open problem.

Still, our axioms are expressive enough to complete concrete proofs of security. We illustrate this with two simple examples: a proof of the formula in Example 1, and a proof of the security of one round of the NSL protocol [27]. Of course, such proofs can be found automatically using our decision procedure.

Example 2. We give a proof of the formula of Example 1:

$$\text{if } g() \text{ then } \mathsf{n}_0 \text{ else } \mathsf{n}_1 \sim \mathsf{n}$$

[1] Note that axioms for the IND-CCA$_2$ cryptographic assumption have already appeared in the literature, in [26]. These axioms are only for a single call to the left-right oracle, and a single key. Our axiom schema is more general.

First, we introduce a conditional $g()$ on the right to match the structure of the left side using R. Then, we split the proof in two using the CS axiom. We conclude using the reflexivity modulo α-renaming axiom (this axiom is subsumed by $\mathsf{CCA2}$, therefore we do not include it in Ax).

$$\dfrac{\dfrac{\dfrac{}{g(), \mathsf{n}_0 \sim g(), \mathsf{n}}\ \textsc{Refl} \qquad \dfrac{}{g(), \mathsf{n}_1 \sim g(), \mathsf{n}}\ \textsc{Refl}}{\text{if } g() \text{ then } \mathsf{n}_0 \text{ else } \mathsf{n}_1 \sim \text{if } g() \text{ then } \mathsf{n} \text{ else } \mathsf{n}}\ \mathsf{CS}}{\text{if } g() \text{ then } \mathsf{n}_0 \text{ else } \mathsf{n}_1 \sim \mathsf{n}}\ R$$

a) Proof of NSL: We consider a simple setting with one initiator A, one responder B and no key server. An execution of the NSL protocol is given in Fig. 3.

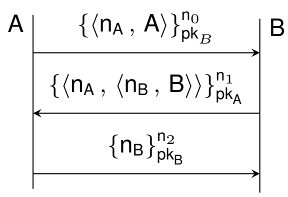


Fig. 3. The NSL protocol.

We write this in the logic. First, we let $\mathsf{pk_A} \equiv \mathsf{pk}(\mathsf{n_A})$ and $\mathsf{sk_A} \equiv \mathsf{sk}(\mathsf{n_A})$ be the public/private key pair of agent A (we define similarly $(\mathsf{pk_B}, \mathsf{sk_B})$). Since A does not wait for any input before sending its first message, we put it into the initial frame:

$$\phi_0 \equiv \mathsf{pk_A}, \mathsf{pk_B}, \{\langle \mathsf{n_A}\ ,\ \mathsf{A}\rangle\}^{\mathsf{n}_0}_{\mathsf{pk}_B}$$

Then, both agents wait for a message before sending a single reply. When receiving $\mathbf{x}_\mathsf{A}$ (resp. $\mathbf{x}_\mathsf{B}$), the answer of agent A (resp. B) is expressed in the logic as follows:

$$\begin{aligned} t_\mathsf{A}[\mathbf{x}_\mathsf{A}] \ &\equiv\ \text{if } \mathsf{eq}(\pi_1(\mathsf{dec}(\mathbf{x}_\mathsf{A}, \mathsf{sk_A})), \mathsf{n_A}) \quad \text{then} \\ &\quad\ \ \text{if } \mathsf{eq}(\pi_2(\pi_2(\mathsf{dec}(\mathbf{x}_\mathsf{A}, \mathsf{sk_A}))), \mathsf{B}) \text{ then} \\ &\quad\ \ \ \ \{\pi_1(\pi_2(\mathsf{dec}(\mathbf{x}_\mathsf{A}, \mathsf{sk_A})))\}^{\mathsf{n}_2}_{\mathsf{pk_B}} \\ t_\mathsf{B}[\mathbf{x}_\mathsf{B}] \ &\equiv\ \text{if } \mathsf{eq}(\pi_2(\mathsf{dec}(\mathbf{x}_\mathsf{B}, \mathsf{sk_B})), \mathsf{A}) \qquad\quad \text{then} \\ &\quad\ \ \ \ \{\langle \pi_1(\mathsf{dec}(\mathbf{x}_\mathsf{B}, \mathsf{sk_B}))\ ,\ \langle \mathsf{n_B}\ ,\ \mathsf{B}\rangle\rangle\}^{\mathsf{n}_1}_{\mathsf{pk_A}} \end{aligned}$$

During an execution of the protocol, the adversary has several choices. First, it decides whether to interact first with A or B. We focus on the case where it first sends a message to B (the other case is similar). Then, it can honestly forward the messages or forge new ones. E.g., when sending the first message to B, it can either forward A's message $\{\langle \mathsf{n_A}\ ,\ \mathsf{A}\rangle\}^{\mathsf{n}_0}_{\mathsf{pk}_B}$ or forge a new message. We are going to prove the security of the protocol in the following case (the other cases are similar):

- the first message, sent to B, is honest. Therefore we take $\mathbf{x}_\mathsf{B} \equiv \{\langle \mathsf{n_A}\ ,\ \mathsf{A}\rangle\}^{\mathsf{n}_0}_{\mathsf{pk}_B}$, and the answer from B is:

$$t_\mathsf{B}[\mathbf{x}_\mathsf{B}] =_R \{\langle \mathsf{n_A}\ ,\ \langle \mathsf{n_B}\ ,\ \mathsf{B}\rangle\rangle\}^{\mathsf{n}_1}_{\mathsf{pk_A}}$$

- the second message, sent to A, is forged. Therefore we take $\mathbf{x}_\mathsf{A} \equiv g(\phi_1)$, where $\phi_1 \equiv \phi_0, t_\mathsf{B}[\mathbf{x}_\mathsf{B}]$. As, a priori, nothing prevents $g(\phi_1)$ from being equal to $t_\mathsf{B}[\mathbf{x}_\mathsf{B}]$, we use the conditional $\mathsf{eq}(g(\phi_1), t_\mathsf{B}[\mathbf{x}_\mathsf{B}])$ to ensure that this message is forged. The answer from A is then:

$$s \equiv \text{if } \mathsf{eq}(g(\phi_1), t_\mathsf{B}[\mathbf{x}_\mathsf{B}]) \text{ then } 0 \text{ else } t_\mathsf{A}[g(\phi_1)] \qquad (1)$$

We show the secrecy of the nonce $\mathsf{n_B}$: we let $t'_\mathsf{B}[\mathbf{x}_\mathsf{B}]$ (resp. s') be the term $t_\mathsf{B}[\mathbf{x}_\mathsf{B}]$ (resp. s) where we replaced all occurrences of $\mathsf{n_B}$ by 0. For example, $t'_\mathsf{B}[\mathbf{x}_\mathsf{B}] =_R \{\langle \mathsf{n_A}\ ,\ \langle 0\ ,\ \mathsf{B}\rangle\rangle\}^{\mathsf{n}_1}_{\mathsf{pk_A}}$. This yields the following goal formula:

$$\phi_0, t_\mathsf{B}[\mathbf{x}_\mathsf{B}], s \sim \phi_0, t'_\mathsf{B}[\mathbf{x}_\mathsf{B}], s' \qquad (2)$$

Remark 3. The process of computing the formula from the protocol description can be done automatically, using a simple procedure similar to the folding procedure from [1]. The formula in (2) has already been split between the honest and dishonest cases using the case study axiom CS (we omit the CS applications to keep the proof readable). For example, the term in (1) is the "else" branch of a CS application on conditional $\mathsf{eq}(g(\phi_1), t_\mathsf{B}[\mathbf{x}_\mathsf{B}])$ (which does not contain nested conditionals, as required by the CS side-condition).

We now proceed with the proof. We let δ be the guarded decryption that will be used in the $\mathsf{CCA2}$ axiom:

$$\begin{aligned} \delta \ \equiv\ &\text{if } \mathsf{eq}(g(\phi_1), t_\mathsf{B}[\mathbf{x}_\mathsf{B}]) \text{ then } \mathbf{0}(\mathsf{dec}(g(\phi_1), \mathsf{sk_A})) \\ &\text{else } \mathsf{dec}(g(\phi_1), \mathsf{sk_A}) \end{aligned} \qquad (3)$$

and s_δ be the term s where all occurrences of $\mathsf{dec}(g(\phi_1), \mathsf{sk_A})$ have been replaced by δ. We have $s =_R s_\delta$. We also introduce shorthands for some subterms of s_δ: we let a_δ, b_δ and e_δ be the terms $\mathsf{eq}(\pi_1(\delta), \mathsf{n_A})$, $\mathsf{eq}(\pi_2(\pi_2(\delta))), \mathsf{B})$ and $\{\pi_1(\pi_2(\delta))\}^{\mathsf{n}_2}_{\mathsf{pk_B}}$. We define $\delta', s'_{\delta'}, a'_{\delta'}, b'_{\delta'}$ and $e'_{\delta'}$ similarly.

We then rewrite s and s' into s_δ and $s'_{\delta'}$ using R. Then we apply FA several times, first to deconstruct s_δ and $s'_{\delta'}$, and then to deconstruct a_δ, b_δ and $a'_{\delta'}, b'_{\delta'}$. Finally, we use Dup to remove duplicates, and we apply $\mathsf{CCA2}$ simultaneously on key pairs $(\mathsf{pk_A}, \mathsf{sk_A})$ and $(\mathsf{pk_B}, \mathsf{sk_B})$ (we omit here the details of the syntactic side-conditions that have to be checked):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\phi_0, t_\mathsf{B}[\mathbf{x}_\mathsf{B}], \mathsf{n_A}, \delta, e_\delta \sim \phi_0, t'_\mathsf{B}[\mathbf{x}_\mathsf{B}], \mathsf{n_A}, \delta', e'_{\delta'}}\ \mathsf{CCA2}}{\phi_0, t_\mathsf{B}[\mathbf{x}_\mathsf{B}], a_\delta, b_\delta, e_\delta \sim \phi_0, t'_\mathsf{B}[\mathbf{x}_\mathsf{B}], a'_{\delta'}, b'_{\delta'}, e'_{\delta'}}\ (\mathsf{FA}, \mathsf{Dup})^*}{\phi_0, t_\mathsf{B}[\mathbf{x}_\mathsf{B}], s_\delta \sim \phi_0, t'_\mathsf{B}[\mathbf{x}_\mathsf{B}], s'_{\delta'}}\ (\mathsf{FA}, \mathsf{Dup})^*}{\phi_0, t_\mathsf{B}[\mathbf{x}_\mathsf{B}], s \sim \phi_0, t'_\mathsf{B}[\mathbf{x}_\mathsf{B}], s'}\ R$$

IV. Main Result and Difficulties

We let Ax be the conjunction of $\mathsf{Struct\text{-}Ax}$ and $\mathsf{CCA2}$. We now state the main result of this paper.

Theorem (Main Result)**.** *The following problem is decidable:*
Input: *A ground formula $\vec{u} \sim \vec{v}$.*
Question: *Is $\mathsf{Ax} \wedge \vec{u} \not\sim \vec{v}$ unsatisfiable?*

We give here an overview of the problems that have to be overcome in order to obtain the decidability result. Before starting, a few comments. We close all rules under permutations. The Sym rule commutes with all the other rules, and the $\mathsf{CCA2}$ unitary axioms are closed under Sym. Therefore we can remove Perm and Sym from the set of rules. Observe that CS, FA, Dup and $\mathsf{CCA2}$ are all *decreasing rules*, i.e. the premises are smaller than the conclusion. The only non-decreasing rules are R, which may rewrite a term into a larger one, and Restr, which we eliminate later. Therefore we now focus on R.

a) Necessary Introductions: As we saw in Example 2, it might be necessary to use R in the "wrong direction", typically to introduce new conditionals. A priori, this yields an unbounded search space. Therefore our goal is to characterize in which situations we need to use R in the "wrong direction", and with which instances. We identify two necessary reasons for introducing new conditionals.

First, to match the shape of the term on the other side, like $g()$ in Example 2. In this case, the introduced conditional is exactly the conditional that appeared on the other side of $\sim$. With more complex examples this may not be the case. Nonetheless, an introduced conditional is always bounded by the conditional it matches.

Second, we might introduce a guard in order to fit to the definition of safe decryptions in the $\mathsf{CCA2}$ axioms, as in (3). Here also, the introduced guard will be of bounded size. Indeed, guards of $\mathsf{dec}(s, \mathsf{sk})$ are of the form $\mathsf{eq}(s, \alpha)$ where α is a subterm of s. Therefore, for a fixed s, there are a bounded number of them, and they are of bounded size.

Example 3 (Cut Elimination). These conditions are actually sufficient. We illustrate this on an example where the CS rule is applied on two conditionals that have just been introduced.

$$\dfrac{\dfrac{a, s \sim b, t \qquad a, s \sim b, t}{\text{if } a \text{ then } s \text{ else } s \sim \text{if } b \text{ then } t \text{ else } t}\ \mathsf{CS}}{s \sim t}\ R$$

Here a and b can be of arbitrary size. Intuitively, this is not a problem since any proof of $a, s \sim b, t$ includes a proof of $s \sim t$. Formally, we have the following weakening lemma.

Lemma 1. *For every proof P of a ground formula $\vec{u}, s \sim \vec{v}, t$, there exists a proof P' of $\vec{u} \sim \vec{v}$ where P' is no larger than P.*

Proof. (*sketch*) The full proof is in Appendix III. We prove by induction on P that the Restr rule is admissible using $\mathsf{Ax}\backslash\{\mathsf{Restr}\}$. For this to work, we need the $\mathsf{CCA2}$ axioms to be closed under Restr. Note that this creates some problems, which are dealt with in Appendix II-A. ■

Using this lemma, we can deal with Example 3 by doing a proof cut elimination. More generally, by induction on the proof size, we can guarantee that no such proof cuts appear.

This is the strategy we are going to follow: look for proof cuts that introduce unbounded new terms, eliminate them, and show that after sufficiently many cut eliminations all the subterms appearing in the proof are bounded by the (R-normal form of the) conclusion.

But a proof may contain more complex behaviors than just the introduction of a conditional followed by a CS application. For example the conditional being matched could have been itself introduced earlier to match another conditional, which itself was introduced to match a third conditional etc.

Example 4. We illustrate this on an example. When it is more convenient, we write terms containing only if_then_else_ and other subterms (handled as constants) as binary trees; we also index some subterms with a number, which helps keeping track of them across rule applications.

$$\dfrac{\dfrac{a_1, b_2, b_3, u_4, w_5, u_6, v_7 \sim d_1, c_2, d_3, s_4, t_5, r_6, p_7}{\begin{array}{ccccccc} &&&&a_1&& \\ &&&/&&\backslash& \\ &&b_2&&&&v_7 \\ &/&&\backslash&&& \\ u_4&&&&b_3&& \\ &&&/&&\backslash& \\ &&w_5&&&&u_6 \end{array} \quad \sim \quad \begin{array}{ccccccc} &&&&d_1&& \\ &&&/&&\backslash& \\ &&c_2&&&&p_7 \\ &/&&\backslash&&& \\ s_4&&&&d_3&& \\ &&&/&&\backslash& \\ &&t_5&&&&r_6 \end{array}}\ \mathsf{FA}^{(3)}}{\text{if } a \text{ then } u \text{ else } v \sim \text{if } c \text{ then } s \text{ else } t}\ R$$

where $p_7 \equiv$ if c then s else t. Here the conditionals b, d and the terms w, r are, a priori, arbitrary. Therefore we would like to bound them or remove them through a cut elimination. The cut elimination technique used in Example 3 does not apply here because we cannot extract a proof of $a \sim c$.

But we can extract a proof of $b_2, b_3 \sim c_2, d_3$. Using Proposition 1, this means that in every appropriate computational model, $[\![b, b]\!] \approx [\![c, d]\!]$. It means that no adversary can distinguish between getting twice the same value sampled from $[\![b]\!]$ and getting a pair of values sampled from $[\![c, d]\!]$. In particular, this means that $[\![c]\!]_{\eta,\rho} = [\![d]\!]_{\eta,\rho}$, except for a negligible number of random tapes ρ.

b) A First Key Lemma: A natural question is to ask whether this semantic equality $[\![c]\!] = [\![d]\!]$ implies a syntactic equality. While this is not the case in general, there are fragments of our logic in which this holds. We annotate the rules FA by the function symbol involved, and we let $\mathsf{FA_s} = \{\mathsf{FA}_f \mid f \in \mathcal{F}_s\}$.

Definition 4. *Let Σ be the set of axiom names, seen as an alphabet. For all $\mathcal{L} \subseteq \Sigma^*$, we let $\mathfrak{F}(\mathcal{L})$ be the fragment of our logic defined by: a formula ϕ is in the fragment iff there exists a proof P such that $P \vdash \phi$ and, for every branch ρ of P, the word w obtained by collecting the axiom names along ρ (starting from the root) is in $\mathcal{L}$.*

Lemma 2. *For all b, b', b'', if $b, b \sim b', b''$ is in the fragment $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2})$ then $b' \equiv b''$.*

Proof. The proof relies on the shape of the $\mathsf{CCA2}$ axioms, and can be found in Appendix IV. ■

Using this lemma, we can deal with Example 4 if $a_1, b_2, b_3 \sim d_1, c_2, d_3$ lies in the fragment $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2})$. Using a first time the lemma on $b_2, b_3 \sim c_2, d_3$ we obtain $c \equiv d$, and using again the lemma on $a_1, b_2 \sim d_1, c_2$ (since $d \equiv c$) we deduce $a \equiv b$. Hence the cut elimination introduced before applies.

c) Proof Sketch: We now state the sketch of the proof:

- **Commutations:** first we show that we can assume that rules are applied in some given order. We prove this by showing some commutation results and adding new rules.
- **Proof Cut Eliminations:** through proof cut eliminations, we guarantee that every conditional appearing in the proof is α-bounded. Intuitively a conditional is α-bounded if it is a subterm of the conclusion or if it guards a decryption appearing in an α-bounded term.

- **Decision Procedure:** we give a procedure that, given a goal formula $t \sim t'$, computes the set of α-bounded terms for this formula. We show that this procedure computes a finite set, and deduce that the proof search is finite. This yields an effective algorithm to decide our problem.

V. Commutations and Cut Eliminations

In this section we show, through rule commutations, that we can restrict ourselves to proofs using rules in some given order. Then, we show how this restricts the shapes of the terms.

A. Rule Commutations

Everything in this subsection applies to any set U of unitary axioms closed under Restr. We specialize to $\mathsf{CCA2}$ later.

We start by showing a set of rule commutations of the form $w \Rightarrow w'$, where w and w' are words over the set of rule names. An entry $w \Rightarrow w'$ means that a derivation in w can be rewritten into a derivation in w', with the same conclusion and premises. Here are the basic commutations we use:

$$\boxed{\begin{array}{ccc} \mathsf{Dup}\cdot R & \Rightarrow & R\cdot\mathsf{Dup} \\ \mathsf{Dup}\cdot\mathsf{FA} & \Rightarrow & \mathsf{FA}^*\cdot\mathsf{Dup} \\ \mathsf{Dup}\cdot\mathsf{CS} & \Rightarrow & \mathsf{CS}\cdot\mathsf{Dup} \end{array}} \qquad \boxed{\begin{array}{ccc} \mathsf{FA}\cdot R & \Rightarrow & R\cdot\mathsf{FA} \\ \mathsf{FA}\cdot\mathsf{CS} & \Rightarrow & R\cdot\mathsf{CS}\cdot\mathsf{FA} \end{array}}$$

Lemma 3. *All the above rule commutations are correct.*

Proof. We show only $\mathsf{FA}\cdot R \Rightarrow R\cdot\mathsf{FA}$ (the full proof is in Appendix III):

$$\dfrac{\dfrac{\vec{u}_1,\vec{v}_1 \sim \vec{u}'_1,\vec{v}'_1}{\vec{u},\vec{v} \sim \vec{u}',\vec{v}'}\,R}{\vec{u},f(\vec{v}) \sim \vec{u}',f(\vec{v}')}\,\mathsf{FA} \quad \Rightarrow \quad \dfrac{\dfrac{\vec{u}_1,\vec{v}_1 \sim \vec{u}'_1,\vec{v}'_1}{\vec{u}_1,f(\vec{v}_1) \sim \vec{u}'_1,f(\vec{v}'_1)}\,\mathsf{FA}}{\vec{u},f(\vec{v}) \sim \vec{u}',f(\vec{v}')}\,R$$

∎

Using these rules, we obtain a first restriction.

Lemma 4. *The ordered strategy $\mathfrak{F}((\mathsf{CS}+R)^*\cdot\mathsf{FA}^*\cdot\mathsf{Dup}^*\cdot\mathsf{U})$ is complete for $\mathfrak{F}((\mathsf{CS}+\mathsf{FA}+R+\mathsf{Dup}+\mathsf{U})^*)$.*

Proof. First, we commute all the Dup to the right, which yields $\mathfrak{F}((\mathsf{CS}+R+\mathsf{FA})^*\cdot\mathsf{Dup}^*\cdot\mathsf{U})$. Then, we commute all FA to the right, stopping at the first Dup. ∎

a) Splitting the FA *Rule:* To go further, we split FA as follows: if the deconstructed symbol is $\mathsf{if\_then\_else\_}$ then we denote the function application by $\mathsf{FA}(b,b')$, where b,b' are the involved conditionals; if the deconstructed symbol f is in $\mathcal{F}_s$, then we denote the function application by FA_f. We give below the two new rules:

$$\dfrac{\vec{w},a,u,v \sim \vec{r},b,s,t}{\begin{array}{c}\vec{w},\text{if } a \text{ then } u \text{ else } v\\ \sim \vec{r},\text{if } b \text{ then } s \text{ else } t\end{array}}\,\mathsf{FA}(b,b') \qquad \dfrac{\vec{u},\vec{v} \sim \vec{s},\vec{t}}{\vec{u},f(\vec{v}) \sim \vec{s},f(\vec{t})}\,\mathsf{FA}_f$$

The set of rule names is now infinite, since there exists one rule $\mathsf{FA}(b,b')$ for every pair of ground terms b,b'.

b) Further Commutations: Intuitively, we want to use R at the beginning of the proof only. This is helpful since, as we observed earlier, all the other rules are decreasing (i.e. premises are smaller than the conclusion). The problem is that we cannot fully commute CS and R. For example, in:

$$\dfrac{\dfrac{a',u' \sim b',s'}{a,u \sim b,s}\,R \qquad \dfrac{a'',v' \sim b'',t'}{a,v \sim b,t}\,R}{\text{if } a \text{ then } u \text{ else } v \sim \text{if } b \text{ then } s \text{ else } t}\,\mathsf{CS}$$

we can commute the rewritings on u,v,s and t, but not on a and b because they appear twice in the premises, and a' and a'' may be different (same for b' and b'').

c) New Rules: We handle this problem by adding new rules to track relations between branches. We give only simplified versions here, the full rules are in Appendix III. For every a,c in $\mathcal{T}(\mathcal{F}_s,\mathcal{N})$ in R-normal form, we have the rules:

$$\dfrac{\vec{u},C\left[\boxed{a}\boxed{a}_a\right] \sim \vec{v},C'\left[\boxed{c}\boxed{c}_c\right]}{\vec{u},C[a] \sim \vec{v},C'[c]}\,\mathsf{2Box}^{\mathsf{s}}$$

$$\dfrac{a_1,u \sim c_1,s \qquad a_2,v \sim c_2,t}{\text{if } \boxed{a_1}\boxed{a_2}_a \text{ then } u \text{ else } v \sim \text{if } \boxed{c_1}\boxed{c_2}_c \text{ then } s \text{ else } t}\,\mathsf{CS}^{\mathsf{s}}_{\Box}$$

where $\boxed{}\boxed{}_a$ is a new symbol of sort $\mathcal{S}_{\mathsf{b}}^2 \to \mathcal{S}_{\mathsf{b}}$, and of fixed semantics: it ignores its arguments and has the semantics $[\![a]\!]$. Intuitively, $\boxed{a_1}\boxed{a_2}_a$ stands for the conditional a, and a_1,a_2 are, respectively, the left and right versions of a.

Remark that for the $\mathsf{CS}_{\Box}$ rule to be sound we need $[\![a_1]\!]$, $[\![a_2]\!]$ and $[\![a]\!]$ to be equal, up to a negligible number of samplings (same for c_1,c_2 and c). This is not enforced by the rules, so it has to be an invariant of our strategy. We denote $\mathcal{B}$ the set of new function symbols. We need the functions in $\mathcal{B}$ to block the if-homomorphism to ensure that for all $\boxed{a}\boxed{c}_b \in \mathsf{st}(t)$, $[\![a]\!] = [\![c]\!] = [\![b]\!]$. Therefore the TRS R_2 is *not* extended to $\mathcal{B}$. For example we have:

$$\boxed{\text{if } a \text{ then } c \text{ else } d}\boxed{e}_b \not\to^*_R \text{if } a \text{ then } \boxed{c}\boxed{e}_b \text{ else } \boxed{d}\boxed{e}_b$$

The R rule is replaced by $R_{\Box}$ which has an extra side-condition. $R_{\Box}$ can rewrite $u[s]$ into $u[t]$ as long as:

$$\left\{\boxed{a}\boxed{c}_b \in \mathsf{st}(t)\right\} \subseteq \left\{\boxed{a}\boxed{c}_b \in \mathsf{st}(u[s])\right\}$$

This ensures that no new arbitrary $\boxed{a}\boxed{c}_b$ is introduced. New boxed conditionals are only introduced through the $\mathsf{2Box}$ rule. Similarly, the FA axiom is *not* extended to $\mathcal{B}$.

Definition 5. *A term t is* well-formed *if for every $\boxed{a}\boxed{c}_b \in \mathsf{st}(t)$, $a =_R c =_R b$. We lift this to formulas as expected.*

Proposition 4. *The following rules preserve well-formedness:*

$$R_{\Box}, \mathsf{2Box}, \mathsf{CS}_{\Box}, \mathsf{FA_s}, \{\mathsf{FA}(b,b')\}, \mathsf{Dup}$$

Besides, $R_{\Box}$, $\mathsf{CS}_{\Box}$ and $\mathsf{2Box}$ are sound on well-formed formulas.

Proof. The only rule not obviously preserving well-formedness is $R_{\Box}$, but its side-conditions guarantee the well-formedness invariant. The only rule that is not always sound is $\mathsf{CS}_{\Box}$, and it is trivially sound on well-formed formulas. ∎

d) Ordered Strategy: We have new rule commutations.

$\mathsf{FA_s} \cdot \mathsf{FA}(b, b')$	$\Rightarrow$	$R \cdot \mathsf{FA}(b, b') \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}$
$\mathsf{CS}_\Box \cdot R_\Box$	$\Rightarrow$	$R_\Box \cdot \mathsf{CS}_\Box$
$\mathsf{CS}_\Box \cdot \mathsf{2Box}$	$\Rightarrow$	$R_\Box \cdot \mathsf{2Box} \cdot \mathsf{CS}_\Box$

Lemma 5. *All the rule commutations above are correct.*

Proof. The proof can be found in Appendix III. ∎

This allows to have $R_\Box$ rules only at the beginning of the proof.

Lemma 6. *The ordered strategy:*

$$\mathfrak{F}((\mathsf{2Box} + R_\Box)^* \cdot \mathsf{CS}_\Box^* \cdot \{\mathsf{FA}(b, b')\}^* \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{U})$$

is complete for $\mathfrak{F}((\mathsf{CS} + \mathsf{FA} + R + \mathsf{Dup} + \mathsf{U})^*)$.

Proof. We start from the result of Lemma 4, split the FA rules and commute rules until we get:

$$\mathfrak{F}((\mathsf{CS} + R)^* \cdot \{\mathsf{FA}(b, b')\}^* \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{U})$$

We then replace all applications of CS by $\mathsf{2Box}.\mathsf{CS}_\Box$. All $\boxed{a}\boxed{a}_{a}$ introduced are immediately "opened" by a $\mathsf{CS}_\Box$ application, hence we know that the side-conditions of $R_\Box$ hold every time we apply R. Therefore we can replace all applications of R by $R_\Box$, which yields:

$$\mathfrak{F}((\mathsf{CS}_\Box + \mathsf{2Box} + R_\Box)^* \cdot \{\mathsf{FA}(b, b')\}^* \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{U})$$

Finally we commute the $\mathsf{CS}_\Box$ applications to the right. ∎

B. The Freeze Strategy

We now show that we can restrict the terms on which the rules in $\{\mathsf{FA}(b, b')\}$ can be applied: when we apply a rule in $\{\mathsf{FA}(b, b')\}$, we "freeze" the conditionals b and b' to forbid any further applications of $\{\mathsf{FA}(b, b')\}$ to them.

Example 5. Let $a_i \equiv \mathsf{if}\ b_i\ \mathsf{then}\ c_i\ \mathsf{else}\ d_i$ $(i \in \{1, 2\})$, we want to forbid the following partial derivation to appear:

$$\dfrac{\dfrac{b_1, c_1, d_1, u_1, v_1 \sim b_2, c_2, d_2, u_2, v_2}{a_1, u_1, v_1 \sim a_2, u_2, v_2}\ \mathsf{FA}(b_1, b_2)}{\mathsf{if}\ a_1\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \sim \mathsf{if}\ a_2\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2}\ \mathsf{FA}(a_1, a_2)$$

a) Freeze Strategy: We let $^-$ be a new function symbol of arity one, and for every ground term s we let $\widetilde{s}$ be the term:

$$\widetilde{s} \equiv \begin{cases} \mathsf{if}\ \bar{b}\ \mathsf{then}\ u\ \mathsf{else}\ v & \text{if } s \equiv \mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v \\ s & \text{if } s \in \mathcal{T}(\mathcal{F}_s, \mathcal{N}) \end{cases}$$

Moreover we replace every $\mathsf{FA}(b_1, b_2)$ rule by the rule:

$$\dfrac{\vec{w}_1, \widetilde{b_1}, u_1, v_1 \sim \vec{w}_2, \widetilde{b_2}, u_2, v_2}{\vec{w}_1, \mathsf{if}\ b_1\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \sim \vec{w}_2, \mathsf{if}\ b_2\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2}\ \mathsf{BFA}(b_1, b_2)$$

We let $\{\overline{\mathsf{BFA}}(b_1, b_2)\}$ be the restriction of $\{\mathsf{BFA}(b_1, b_2)\}$ to the rules where b_1 and b_2 are not frozen conditionals. Finally, we add a new rule, UnF, which unfreezes all conditionals: every $\bar{b}$ is replaced by b.

Lemma 7. *The following strategy:*

$$\mathfrak{F}((\mathsf{2Box} + R_\Box)^* \cdot \mathsf{CS}_\Box^* \cdot \{\overline{\mathsf{BFA}}(b, b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{U})$$

is complete for $\mathfrak{F}((\mathsf{CS} + \mathsf{FA} + R + \mathsf{Dup} + \mathsf{U})^*)$.

Proof. Basically, the proof consists in eliminating all proof cuts of the shape given in Example 5. The cut elimination is simple, though voluminous, and is given in Appendix III. ∎

VI. Proof Form and Key Properties

The goal of this section is to show that we can assume w.l.o.g. that the terms appearing in the proof (following the ordered freeze strategy) after the $(\mathsf{2Box} + R_\Box)^*$ part have a particular form, that we call proof form. We also show properties of this restricted shape that allow more cut eliminations.

A. Shape of the Terms

Most of the completeness results shown before are for any set of unitary axioms closed under Restr. We now specialize these results to $\mathsf{CCA2}$, to get some further restrictions.

When applying the unitary axioms $\mathsf{CCA2}$, we would like to require that terms are in R-normal form, e.g. to avoid the application of $\mathsf{CCA2}$ to terms with an unbounded component, such as $\pi_1(\langle u\ ,\ v\rangle)$. Unfortunately, the side-conditions of $\mathsf{CCA2}$ are not stable under R. E.g., consider the $\mathsf{CCA2}$ instance:

$$\dfrac{}{\{\mathsf{if}\ \mathsf{eq}(g(\mathsf{n}_u), \mathsf{n}_u)\ \mathsf{then}\ A\ \mathsf{else}\ B\}_{\mathsf{pk(n)}}^{\mathsf{n}_r} \sim \{C\}_{\mathsf{pk(n)}}^{\mathsf{n}_r}}\ \mathsf{CCA2}$$

The R-normal form of the left term is:

$$\mathsf{if}\ \mathsf{eq}(g(\mathsf{n}_u), \mathsf{n}_u)\ \mathsf{then}\ \{A\}_{\mathsf{pk(n)}}^{\mathsf{n}_r}\ \mathsf{else}\ \{B\}_{\mathsf{pk(n)}}^{\mathsf{n}_r}$$

which cannot be used in a valid $\mathsf{CCA2}$ instance, since the conditional $\mathsf{eq}(g(\mathsf{n}_u), \mathsf{n}_u)$ should be somehow "hidden" by the encryption. To avoid this difficulty, we use a different normal form for terms: we try to be as close as possible to the R-normal form, while keeping conditional branching below their encryption. First, we illustrate this on an example. The term:

$$\left\{\mathsf{if}\ (\mathsf{if}\ b\ \mathsf{then}\ a\ \mathsf{else}\ c)\ \mathsf{then}\ \{\mathsf{if}\ d\ \mathsf{then}\ u\ \mathsf{else}\ v\}_{\mathsf{pk}}^{\mathsf{n}_1}\ \mathsf{else}\ w\right\}_{\mathsf{pk}}^{\mathsf{n}_2}$$

is normalized as follows:

$$\left\{\begin{array}{l} \mathsf{if}\ b\ \mathsf{then}\ \mathsf{if}\ a\ \mathsf{then}\ \{\mathsf{if}\ d\ \mathsf{then}\ u\ \mathsf{else}\ v\}_{\mathsf{pk}}^{\mathsf{n}_1}\ \mathsf{else}\ w \\ \quad \mathsf{else}\ \mathsf{if}\ c\ \mathsf{then}\ \{\mathsf{if}\ d\ \mathsf{then}\ u\ \mathsf{else}\ v\}_{\mathsf{pk}}^{\mathsf{n}_1}\ \mathsf{else}\ w \end{array}\right\}_{\mathsf{pk}}^{\mathsf{n}_2}$$

a) Basic Terms: We omit the rewriting strategy here (C.f. Appendix IV) , and describe instead the properties of the normalized terms. We let $\mathcal{A}_\succ$ be the ordered strategy from Lemma 7, and $\mathcal{A}_{\mathsf{CS}_\Box}$ be its restriction to proofs with an empty $(\mathsf{2Box} + R_\Box)^*$ part. The rule $\mathsf{CS}_\Box$ is the only branching rule, therefore, after applying all the $\mathsf{CS}_\Box$ rules, we can associate to each branch l of the proof an instance $\mathcal{S}_l = (\mathcal{K}_l, \mathcal{R}_l, \mathcal{E}_l, \mathcal{D}_l)$ of the $\mathsf{CCA2}$ axiom, where $\mathcal{K}_l$, $\mathcal{R}_l$, $\mathcal{E}_l$ and $\mathcal{D}_l$ are the sets of, respectively, secret keys, encryption randomness, encryptions and decryptions. We use $\mathcal{S}_l$ to define a normal form for the terms appearing in branch l. This is done through four mutually inductive definitions: *$\mathcal{S}_l$-encryption oracle calls* are well-formed encryptions; *$\mathcal{S}_l$-decryption oracle calls* are well-formed decryptions; *$\mathcal{S}_l$-normalized basic terms* are terms built using function symbols in $\mathcal{F}_s$ and well-formed encryptions and decryptions; and *$\mathcal{S}_l$-normalized simple terms* are combinations of normalized basic terms using

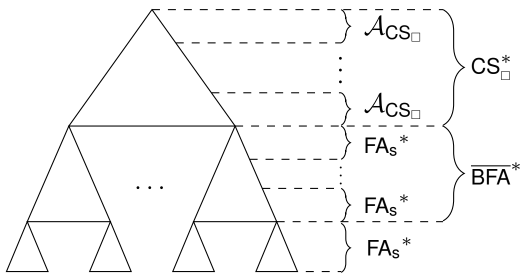


Fig. 4. The shape of the term is determined by the proof.

if_then_else_. We give only the definition of $\mathcal{S}_l$-normalized basic terms (the full definitions are in Appendix IV).

Definition 6. *A* $\mathcal{S}_l$-normalized basic term *is a term t of the form $U[\vec{w}, (\alpha_j)_j, (\mathsf{dec}_k)_k]$ where:*

- *U and $\vec{w}$ are if-free and $\mathcal{R}_l, \mathcal{K}_l$ do not appear in $\vec{w}$.*
- *$U[\vec{w}, (\{[]_j\}^{\mathsf{n}_j}_{\mathsf{pk}_j})_j, (\mathsf{dec}([]_k, \mathsf{sk}_k))_k]$ is in R-normal form.*
- *$(\alpha_j)_j$ are* $\mathcal{S}_l$-encryption oracle calls *under $(\mathsf{pk}_j, \mathsf{sk}_j)_j$.*
- *$(\mathsf{dec}_k)_k$ are* $\mathcal{S}_l$-decryption oracle calls *under $(\mathsf{pk}_k, \mathsf{sk}_k)_k$.*

If t is of sort bool, we say that it is a $\mathcal{S}_l$-normalized basic conditional*.*

b) Normalized Proof Form: Every application of $\mathsf{CS}_\square$:

$$\frac{a_1, u \sim b_1, s \qquad a_2, v \sim b_2, t}{\text{if } \boxed{a_1 \mid a_2}_{\,a} \text{ then } u \text{ else } v \sim \text{if } \boxed{b_1 \mid b_2}_{\,b} \text{ then } s \text{ else } t}\ \mathsf{CS}_\square$$

is such that if we extract the sub-proof of $a_i \sim b_i$ (for $i \in \{1, 2\}$), we get a proof in $\mathcal{A}_{\mathsf{CS}_\square}$. Therefore, we can check that terms after $(\mathsf{2Box} + R_\square)^*$ are of the form informally described in Fig. 4. We define a normal form for such proofs, called *normalized proof form*, and we define $\vdash^{\mathsf{npf}}$ by $P \vdash^{\mathsf{npf}} t \sim t'$ if and only if $P \vdash t \sim t'$, the proof P is in $\mathcal{A}_\succ$ and is in *normalized proof form*. We do not give the full definition, but one of the key ingredients is to require that for every term s appearing in a branch l of the proof P, if s is the conclusion of a sub-proof in the fragment $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{U})$ then s is a $\mathcal{S}_l$-normalized basic term.

Lemma 8. *Every formula in $\mathfrak{F}((\mathsf{CS} + \mathsf{FA} + R + \mathsf{Dup} + \mathsf{CCA2})^*)$ is provable using the strategy $\vdash^{\mathsf{npf}}$.*

Proof. (*sketch*) The full proof is in Appendix IV. First, we rewrite terms by pulling conditionals upward without crossing an encryption function symbol, and without modifying decryption guards. Then, we remove all redexes from R_1 (e.g. $\pi_1(\langle u\,,\,v\rangle) \to u$) using a cut elimination procedure. E.g., the following cut can be eliminated using Lemma 1:

$$\cfrac{\cfrac{u, v \sim u', v'}{\pi_1(\langle u\,,\,v\rangle) \sim \pi_1(\langle u'\,,\,v'\rangle)}\ \mathsf{FA}_{\langle\,,\,\rangle}}{u \sim u'}\ R$$

■

B. Key Properties

A term in R-normal form is in the following grammar:

$$t ::= u \in \mathcal{T}(\mathcal{F}_s, \mathcal{N}) \mid \text{if } b \text{ then } t \text{ else } t \ \ (\text{with } b \in \mathcal{T}(\mathcal{F}_s, \mathcal{N}))$$

Given a term t in R-normal form, we let $\mathsf{cond\text{-}st}(t)$ be its set of conditionals, and $\mathsf{leave\text{-}st}(t)$ its set of leaves.

a) Characterization of Basic Terms: We give a key characterization proposition for basic terms: if two $\mathcal{S}_l$-normalized basic terms β and β' are such that, when R-normalizing them, they share a leaf term, then they are identical.

Proposition 5. *For all $\mathcal{S}_l$-normalized basic terms β, β', if we have $\mathsf{leave\text{-}st}(\beta\downarrow_R) \cap \mathsf{leave\text{-}st}(\beta'\downarrow_R) \neq \emptyset$ then $\beta \equiv \beta'$.*

Proof. (*sketch*) The full proof is in Appendix V. We give the intuition: since they are $\mathcal{S}_l$-normalized basic terms, we know that $\beta \equiv U[\vec{w}, (\alpha_j)_j, (\mathsf{dec}_k)_k]$, $\beta' \equiv U'[\vec{w}', (\alpha'_j)_j, (\mathsf{dec}'_k)_k]$ and:

$$U[\vec{w}, (\{[]_j\}^{\mathsf{n}_j}_{\mathsf{pk}_j})_j, (\mathsf{dec}([]_k, \mathsf{sk}_k))_k]$$
$$U'[\vec{w}', (\{[]'_j\}^{\mathsf{n}_j}_{\mathsf{pk}'_j})_j, (\mathsf{dec}([]'_k, \mathsf{sk}'_k))_k]$$

are in R-normal form. Using the fact that $U, U', \vec{w}, \vec{w}'$ are if-free, and the hypothesis that β and β' share a leaf term, we first show that we can assume $U \equiv U'$ and $\vec{w} \equiv \vec{w}'$ by induction on the number of positions where U and U' differ. Take p where they differ, w.l.o.g. assume $\beta'_{|p}$ to be a hole of U' (otherwise swap β and β'). We have three cases: i) if $\beta_{|p}$ is in $\vec{w}$, we simply change U to include everything up to p; ii) if $\beta_{|p}$ is in some encryption $\alpha_j \equiv \{m\}^{\mathsf{n}}_{\mathsf{pk}}$, then we know that n appears in $\vec{w}$, which is not possible since, as β is a $\mathcal{S}_l$-normalized basic term, $\mathsf{n} \in \mathcal{R}_l$ does not appear in $\vec{w}$; iii) if $\beta_{|p}$ is in some decryption $\mathsf{dec}_k \equiv \mathsf{dec}(u_k, \mathsf{sk}_k)$ then, similarly to the previous case, we have sk_k appearing in $\vec{w}$, which contradicts the fact that $\mathsf{sk}_k \in \mathcal{K}_l$ do not appear in $\vec{w}$.

Knowing that $U \equiv U'$ and $\vec{w} \equiv \vec{w}'$, it only remains to show that the encryptions $(\alpha_j)_j$ and $(\alpha'_j)_j$, and the decryptions $(\mathsf{dec}_k)_k$ and $(\mathsf{dec}'_k)_k$ are identical. The former follows from the fact that, for a given encryption randomness $\mathsf{n} \in \mathcal{R}_l$, there exists a unique m such $\{m\}^{\mathsf{n}}_{\_} \in \mathcal{E}_l$; and the latter follows from the fact that there is a unique way to guard a decryption in $\mathcal{D}_l$ (this is not obvious, and relies on $\mathsf{CCA2}$ side-conditions). ■

b) Proofs of $b \sim$ false or true: Using the previous proposition, we can show that for all b, if b is if-free then there is no derivation of $b \sim \mathsf{true}$ or $b \sim \mathsf{false}$ in $\mathcal{A}_\succ$. Such derivations would be problematic since true and false are conditionals of constant size, but b could be of any size (and we are trying to bound all conditionals appearing in a proof). Also, the else branch of a true conditional can contain anything and is, a priori, not bounded by the proof conclusion.

Proposition 6. *Let b an if-free conditional in R-normal form, with $b \not\equiv \mathsf{false}$ (resp. $b \not\equiv \mathsf{true}$). Then there exists no derivation of $b \sim \mathsf{false}$ (resp. $b \sim \mathsf{true}$) in $\mathcal{A}_\succ$.*

Proof. This is shown by induction on the size of the derivation. The full proof is in Appendix VI, and relies on Proposition 5. ■

VII. Bounding the Proof and Decision Procedure

We give here two similar proof cut eliminations, one used on $\overline{\mathsf{BFA}}$ conditionals and the other on $\mathsf{CS}_\square$ conditionals.

a) $\overline{\mathsf{BFA}}$ Rule: We already used this cut elimination to deal with Example 4 for conditionals involved in $\overline{\mathsf{BFA}}$ applications. The cuts we want to eliminate are of the form:

$$\dfrac{a_1, a_2, u_3, v_4, w_5 \sim b_1, c_2, r_3, s_4, t_5}{\underbrace{\begin{array}{ccc} & a_1 & \\ & / \quad \backslash & \\ a_2 & & w_5 \\ / \ \backslash & & \\ u_3 \quad v_4 & & \end{array}}_{\sigma} \sim \underbrace{\begin{array}{ccc} & b_1 & \\ & / \quad \backslash & \\ c_2 & & t_5 \\ / \ \backslash & & \\ r_3 \quad s_4 & & \end{array}}_{\tau}}\ \overline{\mathsf{BFA}}^{(2)} \tag{4}$$

Using Lemma 1, we extract a proof of $a_1, a_2 \sim b_1, c_2$, which, thanks to the ordered strategy, is in $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2})$. From Lemma 2 we get that $b \equiv c$. We then replace (4) by:

$$\dfrac{\dfrac{a_1, u_3, w_5 \sim b_1, r_3, t_5}{\begin{array}{ccc} & a_1 & \\ & / \quad \backslash & \\ u_3 & & w_5 \end{array} \sim \begin{array}{ccc} & b_1 & \\ & / \quad \backslash & \\ r_3 & & t_5 \end{array}}\ \overline{\mathsf{BFA}}}{\sigma \sim \tau}\ R$$

We retrieve a proof in $\mathcal{A}_{\succ}$ by pulling R to the beginning of the proof.

b) $\mathsf{CS}_\Box$ Rule: The $\mathsf{CS}_\Box$ case is more complicated. E.g., take two boxed $\mathsf{CS}_\Box$ conditionals for the same if-free conditional a, and two arbitrary $\mathsf{CS}_\Box$ conditionals on the right side:

$$a_i^\Box \equiv \boxed{a_i^l}\boxed{a_i^r}_{\,a} \quad (i \in \{1,2\}) \qquad b_1^\Box \equiv \boxed{b_1^l}\boxed{b_1^r}_{\,b} \qquad c_2^\Box \equiv \boxed{c_2^l}\boxed{c_2^r}_{\,c}$$

Consider the following cut:

$$\dfrac{\begin{array}{c}\vdots\ (A) \\ a_1^l, a_2^l, u_3 \sim b_1^l, c_2^l, r_3\end{array} \quad \begin{array}{c}\vdots\ (B) \\ a_1^l, a_2^r, v_4 \sim b_1^l, c_2^r, s_4\end{array} \quad \begin{array}{c}\vdots\ (C) \\ a_1^r, w_5 \sim b_1^r, t_5\end{array}}{\underbrace{\begin{array}{ccc} & a_1^\Box & \\ & / \quad \backslash & \\ a_2^\Box & & w_5 \\ / \ \backslash & & \\ u_3 \quad v_4 & & \end{array}}_{\sigma} \sim \underbrace{\begin{array}{ccc} & b_1^\Box & \\ & / \quad \backslash & \\ c_2^\Box & & t_5 \\ / \ \backslash & & \\ r_3 \quad s_4 & & \end{array}}_{\tau}}\ \mathsf{CS}_\Box^{(2)}$$

As we did for $\overline{\mathsf{BFA}}$, we can extract from (A), using Lemma 1, a proof of $a_1^l, a_2^l \sim b_1^l, c_2^l$. But using the ordered strategy, we get that this proof is in $\mathcal{A}_{\mathsf{CS}_\Box}$, which we recall is the fragment:

$$\mathsf{CS}_\Box^* \cdot \{\overline{\mathsf{BFA}}(b, b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2}$$

Therefore we cannot apply Lemma 2. To deal with this cut, we generalize Lemma 2 to the case where the proof is in $\mathcal{A}_{\mathsf{CS}_\Box}$. For this, we need the extra assumptions that $a_1^l, a_2^l, b_1^l, c_2^l$ are if-free, which is a side-condition of $\mathsf{CS}_\Box$.

Lemma 9. *For all a, a', b, c such that their R-normal form is if-free and $a =_R a'$, if $P \vdash^{\mathsf{npf}} a, a' \sim b, c$ then $b =_R c$.*

Proof. (*sketch*) The full proof is given in Appendix VI. It uses Proposition 6 to obtain a proof P' of $a, a' \sim b, c$ without any false and true, and also relies on Proposition 5 and Lemma 2. ■

We now deal with the cut above. Using Lemma 9, we know that $b =_R c$. Since b, c are in R-normal form, $b \equiv c$ and therefore $b_1^\Box =_{R_\Box} b =_{R_\Box} c_2^\Box$ (using well-formedness). Similarly $a_1^\Box =_{R_\Box} a =_{R_\Box} a_2^\Box$. This yields the (cut-free) proof:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\ (A') \\ a_1^l, u_3 \sim b_1^l, r_3\end{array} \qquad \begin{array}{c}\vdots\ (C) \\ a_1^r, w_5 \sim b_1^r, t_5\end{array}}{\begin{array}{ccc} & a_1^\Box & \\ & / \quad \backslash & \\ u_3 & & w_5 \end{array} \sim \begin{array}{ccc} & b_1^\Box & \\ & / \quad \backslash & \\ r_3 & & t_5 \end{array}}\ \mathsf{CS}_\Box}{\sigma \sim \tau}\ R_\Box$$

where (A') is extracted from (A) by Lemma 1. Finally, to get a proof in $\mathcal{A}_{\succ}$, we commute the $R_\Box$ rewriting to the beginning.

A. Decision Procedure

Now, we explain how we obtain a decision procedure for our logic. Because the proofs and definitions are long and technical, we omit most of the details and focus instead on giving a high level sketch of the proof and decision procedure.

a) Spurious Conditionals: A conditional b without $\mathsf{if\_then\_else\_}$ and in R-normal form is said to be *spurious* in t if, when R-normalizing t, the conditional b disappears. Formally, b is spurious in t if $b \notin \mathsf{cond\text{-}st}(t\downarrow_R)$. E.g., the conditional $\mathsf{eq}(\mathsf{n}_0, \mathsf{n}_1)$ is spurious in:

$$\mathsf{if}\ \mathsf{eq}(\mathsf{n}_0, \mathsf{n}_1)\ \mathsf{then}\ g(\mathsf{n})\ \mathsf{else}\ g(\mathsf{n})$$

We say that a basic conditional β, which may not be if-free, is spurious in t if all its leaf terms are spurious in t (i.e. $\mathsf{leave\text{-}st}(\beta\downarrow_R) \cap \mathsf{cond\text{-}st}(t\downarrow_R) = \emptyset$). As we saw in Example 2, we may need to introduce spurious basic conditionals to carry out a proof. Still, we need to bound such terms. To do this, we characterize the basic conditionals that *cannot* be removed: basically, a basic conditional is α-bounded in a proof of $t \sim t'$ if it is not spurious in t or t', or if it is a guard for a decryption appearing in a α-bounded conditional of $t \sim t'$ (indeed, we cannot remove a decryption's guards, as this would not yield a valid $\mathsf{CCA2}$ instance).

We let $\vdash_\alpha^{\mathsf{npf}}$ be the restriction of $\vdash^{\mathsf{npf}}$ to proofs such that all basic conditionals appearing in the derivation are α-bounded. Using the cut eliminations we introduced earlier, plus some additional cut eliminations that are given in Appendix VI, we can show the following completeness result (the full proof is in Appendix VII).

Lemma 10. *$\vdash_\alpha^{\mathsf{npf}}$ is complete with respect to $\vdash^{\mathsf{npf}}$.*

b) Bounding α-bounded Basic Conditionals: Finally, it remains to bound the size of α-bounded basic conditionals. Since basic conditionals can be nested (e.g. a basic conditional can contain decryption guards, which are themselves basic conditionals etc), we need to bound the length of sequences of nested basic conditionals.

Given a sequence of nested basic conditionals $\beta_1 <_{\mathsf{st}} \cdots <_{\mathsf{st}} \beta_n$, (where $u <_{\mathsf{st}} v$ iff $u \not\equiv v$ and $u \in \mathsf{st}(v)$), we show that we can associate to each β_i a "frame term" $\lambda_i \in \mathcal{B}(t, t')$ (where $\mathcal{B}(t, t')$ is a set of terms of bounded size w.r.t. $|t| + |t'|$). Basically, λ_i is obtained from β_i by "flattening" it: we remove all decryption guards, and replace the content of every encryption $\{m\}^{\mathsf{n}}_{\mathsf{pk}}$ by a term $\{\tilde{m}\}^{\mathsf{n}}_{\mathsf{pk}}$, where $\tilde{m}$ is if-free and in $\mathcal{B}(t,t')$. Moreover, we show that, for every $\mathcal{S}_l$-normalized basic terms β,γ and their associated frame terms λ,μ, if $\lambda \equiv \mu$ then $\beta \equiv \gamma$ (this result is similar to Proposition 5).

Since the β_is are all pair-wise distinct (as $<_{\mathsf{st}}$ is strict), and since for every i, the frame term λ_i uniquely characterizes β_i, we know that the λ_is are pair-wise distinct. Using a pigeon-hole argument, this shows that $n \leq |\mathcal{B}(t,t')|$. Then, by induction on the number of nested basic conditionals, we show a triple exponential upper-bound in $|t|+|t'|$ on the size of the basic conditionals appearing in a cut-free proof of $t \sim t'$.

c) Decision Procedure: To conclude, we show that there exists a non-deterministic procedure that, given two terms t and t', non-deterministically guesses a set of α-bounded basic terms that can appear in a proof P of $P \vdash^{\mathsf{npf}}_{\alpha} t \sim t'$ (in triple exponential time in $|t|+|t'|$). Then the procedure guesses the rule applications, and checks that the candidate derivation is a valid proof (in polynomial time in the candidate derivation size). This yields a 3-NExpTime decision procedure that shows the decidability of our problem.

Theorem (Main Result)**.** *The following problem is decidable:*
Input: *A ground formula $\vec{u} \sim \vec{v}$.*
Question: *Is $\mathsf{Ax} \wedge \vec{u} \not\sim \vec{v}$ unsatisfiable?*

VIII. Related Works

In [28], the authors design a set of inference rules to prove CPA and CCA security of asymmetric encryption schemes in the Random Oracle Model. The paper also presents an attack finding algorithm. The authors of [28] do not provide decision algorithm for the designed inference rules. However, they designed proof search heuristics and implemented an automated tool, called ZooCrypt, to synthesize new CCA encryption schemes. For small schemes, this procedure can show CCA security or find an attack in more than 80% of the cases. In 20% of the cases, security remains undecided. Additionally, ZooCrypt automatically generates concrete security bounds.

As seen in the introduction, the problem of showing CPA security can be cast into the BC logic. Take a candidate encryption scheme $x \mapsto t[x]$, where $t[]$ is a context built using, e.g., pairs, a one-way permutation f using public key $\mathsf{pk}(\mathsf{n})$, hash functions and xor. Then this scheme is CPA if the following formula is valid in every computational model satisfying some implementation assumptions (mostly, f is OW-CPA and the hash functions are PRF):

$$t[\pi_1(f(\mathsf{pk}(\mathsf{n})))] \sim t[\pi_2(f(\mathsf{pk}(\mathsf{n})))]$$

This formula has a particular shape, which stems from the limitations on the adversary's interactions: the adversary can only interact with the (candidate) encryption scheme through the CPA or CCA game. There is no complex and arbitrary interactions with the adversary, as it is the case with a security protocol. We don't have such restrictions.

In [29], the authors study proof automation in the UC framework [30]. They design a complete procedure for deciding the existence of a simulator, for ideal and real functionalities using if-then-else, equality, random samplings and xor. Therefore their algorithm cannot be used to analyse functionalities relying on more complex functions (e.g., public key encryption), or stateful functionalities. This restricts the protocols that can be checked. Still, their method is *semantically* complete (while we are complete w.r.t. a fixed set of inference rules): if there exists a simulator, they will find it.

In [31], the authors show the decidability of the problem of the equality of two distributions, for a *specific* equational theory (concatenation, projection and xor). Then, for *arbitrary* equational theories, they design a proof system for proving the equality of two distributions. This second contribution has similarities with our work, but differ in two ways.

First, the proof system of [31] shares some rules with ours, e.g. the R, Dup and FA rules. But it does not allow for reasoning on terms using if_then_else_. E.g., they do not have a counterpart to the CS rule. This is a major difference, as most of the difficulties encountered in the design of our decision procedure result from the if_then_else_ conditionals. Moreover, there are no rules corresponding to cryptographic assumptions, as our CCA2 rules. Because of this and the lack of support for reasoning on branching terms, the analysis of security protocols is out of the scope of [31].

Second, the authors do not provide a decision procedure for their inference rules, but instead rely on heuristics.

IX. Conclusion

We designed a decision procedure for the Bana-Comon indistinguishability logic. This allows to automatically verify that a security protocol satisfies some security property. Our result can be reinterpreted, in the cryptographic game transformation setting, as a cut elimination procedure that guarantees that all intermediate games introduced in a proof are of bounded size w.r.t. the protocol studied.

A lot of work remains to be done. First, our decision procedure is in 3-NExpTime, which is a high complexity. But, as we do not have any lower-bound, there may exist a more efficient decision procedure. Finding such a lower-bound is another interesting direction of research. Then, our completeness result was proven for CCA2 only. We believe it can be extended to more primitives and cryptographic assumptions. For example, signatures and EUF-CMA are very similar to asymmetric encryption and IND-CCA$_2$, and should be easy to handle (even combined with the CCA2 axioms).

Acknowledgment

We thank Hubert Comon for his help and useful comments.

This research has been partially funded by the French National Research Agency (ANR) under the project TECAP (ANR-17-CE39-0004-01).

References

[1] G. Bana and H. Comon-Lundh, "A computationally complete symbolic attacker for equivalence properties," in *2014 ACM Conference on Computer and Communications Security, CCS '14*. ACM, 2014, pp. 609–620.

[2] G. Bana and H. Comon-Lundh, “Towards unconditional soundness: Computationally Complete Symbolic Attacker,” in *Principles of Security and Trust, 2012*, ser. LNCS, vol. 7215. Springer, 2012, pp. 189–208.

[3] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Z. Béguelin, and P. Zimmermann, “Imperfect forward secrecy: How Diffie-Hellman fails in practice,” in *ACM Conference on Computer and Communications Security*. ACM, 2015, pp. 5–17.

[4] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Pironti, and P. Strub, “Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS,” in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2014, pp. 98–113.

[5] B. Blanchet, PROVERIF*: Cryptographic protocols verifier in the formal model*, available at http://proseccco.gforge..inria.fr/personal/bblanchet/proverif/.

[6] S. Meier, B. Schmidt, C. Cremers, and D. Basin, “The TAMARIN prover for the symbolic analysis of security protocols,” in *25th International Conference on Computer Aided Verification, CAV'13*. Springer-Verlag, 2013, pp. 696–701.

[7] V. Cheval, S. Kremer, and I. Rakotonirina, “DEEPSEC: deciding equivalence properties in security protocols theory and practice,” in *2018 IEEE Symposium on Security and Privacy, SP 2018*. IEEE, 2018, pp. 529–546.

[8] G. Barthe, B. Grégoire, S. Heraud, and S. Z. Béguelin, “Computer-aided security proofs for the working cryptographer,” in *Advances in Cryptology - CRYPTO, 2011*, ser. LNCS, vol. 6841. Springer, 2011, pp. 71–90.

[9] B. Blanchet, “A computationally sound mechanized prover for security protocols,” *IEEE Trans. Dependable Sec. Comput.*, vol. 5, no. 4, pp. 193–207, 2008.

[10] H. Comon-Lundh, V. Cortier, and E. Zalinescu, “Deciding security properties for cryptographic protocols. application to key cycles,” *ACM Trans. Comput. Log.*, vol. 11, no. 2, pp. 9:1–9:42, 2010.

[11] E. D'Osualdo, L. Ong, and A. Tiu, “Deciding secrecy of security protocols for an unbounded number of sessions: The case of depth-bounded processes,” in *CSF*. IEEE Computer Society, 2017, pp. 464–480.

[12] A. Finkel and P. Schnoebelen, “Well-structured transition systems everywhere!” *Theor. Comput. Sci.*, vol. 256, no. 1-2, pp. 63–92, 2001.

[13] R. Chrétien, V. Cortier, and S. Delaune, “Decidability of trace equivalence for protocols with nonces,” in *CSF*. IEEE Computer Society, 2015, pp. 170–184.

[14] V. Cheval, H. Comon-Lundh, and S. Delaune, “A procedure for deciding symbolic equivalence between sets of constraint systems,” *Inf. Comput.*, vol. 255, pp. 94–125, 2017.

[15] H. Comon-Lundh, V. Cortier, and G. Scerri, “Tractable inference systems: An extension with a deducibility predicate,” in *CADE*, ser. LNCS, vol. 7898. Springer, 2013, pp. 91–108.

[16] M. Abadi and P. Rogaway, “Reconciling two views of cryptography (the computational soundness of formal encryption),” *J. Cryptology*, vol. 15, no. 2, pp. 103–127, 2002.

[17] M. Backes, A. Malik, and D. Unruh, “Computational soundness without protocol restrictions,” in *ACM Conference on Computer and Communications Security*. ACM, 2012, pp. 699–711.

[18] M. Backes, E. Mohammadi, and T. Ruffing, “Computational soundness results for proverif - bridging the gap from trace properties to uniformity,” in *POST*, ser. Lecture Notes in Computer Science, vol. 8414. Springer, 2014, pp. 42–62.

[19] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, “Relations among notions of security for public-key encryption schemes,” in *CRYPTO*, ser. LNCS, vol. 1462. Springer, 1998, pp. 26–45.

[20] V. Shoup, “Sequences of games: a tool for taming complexity in security proofs,” *IACR Cryptology ePrint Archive*, vol. 2004, p. 332, 2004, https://eprint.iacr.org/2004/332.

[21] M. Bellare and P. Rogaway, “The security of triple encryption and a framework for code-based game-playing proofs,” in *EUROCRYPT*, ser. LNCS, vol. 4004. Springer, 2006, pp. 409–426.

[22] H. Comon and A. Koutsos, “Formal computational unlinkability proofs of RFID protocols,” in *30th Computer Security Foundations Symposium, 2017*. IEEE Computer Society, 2017, pp. 100–114.

[23] G. Scerri and R. Stanley-Oakes, “Analysis of key wrapping APIs: Generic policies, computational security,” in *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*. IEEE Computer Society, 2016, pp. 281–295.

[24] G. Bana, R. Chadha, and A. K. Eeralla, “Formal analysis of vote privacy using computationally complete symbolic attacker,” in *ESORICS (2)*, ser. LNCS, vol. 11099. Springer, 2018, pp. 350–372.

[25] C. Chang and R. C. T. Lee, *Symbolic logic and mechanical theorem proving*, ser. Computer science classics. Academic Press, 1973.

[26] G. Bana and R. Chadha, “Verification methods for the computationally complete symbolic attacker based on indistinguishability,” *IACR Cryptology ePrint Archive*, vol. 2016, p. 69, 2016. [Online]. Available: http://eprint.iacr.org/2016/069

[27] G. Lowe, “An attack on the Needham-Schroeder public-key authentication protocol,” *Inf. Process. Lett.*, vol. 56, no. 3, pp. 131–133, 1995.

[28] G. Barthe, J. M. Crespo, B. Grégoire, C. Kunz, Y. Lakhnech, B. Schmidt, and S. Z. Béguelin, “Fully automated analysis of padding-based encryption in the computational model,” in *ACM Conference on Computer and Communications Security*. ACM, 2013, pp. 1247–1260.

[29] C. S. Jutla and A. Roy, “Decision procedures for simulatability,” in *ESORICS*, ser. LNCS, vol. 7459. Springer, 2012, pp. 573–590.

[30] R. Canetti, “Universally composable security: A new paradigm for cryptographic protocols,” in *FOCS*. IEEE Computer Society, 2001, pp. 136–145.

[31] G. Barthe, M. Daubignard, B. M. Kapron, Y. Lakhnech, and V. Laporte, “On the equality of probabilistic terms,” in *Logic for Programming, Artificial Intelligence, and Reasoning - 16th International Conference, LPAR-16, Dakar, Senegal, April 25-May 1, 2010, Revised Selected Papers*, ser. LNCS, E. M. Clarke and A. Voronkov, Eds., vol. 6355. Springer, 2010, pp. 46–63.

[32] M. Bellare, A. Boldyreva, and S. Micali, “Public-key encryption in a multi-user setting: Security proofs and improvements,” in *EUROCRYPT*, ser. LNCS, vol. 1807. Springer, 2000, pp. 259–274.

CONTENTS

APPENDIX I
THE TERM REWRITING SYSTEM R

A. Notations

Definition 7. *A* position *is a word in $\mathbb{N}^*$. The value of a term t at a position p, denoted by $(t)_{|p}$, is the partial function defined inductively as follows:*

$$\begin{array}{lcl} (t)_{|\epsilon} & = & t \\ (f(u_0,\dots,u_{n-1}))_{|i.p} & = & \begin{cases} (u_i)_{|p} & \textit{if } i<n \\ \textit{undefined} & \textit{otherwise} \end{cases} \end{array}$$

We say that a position in valid *is t if $(t)_{|p}$ is defined. The set of positions of a term is the set of positions which are valid in t.*

Definition 8. *A context $D[]_{\vec{x}}$ (sometimes written D when there is no confusion) is a term in $\mathcal{T}(\mathcal{F},\mathcal{N},\{[]_y \mid y\in\vec{x}\})$ where $\vec{x}$ are distinct special variables called holes.*

For all contexts $D[]_{\vec{x}}, C_0,\dots,C_{n-1}$ with $|\vec{x}|=n$, we let $D[(C_i)_{i<n}]$ be the context $D[]_{\vec{x}}$ in which we substitute, for all $0\le i<n$, all occurrences of the hole $[]_{x_i}$ by C_i.

A one-holed context is a context with one hole (in which case we write $D[]$ where $[]$ is the only variable).

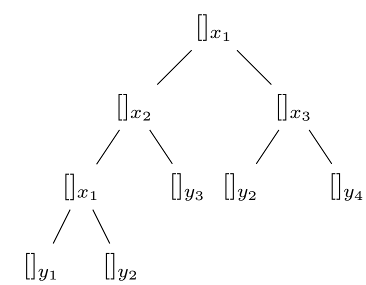
Often, we want to distinguish between holes that contain "internal" conditionals, and holes that contain terms appearing at the leaves. To do this we introduce the notion of if-context:

Definition 9. *For all distinct variables $\vec{x},\vec{y}$, an if-context $D[]_{\vec{x}\diamond\vec{y}}$ is a context in $\mathcal{T}(\textsf{if \_ then \_ else \_},\{[]_z \mid z\in\vec{x}\cup\vec{y}\})$ such that for all position p, $D_{|p}\equiv \textsf{if } b \textsf{ then } u \textsf{ else } v$ implies:*

- $b\in\{[]_z \mid z\in\vec{x}\}$
- $u,v\notin\{[]_z \mid z\in\vec{x}\}$

Example 6. Let $\vec{x}=x_1,x_2,x_3$ and $\vec{y}=y_1,y_2,y_3,y_4$, we give below two representations of the same if-context $D[]_{\vec{x}\diamond\vec{y}}$ (the term on the left, and the labelled tree on the right):

$$\begin{array}{l}\textsf{if } []_{x_1} \textsf{ then } \left(\begin{array}{l}\textsf{if } []_{x_2} \textsf{ then if } []_{x_1} \textsf{ then } []_{y_1} \textsf{ else } []_{y_2} \\ \qquad\quad \textsf{else } []_{y_3}\end{array}\right) \\ \qquad \textsf{else } (\textsf{if } []_{x_3} \textsf{ then } []_{y_2} \textsf{ else } []_{y_4})\end{array}$$



Definition 10. *For every term t, we let $\mathsf{st}(t)$ be the set of subterms of t.*

If $t\equiv C[\vec{b}\diamond\vec{u}]$ where $\vec{b},\vec{u}$ are if-free terms then we let $\mathsf{cond\text{-}st}(t)$ be the set of conditionals $\vec{b}$, and $\mathsf{leave\text{-}st}(t)$ be the set of terms $\vec{u}$.

Definition 11. *A* directed path *${}^{\delta}\vec{\rho}$ is a sequence $(b_0,d_0),\dots,(b_n,d_n)$ where $b_0,\dots,b_n$ are conditionals and $d_0,\dots,d_n$ (the directions) are in $\{\textsf{then},\textsf{else}\}$.*

Two directed paths ${}^{\delta}\vec{\rho}$ and ${}^{\delta}\vec{\rho}'$ are said to have the same directions if:

- *they have the same length.*
- *the sequences of* directions *$d_0,\dots,d_n$ and $d'_0,\dots,d'_n$ extracted from, respectively, ${}^{\delta}\vec{\rho}$ and ${}^{\delta}\vec{\rho}'$, are equal.*

Given a directed path ${}^{\delta}\vec{\rho}$, we let $\vec{\rho}$ stands for the sequence of conditionals *extracted from ${}^{\delta}\vec{\rho}$.*

B. Convergence of R

a) Lexicographic Path Ordering:: Let $\succ_f$ be a total precedence over function symbols. The lexicographic path ordering associated with $\succ_f$ is the pre-order defined by:

$$s=f(s_1,\dots,s_n)\succ t=g(t_1,\dots,t_m) \text{ iff } \begin{cases} \exists i\in[\![1,n]\!] \text{ s.t. } s_i\succeq t \\ f=g\wedge\forall j\in[\![1,m]\!], s\succ t_j \wedge s_1,\dots,s_n \succ_{lex} t_1,\dots,t_n \\ f\succ_f g\wedge \forall j\in[\![1,m]\!], s\succ t_j \end{cases}$$

$$\rightarrow_{R'_2} \quad \begin{cases} f(\vec{u}, \mathsf{if}_b(x\,,\,y), \vec{v}) \rightarrow \mathsf{if}_b(f(\vec{u}, x, \vec{v})\,,\, f(\vec{u}, y, \vec{v})) & (f \in \mathcal{F}_s) \\ \mathsf{if}_{(\mathsf{if}_b(a\,,\,c))}(x\,,\,y) \rightarrow \mathsf{if}_b((\mathsf{if}_a(x\,,\,y))\,,\,(\mathsf{if}_c(x\,,\,y))) \end{cases}$$

$$\rightarrow_{R'_3} \quad \begin{cases} \mathsf{if}_{\mathsf{true}}(x\,,\,y) \rightarrow x \\ \mathsf{if}_{\mathsf{false}}(x\,,\,y) \rightarrow y \\ \mathsf{if}_b(x\,,\,x) \rightarrow x \\ \mathsf{if}_b((\mathsf{if}_b(x\,,\,y))\,,\,z) \rightarrow \mathsf{if}_b(x\,,\,z) \\ \mathsf{if}_b(x\,,\,(\mathsf{if}_b(y\,,\,z))) \rightarrow \mathsf{if}_b(x\,,\,z) \end{cases}$$

$$\rightarrow_{R^0_4} \quad \begin{cases} \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ z \rightarrow & \\ \quad \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ y\ \mathsf{else}\ z) & (b \succ a,\ a,b \text{ not if-free or not in } R\text{-normal form}) \\ \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ y\ \mathsf{else}\ z) \rightarrow & \\ \quad \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z) & (b \succ a,\ a,b \text{ not if-free or not in } R\text{-normal form}) \end{cases}$$

$$\rightarrow_{R^1_4} \quad \begin{cases} \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}_a(x\,,\,y))\ \mathsf{else}\ z \rightarrow & \\ \quad \mathsf{if}_a((\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z)\,,\,(\mathsf{if}\ b\ \mathsf{then}\ y\ \mathsf{else}\ z)) & (b \text{ not if-free or not in } R\text{-normal form}) \\ \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ (\mathsf{if}_a(y\,,\,z)) \rightarrow & \\ \quad \mathsf{if}_a((\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ y)\,,\,(\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z)) & (b \text{ not if-free or not in } R\text{-normal form}) \end{cases}$$

$$\rightarrow_{R^2_4} \quad \begin{cases} \mathsf{if}_b((\mathsf{if}_a(x\,,\,y))\,,\,z) \rightarrow \mathsf{if}_a((\mathsf{if}_b(x\,,\,z))\,,\,(\mathsf{if}_b(y\,,\,z))) & (b \succ_u a) \\ \mathsf{if}_b(x\,,\,(\mathsf{if}_a(y\,,\,z))) \rightarrow \mathsf{if}_a((\mathsf{if}_b(x\,,\,y))\,,\,(\mathsf{if}_b(x\,,\,z))) & (b \succ_u a) \end{cases}$$

$$\rightarrow_{R^i} \quad \{\ \mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v \rightarrow \mathsf{if}_b(u\,,\,v) \qquad (b \text{ if-free and in } R\text{-normal form})$$

Fig. 5. The Relations $\rightarrow_{R'_2}, \rightarrow_{R'_3}, \rightarrow_{R^0_4}, \rightarrow_{R^1_4}, \rightarrow_{R^2_4}$ and $\rightarrow_{R^i}$ used for termination

Let $\succ_f$ be a total precedence on $\mathcal{F}, \mathcal{N}$ such that if_then_else_ is the smallest element (elements of $\mathcal{N}$ are treated as function symbols of arity zero). Let $\succ$ be the lexicographic path ordering on $\mathcal{T}(\mathcal{F}, \mathcal{N})$ using precedence $\succ_f$. Let $\succ_u$ be a user-chosen total order on if-free conditionals in R-normal form. We define the total ordering $\succ_c$ on conditionals as follows:

$$b \succ_c a = \begin{cases} b \succ_u a & \text{if } a \text{ and } b \text{ are if-free and } R\text{-irreducible} \\ b \succ a & \text{if } a \text{ and } b \text{ are not if-free or not } R\text{-irreducible} \\ \mathsf{true} & \text{if } a \text{ is if-free and } R\text{-irreducible, and } b \text{ is not} \\ \mathsf{false} & \text{if } b \text{ is if-free and } R\text{-irreducible, and } a \text{ is not} \end{cases}$$

We then order $\rightarrow_{R_4^{\succ_u}}$ as follows:

$$\begin{array}{ll} \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ z \rightarrow \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ y\ \mathsf{else}\ z) & \text{when } b \succ_c a \\ \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ y\ \mathsf{else}\ z) \rightarrow \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z) & \text{when } b \succ_c a \end{array}$$

Let $\rightarrow_{R^{\succ_u}} = \rightarrow_{R_1} \cup \rightarrow_{R_2} \cup \rightarrow_{R_3} \cup \rightarrow_{R_4^{\succ_u}}$. The term rewriting system $\rightarrow_{R^{\succ_u}}$ is an orientation of the rules given in Fig. 1. When we do not care about the choice of total ordering on if-free conditionals in R-normal form $\succ_u$, we write $\rightarrow_R$.

Theorem 2. *For all $\succ_u$, the term rewriting system $\rightarrow_{R^{\succ_u}}$ is convergent on ground terms.*

Proof. We show that $\rightarrow_{R^{\succ_u}}$ is locally confluent and terminating, and conclude by Newman's lemma.

b) Local Confluence: We show that all critical pairs are joinable. Normally, we would rely on some automated checker for local confluence. Unfortunately, as we rely on a side-condition to orient R_4 (using a LPO), writing down the rules in a tool is not straightforward. By consequence we believe it is simpler to manually check that every critical pair is joinable. We give below the most interesting critical pairs, and show how we join them. For every critical pair, we underline the starting term.

- **Critical Pairs** $R_1/(R_1 \cup R_2 \cup R_3 \cup R_4)$**:** we only show the critical pairs involving $\pi_1(\_)$ (the critical pairs with $\pi_2(\_)$ are similar), and for $\mathsf{eq}(\_,\_)$. The critical pairs involving $\mathsf{dec}(\_,\_)$ are similar to the critical pairs involving $\pi_1(\_)$.

$$\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v \leftarrow^2 \mathsf{if}\ b\ \mathsf{then}\ \pi_1(\langle u\,,\,w\rangle)\ \mathsf{else}\ \pi_1(\langle v\,,\,w\rangle) \leftarrow \underline{\pi_1(\langle \mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v\,,\,w\rangle)} \rightarrow \mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v$$

$$w \leftarrow \mathsf{if}\ b\ \mathsf{then}\ w\ \mathsf{else}\ w \leftarrow^2 \mathsf{if}\ b\ \mathsf{then}\ \pi_1(\langle w\,,\,u\rangle)\ \mathsf{else}\ \pi_2(\langle w\,,\,v\rangle) \leftarrow \underline{\pi_1(\langle w\,,\,\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v\rangle)} \rightarrow w$$

$$
\begin{array}{ll}
& \text{true} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \leftarrow \\
& \underline{\text{eq}(\text{if } b \text{ then } u \text{ else } v, \text{if } b \text{ then } u \text{ else } v)} \\
\rightarrow & \text{if } b \text{ then eq}(u, \text{if } b \text{ then } u \text{ else } v) \text{ else eq}(v, \text{if } b \text{ then } u \text{ else } v) \\
\rightarrow & \text{if } b \text{ then (if } b \text{ then eq}(u,u) \text{ else eq}(u,v)) \text{ else eq}(v, \text{if } b \text{ then } u \text{ else } v) \\
\rightarrow & \text{if } b \text{ then eq}(u,u) \text{ else eq}(v, \text{if } b \text{ then } u \text{ else } v) \\
\rightarrow & \text{if } b \text{ then true else eq}(v, \text{if } b \text{ then } u \text{ else } v) \\
\rightarrow^* & \text{if } b \text{ then true else true} \\
\rightarrow & \text{true}
\end{array}
$$

- **Critical Pairs R_2/R_2:** we assume that $b \succ_c c$. The other possible orderings are handled in the same fashion.

$$
\begin{array}{lrl}
& \text{if } c \text{ then (if } b \text{ then } f(u,s) \text{ else } f(v,s)) \text{ else (if } b \text{ then } f(u,t) \text{ else } f(v,t)) & \leftarrow^2 \\
& \text{if } c \text{ then } f(\text{if } b \text{ then } u \text{ else } v, s) \text{ else } f(\text{if } b \text{ then } u \text{ else } v, t) & \leftarrow \\
& \underline{f(\text{if } b \text{ then } u \text{ else } v, \text{if } c \text{ then } s \text{ else } t)} & \\
\rightarrow & \text{if } b \text{ then } f(u, \text{if } c \text{ then } s \text{ else } t) \text{ else } f(v, \text{if } c \text{ then } s \text{ else } t) & \\
\rightarrow^2 & \text{if } b \text{ then (if } c \text{ then } f(u,s) \text{ else } f(u,t)) \text{ else (if } c \text{ then } f(v,s) \text{ else } f(v,t)) & \\
\rightarrow^* & \text{if } c \text{ then (if } b \text{ then } f(u,s) \text{ else } f(v,s)) \text{ else (if } b \text{ then } f(u,t) \text{ else } f(v,t)) &
\end{array}
$$

- **Critical Pairs R_2/R_3:**

$$f(u,w) \leftarrow \underline{f(\text{if true then } u \text{ else } v, w)} \rightarrow \text{if true then } f(u,w) \text{ else } f(v,w) \rightarrow f(u,w)$$

$$f(u,v) \leftarrow \underline{f(\text{if } b \text{ then } u \text{ else } u, v)} \rightarrow \text{if } b \text{ then } f(u,v) \text{ else } f(u,v) \rightarrow f(u,v)$$

$$
\begin{array}{lll}
& \text{if } b \text{ then } f(u,s) \text{ else } f(w,s) & \leftarrow \\
& f(\text{if } b \text{ then } u \text{ else } w, s) & \leftarrow \\
& \underline{f(\text{if } b \text{ then (if } b \text{ then } u \text{ else } v) \text{ else } w, s)} & \\
\rightarrow & \text{if } b \text{ then } f(\text{if } b \text{ then } u \text{ else } v, s) \text{ else } f(w,s) & \\
\rightarrow & \text{if } b \text{ then (if } b \text{ then } f(u,s) \text{ else } f(v,s)) \text{ else } f(w,s) & \\
\rightarrow & \text{if } b \text{ then } f(u,s) \text{ else } f(w,s) &
\end{array}
$$

- **Critical Pairs R_2/R_4:** we assume that $a \succ_c b \succ_c c \succ_c d$. The other possible orderings are handled in the same fashion.

$$
\begin{array}{lll}
& \text{if } d \text{ then (if } b \text{ then (if } a \text{ then } u \text{ else } v) \text{ else } w) \text{ else (if } c \text{ then (if } a \text{ then } u \text{ else } v) \text{ else } w) & \leftarrow^* \\
& \text{if } a \text{ then if } d \text{ then (if } b \text{ then } u \text{ else } w) \text{ else (if } c \text{ then } u \text{ else } w) & \leftarrow^2 \\
& \quad \text{else if } d \text{ then (if } b \text{ then } v \text{ else } w) \text{ else (if } c \text{ then } v \text{ else } w) & \\
& \text{if } a \text{ then (if (if } d \text{ then } b \text{ else } c) \text{ then } u \text{ else } w) \text{ else (if (if } d \text{ then } b \text{ else } c) \text{ then } v \text{ else } w) & \leftarrow \\
& \underline{\text{if (if } d \text{ then } b \text{ else } c) \text{ then (if } a \text{ then } u \text{ else } v) \text{ else } w} & \\
\rightarrow & \text{if } d \text{ then (if } b \text{ then (if } a \text{ then } u \text{ else } v) \text{ else } w) \text{ else (if } c \text{ then (if } a \text{ then } u \text{ else } v) \text{ else } w) &
\end{array}
$$

- **Critical Pairs R_3/R_3:**

$$u \leftarrow \underline{\text{if true then } u \text{ else } u} \rightarrow u$$

$$u \leftarrow \text{if true then } u \text{ else } v \leftarrow \underline{\text{if true then (if true then } u \text{ else } v) \text{ else } w} \rightarrow \text{if true then } u \text{ else } w \rightarrow u$$

$$
\begin{array}{l}
\text{if } b \text{ then } u \text{ else } v \leftarrow \underline{\text{if } b \text{ then (if } b \text{ then } u \text{ else } v) \text{ else (if } b \text{ then } u \text{ else } v)} \rightarrow \text{if } b \text{ then } u \text{ else (if } b \text{ then } u \text{ else } v) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \rightarrow \text{if } b \text{ then } u \text{ else } v
\end{array}
$$

- **Critical Pairs R_3/R_4:**

$$\begin{array}{ll}
& \mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ v \qquad\qquad \leftarrow \\
& \underline{\mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ v)\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ v)} \\
\to & \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ v))\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ v\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ v)) \\
\to^2 & \mathsf{if}\ a\ \ \mathsf{then}\ \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ u)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v) \\
& \qquad \mathsf{else}\ \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ v\ \mathsf{else}\ u)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ v\ \mathsf{else}\ v) \\
\to^2 & \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ u)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ v\ \mathsf{else}\ v) \\
\to^2 & \mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ v
\end{array}$$

- **Critical Pairs R_4/R_4:** we assume that $a \succ_c b \succ_c c$. The other possible orderings are handled in the same fashion.

$$\begin{array}{ll}
& \mathsf{if}\ c\ \ \mathsf{then}\ \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ s)\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ v\ \mathsf{else}\ s) \qquad \leftarrow^2 \\
& \qquad \mathsf{else}\ \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ t)\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ v\ \mathsf{else}\ t) \\
& \mathsf{if}\ c\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v)\ \mathsf{else}\ s)\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ v\ \mathsf{else}\ u)\ \mathsf{else}\ t) \quad \leftarrow \\
& \underline{\mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v)\ \mathsf{else}\ (\mathsf{if}\ c\ \mathsf{then}\ s\ \mathsf{else}\ t)} \\
\to & \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ (\mathsf{if}\ c\ \mathsf{then}\ s\ \mathsf{else}\ t))\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ v\ \mathsf{else}\ (\mathsf{if}\ c\ \mathsf{then}\ s\ \mathsf{else}\ t)) \\
\to^2 & \mathsf{if}\ b\ \ \mathsf{then}\ \mathsf{if}\ c\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ s)\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ t) \\
& \qquad \mathsf{else}\ \mathsf{if}\ c\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ v\ \mathsf{else}\ s)\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ v\ \mathsf{else}\ t) \\
\to^* & \mathsf{if}\ c\ \ \mathsf{then}\ \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ s)\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ v\ \mathsf{else}\ s) \\
& \qquad \mathsf{else}\ \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ t)\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ v\ \mathsf{else}\ t)
\end{array}$$

c) Termination: To prove termination we add to $\mathcal{F}$ a symbol $\mathsf{if}_b(\,,\,)$ for all if-free conditional b in R-normal form. We also extend the precedence $\succ_f$ on function symbol by having the function symbols $\{\mathsf{if}_b(\,,\,)\}$ be smaller than all the other function symbols, and $\mathsf{if}_b(\,,\,) \succ_f \mathsf{if}_a(\,,\,)$ if and only if $b \succ_u a$. Observe that the extended precedence is still a total order.

We then consider the term rewriting system $\to_{R'}$, defined by removing $\to_{R_4}$ from $\to_R$ and adding all the rules in Fig. 5:

$$\to_{R'} = \to_{R_1} \cup \to_{R_2} \cup \to_{R'_2} \cup \to_{R_3} \cup \to_{R'_3} \cup \to_{R_4^0} \cup \to_{R_4^1} \cup \to_{R_4^2} \cup \to_{R^i}$$

One can easily (but tediously) check that $\succ$ is compatible with $\to_{R'}$: the only non-trivial cases are the cases in $\to_{R_2}$ (the first rule is decreasing because $f \succ_f$ $\mathsf{if\_then\_else\_}$, the second rule using the lexicographic order), in $\to_{R'_2}$ (same arguments than for R_2) and the cases in $\to_{R_4^0}, \to_{R_4^1}, \to_{R_4^2}$ (where we use the side conditions $b \succ a$, $b \succ_u a \ldots$).

Since $\succ$ is a lexicographic path ordering we know that it is total and well-founded on ground-terms. Therefore $\to_{R'}$ is a terminating TRS on ground terms.

To conclude, one just has to observe that for every ground terms u, v and integer n, if $u \to_R^{(n)} v$ then there exist u', v' such that $u \to_{R^i}^{!} u'$, $v \to_{R^i}^{!} v'$ and $u' \to_{R'}^{(\geq n)} v'$. That is, we have the following diagram (black edges stand for universal quantifications, red edges for existentials):

$$\begin{tikzcd}
u \arrow[r, "*"', "R"] \arrow[d, "R^i\,!"'] & v \arrow[d, "R^i\,!"'] \\
u' \arrow[r, "*"', "R'"] & v'
\end{tikzcd}$$

This result can be proved by induction on n. Since $\to_{R'}$ is terminating on ground terms, and since any infinite sequence for $\to_R$ can be translated into an infinite sequence for $\to_{R'}$, it follows easily that $\to_R$ is terminating on ground terms. ∎

C. Property of R

Proposition 7. *Let $\succ_u$ and $\succ'_u$ be two total orderings on if-free conditionals in R-normal form. Then for every ground term t we have:*

$$\mathsf{leave\text{-}st}(t\downarrow_{R^{\succ_u}}) = \mathsf{leave\text{-}st}(t\downarrow_{R^{\succ'_u}}) \qquad \textit{and} \qquad \mathsf{cond\text{-}st}(t\downarrow_{R^{\succ_u}}) = \mathsf{cond\text{-}st}(t\downarrow_{R^{\succ'_u}})$$

Proof. Let $\vec{b} = \mathsf{leave\text{-}st}(t\downarrow_{R^{\succ_u}})$ and $\vec{u} = \mathsf{cond\text{-}st}(t\downarrow_{R^{\succ_u}})$, we know that there exists a if-context C such that $t\downarrow_{R^{\succ_u}} \equiv C[\vec{b} \diamond \vec{u}]$. It is then easy to show by induction on the length of the reduction that for all n, if $C[\vec{b} \diamond \vec{u}] \to_{R^{\succ'_u}}^{(n)} v$ then there exists an if-context C' such that $v \equiv C'[\vec{b} \diamond \vec{u}]$. The wanted result follows immediately. ∎

APPENDIX II
THE CCA2 AXIOMS

We define and prove correct a recursive set of axioms for an IND-CCA$_2$ encryption scheme. For the sack of simplicity, we first ignore all length constraints. We explain how length constraints are added and handled to the logic in Section II-B.

a) Multi-Users IND-CCA$_2$ *Game:* Consider the following multi-users IND-CCA$_2$ game: the adversary receives n public-keys. For each key pk_i, he has access to a left-right oracle $\mathcal{O}_{\mathsf{LR}}(\mathsf{pk}_i, b)$ that takes two messages m_0, m_1 as input and returns $\{m_b\}_{\mathsf{pk}_i}^{\mathsf{n}_r}$, where b is an internal random bit uniformly drawn at the beginning by the challenger (the same b is used for all left-right oracles) and n_r is a fresh nonce. Moreover, for all key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$, the adversary has access to an sk_i decryption oracle $\mathcal{O}_{\mathsf{dec}}(\mathsf{sk}_i)$, but cannot call $\mathcal{O}_{\mathsf{dec}}(\mathsf{sk}_i)$ on a cipher-text returned by $\mathcal{O}_{\mathsf{LR}}(\mathsf{pk}_i, b)$ (to do this, the two oracles use a shared memory where all encryption requests are logged). The advantage of an adversary against this game and the multi-user IND-CCA$_2$ security are defined as usual.

It is known that if an encryption scheme is IND-CCA$_2$ then it is also multi-users IND-CCA$_2$ (see [32]). Therefore, we allow multiple key pairs to appear in the CCA2 axioms, and multiple encryptions over different terms using the same public key (each encryption corresponds to one call to a left-right oracle).

b) Decryption Guards: If we want the following to hold in any computational model

$$\mathsf{dec}\big(\underbrace{t\big[\{u_1\}_{\mathsf{pk}}^{\mathsf{n}_1}, \ldots, \{u_n\}_{\mathsf{pk}}^{\mathsf{n}_n}\big]}_{s}, \mathsf{sk}\big) \sim \mathsf{dec}\big(\underbrace{t\big[\{v_1\}_{\mathsf{pk}}^{\mathsf{n}_1}, \ldots, \{v_n\}_{\mathsf{pk}}^{\mathsf{n}_n}\big]}_{s'}, \mathsf{sk}\big)$$

then we need to make sure that s is different from all $\{u_i\}_{\mathsf{pk}}^{\mathsf{n}_i}$ and that s' is different from all $\{v_i\}_{\mathsf{pk}}^{\mathsf{n}_i}$. This is done by introducing all the unwanted equalities in if_then_else_ tests and making sure that we are in the else branch of all these tests, so as to have a "safe call" to the decryption oracle. Moreover, the adversary is allowed to use values obtained from previous calls to the decryption oracle in future calls.

To do this, we use the following function:

Definition 12. *We define the function else* by induction:*

$$\begin{aligned} &\textit{else}^*(\emptyset, x) \equiv x \\ &\textit{else}^*\,((\textit{eq}(a,b)) :: \Gamma, x) \equiv \textit{if } \textit{eq}(a,b) \textit{ then } \mathbf{0}(x) \textit{ else } \textit{else}^*(\Gamma, x) \end{aligned}$$

Example 7. Let $u \equiv t[\{v_1\}_{\mathsf{pk}}^{\mathsf{n}_r^1}, \{v_2\}_{\mathsf{pk}}^{\mathsf{n}_r^2}]$. Then:

$$\begin{aligned} &\mathsf{else}^*\big(\big(\mathsf{eq}(u, \{v_1\}_{\mathsf{pk}}^{\mathsf{n}_r^1}), \mathsf{eq}(u, \{v_2\}_{\mathsf{pk}}^{\mathsf{n}_r^2})\big), \mathsf{dec}(u, \mathsf{sk})\big) \equiv \\ &\qquad \mathsf{if}\ \mathsf{eq}(u, \{v_1\}_{\mathsf{pk}}^{\mathsf{n}_r^1})\ \mathsf{then}\ \mathbf{0}(\mathsf{dec}(u, \mathsf{sk}))\ \mathsf{else}\ \mathsf{if}\ \mathsf{eq}(u, \{v_2\}_{\mathsf{pk}}^{\mathsf{n}_r^2})\ \mathsf{then}\ \mathbf{0}(\mathsf{dec}(u, \mathsf{sk}))\ \mathsf{else}\ \mathsf{dec}(u, \mathsf{sk}) \end{aligned}$$

Morally, this represents a safe call to the decryption oracle.

c) Definition of CCA2*:* We use the following notations: for any finite set $\mathcal{K}$ of valid private keys, $\mathcal{K} \sqsubseteq_d \vec{u}$ holds if for all $\mathsf{sk} \in \mathcal{K}$, the secret key sk appears only in decryption position in $\vec{u}$; $\mathsf{nodec}(\mathcal{K}, \vec{u})$ denotes that for all $\mathsf{sk}(\mathsf{n}) \in \mathcal{K}$, the only occurrences of n are in subterms $\mathsf{pk}(\mathsf{n})$; $\mathsf{hidden\text{-}rand}(\vec{r}; \vec{u})$ denotes that for all $\mathsf{n}_r \in \vec{r}$, n_r appears only in encryption randomness position and is not used with two distinct plaintexts.

We are now going to define by induction the CCA2 axiom. In order to do this we define by induction a binary relation $R^{\mathcal{K}}_{\mathsf{CCA2}^a}$ on CCA2 executions, where $\mathcal{K}$ is the finite set of private keys used in the terms (corresponding to the public keys sent by the challenger).

Definition 13. *Let $\mathcal{K}$ be a set of private keys. $(\phi, \mathcal{X}_{enc}, \mathcal{X}_{dec}, \sigma_{rand}, \theta_{enc}, \lambda_{dec})$ is a* CCA2 *execution if:*

- *ϕ is a vector of ground terms in $\mathcal{T}(\mathcal{F}, \mathcal{N})$.*
- *$\mathcal{X}_{enc}$ and $\mathcal{X}_{dec}$ are two disjoint sets of variables used as handles for, respectively, encryptions and decryptions.*
- *σ_{rand} is a substitution from $\mathcal{X}_{enc}$ to $\mathcal{N}$.*
- *θ_{enc} and λ_{dec} are substitutions from, respectively, $\mathcal{X}_{enc}$ and $\mathcal{X}_{dec}$, to ground terms in $\mathcal{T}(\mathcal{F}, \mathcal{N})$.*

σ_{rand}, θ_{enc} and λ_{dec} co-domains are the sets of, respectively, encryption randomness, encryption oracle calls and decryption oracle calls in ϕ. Intuitively, we have:

$$(\phi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}}) R^{\mathcal{K}}_{\mathsf{CCA2}^a} (\psi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}})$$

when we can build ϕ and ψ using function symbols, matching encryption oracle calls and matching decryption oracle calls.

Definition 14. *Let $\mathcal{K}$ be a finite set of private keys. We define the binary relation $R^{\mathcal{K}}_{\mathsf{CCA2}^a}$ by induction:*

1) ***No Call to the Oracles:*** *if $\mathcal{K} \sqsubseteq_d \phi$ then $(\phi, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset) R^{\mathcal{K}}_{\mathsf{CCA2}^a} (\phi, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$ for every sequence ϕ of ground terms in $\mathcal{T}(\mathcal{F}, \mathcal{N})$ such that $\mathsf{nodec}(\mathcal{K}; \phi)$.*
2) ***Encryption Case:*** *Let x a fresh variable that does not appear in $\mathcal{X}_{\mathsf{enc}} \cup \mathcal{X}_{\mathsf{dec}}$, sk be a secret key in $\mathcal{K}$ and pk the corresponding public key. Then:*

$$\big((\phi, \{u\}^{n_r}_{\mathsf{pk}}), \mathcal{X}_{\mathsf{enc}} \cup \{x\}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}} \cup \{x \mapsto \mathsf{n}_r\}, \theta_{\mathsf{enc}} \cup \{x \mapsto \{u\}^{n_r}_{\mathsf{pk}}\}, \lambda_{\mathsf{dec}}\big)$$
$$R^{\mathcal{K}}_{\mathsf{CCA2}^a} \big((\psi, \{v\}^{n'_r}_{\mathsf{pk}}), \mathcal{X}_{\mathsf{enc}} \cup \{x\}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}} \cup \{x \mapsto \mathsf{n}'_r\}, \theta'_{\mathsf{enc}} \cup \{x \mapsto \{v\}^{n'_r}_{\mathsf{pk}}\}, \lambda'_{\mathsf{dec}}\big)$$

if there exist $t, t' \in \mathcal{T}(\mathcal{F} \setminus \{\boldsymbol{0}\}, \mathcal{N}, \mathcal{X}_{\mathsf{enc}})$ such that:
- $(\phi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}}) R^{\mathcal{K}}_{\mathsf{CCA2}^a} (\psi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}})$
- $u \equiv t\lambda_{\mathsf{dec}}$, $v \equiv t'\lambda'_{\mathsf{dec}}$
- *$\mathsf{nodec}(\mathcal{K}; t, t')$, which ensures that the only decryptions are calls to the oracle.*
- *$\mathsf{fresh}(\mathsf{n}_r, \mathsf{n}'_r; \phi, u, \psi, v)$ and $\mathsf{hidden\text{-}rand}(\mathcal{X}_{\mathsf{enc}}\sigma_{\mathsf{rand}} \cup \mathcal{X}_{\mathsf{enc}}\sigma'_{\mathsf{rand}}; \phi, u, \psi, v)$*

3) ***Decryption Case:*** *Let $\mathsf{sk} \in \mathcal{K}$, pk the corresponding public key and z be a fresh variable. Then:*

$$\big(\left(\phi, \mathsf{else}^*(l, \mathsf{dec}(u, \mathsf{sk}))\right), \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}} \cup \{z\}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}} \cup \{z \mapsto \mathsf{else}^*(l, \mathsf{dec}(u, \mathsf{sk}))\}\big)$$
$$R^{\mathcal{K}}_{\mathsf{CCA2}^a} \big(\left(\psi, \mathsf{else}^*(l', \mathsf{dec}(v, \mathsf{sk}))\right), \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}} \cup \{z\}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}} \cup \{z \mapsto \mathsf{else}^*(l', \mathsf{dec}(v, \mathsf{sk}))\}\big)$$

if there exists $t \in \mathcal{T}(\mathcal{F} \setminus \{\mathsf{if\_then\_else\_}, \boldsymbol{0}\}, \mathcal{N}, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}})$ such that:
- $(\phi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}}) R^{\mathcal{K}}_{\mathsf{CCA2}^a} (\psi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}})$
- *$u \equiv t\theta_{\mathsf{enc}}\lambda_{\mathsf{dec}}$ and $v \equiv t\theta'_{\mathsf{enc}}\lambda'_{\mathsf{dec}}$.*
- *Consider the set $\mathcal{Y}_u$ of variables $x \in \mathcal{X}_{\mathsf{enc}}$ such that the encryption binded to x directly appears in u, i.e. appears outside of another encryption. That is, x must appear in the term u where we substituted every encryption $\{\_\}^{n_x}_{\mathsf{pk}} \in \mathsf{codom}(\theta_{\mathsf{enc}})$ by $\{0\}^{n_x}_{\mathsf{pk}}$:*
$$x\sigma_{\mathsf{rand}} \in u\big\{\{0\}^{n_x}_{\mathsf{pk}} / \{\_\}^{n_x}_{\mathsf{pk}} \mid \{\_\}^{n_x}_{\mathsf{pk}} \in \mathsf{codom}(\theta_{\mathsf{enc}})\big\} \downarrow_R$$
Then l is the sequence of guards $l \equiv (\mathsf{eq}(u, y_1), \ldots, \mathsf{eq}(u, y_m))$ where $(y_1, \ldots, y_m) = \mathsf{sort}(\mathcal{Y}_u\theta_{\mathsf{enc}})$.
Similarly, $l' \equiv (\mathsf{eq}(v, y'_1), \ldots, \mathsf{eq}(v, y'_m))$ where $(y'_1, \ldots, y'_m) = \mathsf{sort}(\mathcal{Y}_u\theta'_{\mathsf{enc}})$[2].
- *$\mathsf{nodec}(\mathcal{K}; t)$ and $\mathsf{hidden\text{-}rand}(\mathcal{X}_{\mathsf{enc}}\sigma_{\mathsf{rand}} \cup \mathcal{X}_{\mathsf{enc}}\sigma'_{\mathsf{rand}}; \phi, u, \psi, v)$*

where sort is a deterministic function sorting terms according to an arbitrary linear order.

Remark 4. In the decryption case, we add a guard only for encryption that appear directly in u. Without this restriction, we would add one guard $\mathsf{eq}(u, x\theta_{\mathsf{enc}})$ for every $x \in \mathcal{X}_{\mathsf{enc}}$ such that $x\theta_{\mathsf{enc}}$ is an encryption using public-key pk.

For example, if $\mathcal{X}_{\mathsf{enc}} = \{x_0, x_1, x_2\}$ and $\theta_{\mathsf{enc}} = \{x_0 \mapsto \alpha_0, x_1 \mapsto \alpha_1, x_2 \mapsto \alpha_2\}$ where:

$$\alpha_0 \mapsto \{m_0\}^{\mathsf{n}_0}_{\mathsf{pk}} \qquad \alpha_1 \mapsto \{m_1\}^{\mathsf{n}_1}_{\mathsf{pk}} \qquad \alpha_2 \mapsto \{\alpha_1\}^{\mathsf{n}_2}_{\mathsf{pk}}$$

then to guard $\mathsf{dec}(g(\alpha_2), \mathsf{sk})$, we need to add three guards, $\mathsf{eq}(g(\alpha_2), \alpha_0)$, $\mathsf{eq}(g(\alpha_2), \alpha_1)$ and $\mathsf{eq}(g(\alpha_2), \alpha_2)$. This yields the term:

$$\begin{array}{ll} \mathsf{if} & \mathsf{eq}(g(\alpha_2), \alpha_0) \;\mathsf{then}\; \boldsymbol{0}(\mathsf{dec}(g(\alpha_2), \mathsf{sk})) \\ \mathsf{else\ if} & \mathsf{eq}(g(\alpha_2), \alpha_1) \;\mathsf{then}\; \boldsymbol{0}(\mathsf{dec}(g(\alpha_2), \mathsf{sk})) \\ \mathsf{else\ if} & \mathsf{eq}(g(\alpha_2), \alpha_2) \;\mathsf{then}\; \boldsymbol{0}(\mathsf{dec}(g(\alpha_2), \mathsf{sk})) \\ \mathsf{else} & \qquad\qquad\qquad\qquad \mathsf{dec}(g(\alpha_2), \mathsf{sk}) \end{array}$$

But here, the adversary, represented by the adversarial function g, is computing the query to the decryption oracle using only α_2. Hence, it cannot use α_1, which is hidden by the encryption, nor α_0 which does not appear at all. Therefore, there is no need to add the guards $\mathsf{eq}(g(\alpha_2), \alpha_0)$ and $\mathsf{eq}(g(\alpha_2), \alpha_1)$, since g has a negligible probability of returning α_0 or α_1.

To remove unnecessary guards when building the decryption oracle call $\mathsf{dec}(u, \mathsf{sk})$, we require that $\mathsf{eq}(u, \alpha)$ is added to the list of guards if and only if $\alpha \equiv \{\_\}^{\mathsf{n}}_{\mathsf{pk}}$ appears directly in u. This yields smaller axioms, e.g. the term $\mathsf{dec}(g(\alpha_2), \mathsf{sk})$ is guarded by:

$$\begin{array}{l} \mathsf{if}\; \mathsf{eq}(g(\alpha_2), \alpha_2) \;\mathsf{then}\; \boldsymbol{0}(\mathsf{dec}(g(\alpha_2), \mathsf{sk})) \\ \qquad\qquad\qquad\qquad \mathsf{else}\; \mathsf{dec}(g(\alpha_2), \mathsf{sk}) \end{array}$$

Finally, the sort function is used to ensure that guards are always in the same order, which guarantees that two calls with the same terms are guarded in the same way.

[2]Remark that we use, for v, the set $\mathcal{Y}_u$ defined using u. As we will see later, this is not a problem because $\mathcal{Y}_u = \mathcal{Y}_v$.

We can now define the recursive set of axioms $\mathsf{CCA2}^a$ and show their validity. We also state and prove a key property of these axioms.

Definition 15. *$\mathsf{CCA2}^a$ is the set of unitary axioms $\phi \sim \psi\mu$, where μ is a renaming of names in $\mathcal{N}$ and there exist two* CCA2 *executions $\mathcal{Y}, \mathcal{Y}'$ such that:*

$$\mathcal{Y} = (\phi, \mathcal{X}_{\mathit{enc}}, \mathcal{X}_{\mathit{dec}}, \sigma_{\mathit{rand}}, \theta_{\mathit{enc}}, \lambda_{\mathit{dec}}) \qquad \mathcal{Y}' = (\psi, \mathcal{X}_{\mathit{enc}}, \mathcal{X}_{\mathit{dec}}, \sigma'_{\mathit{rand}}, \theta'_{\mathit{enc}}, \lambda'_{\mathit{dec}}) \qquad \mathcal{Y}\, R^{\mathcal{K}}_{\mathsf{CCA2}^a}\, \mathcal{Y}'$$

In that case, we say that $(\mathcal{Y}, \mathcal{Y}')$ is a valid $\mathsf{CCA2}^a$ application, and $\phi \sim \psi\mu$ is a valid $\mathsf{CCA2}^a$ instance.

Proposition 8. *All formulas in $\mathsf{CCA2}^a$ are computationally valid if the encryption scheme is* IND-CCA$_2$.

Proof. First, $\phi \sim \psi\mu$ is computationally valid if and only if $\phi \sim \psi$ is computationally valid. Hence, w.l.o.g. we consider μ empty. Let $\mathcal{M}_c$ be a computational model where the encryption and decryption symbol are interpreted as an IND-CCA$_2$ encryption scheme. Let $\phi \sim \psi$ be a valid instance of $\mathsf{CCA2}^a$ such that $[\![\phi]\!] \not\approx_{\mathcal{M}_c} [\![\psi]\!]$ i.e. there is a PPTM $\mathcal{A}$ that has a non-negligible advantage of distinguishing these two distributions.

Since $\phi \sim \psi$ is an instance of CCA2 we know that there exist two CCA2 executions such that:

$$(\phi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}}) R^{\mathcal{K}}_{\mathsf{CCA2}^a} (\psi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}})$$

We are going to build from ϕ and ψ a winning attacker against the multi-user IND-CCA$_2$ game. This attacker has access to a LR oracle and a decryption oracle for all keys in $\mathcal{K}$. We are going to build by induction on $R^{\mathcal{K}}_{\mathsf{CCA2}^a}$ a algorithm $\mathcal{B}$ that samples from $[\![\phi]\!]$ or $[\![\psi]\!]$ (depending on the oracles internal bit). The algorithm $\mathcal{B}$ uses a memoisation technique: it builds a store whose keys are subterms of ϕ, ψ already encountered and variable in $\mathcal{X}_{\mathsf{enc}} \cup \mathcal{X}_{\mathsf{dec}}$, and values are elements of the $\mathcal{M}_c$ domain.

1) $(\phi, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset) R^{\mathcal{K}}_{\mathsf{CCA2}^a} (\phi, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$: for every term t in the vector ϕ, $\mathcal{B}$ samples from $[\![t]\!]$ by induction as follows:
   - if t is in the store then $\mathcal{B}$ returns its value.
   - nonce n: $\mathcal{B}$ draws n uniformly at random and stores the drawn value.
     Remark that $\mathsf{nodec}(\mathcal{K}, \phi)$ ensures that n is not used in a secret key sk appearing in $\mathcal{K}$, which we could not compute. If it is a public key pk, either the corresponding secret key sk is such that $\mathsf{sk} \in \mathcal{K}$ and the challenger sent us a random sample from $[\![\mathsf{pk}]\!]$, or sk does not appear in $\mathcal{K}$ and then $\mathcal{B}$ can draw the corresponding key pair itself.
   - $f(t_1, \ldots, t_n)$, then $\mathcal{B}$ inductively samples the function arguments $([\![t_1]\!], \ldots, [\![t_1]\!])$ and then samples from $[\![f]\!]\,([\![t_1]\!], \ldots, [\![t_1]\!])$. $\mathcal{B}$ stores the value at the key $f(t_1, \ldots, t_n)$.

2) **Encryption Case**:

$$\big((\phi, \{u\}^{\mathsf{n}_r}_{\mathsf{pk}}), \mathcal{X}_{\mathsf{enc}} \cup \{x\}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}} \cup \{x \mapsto \mathsf{n}_r\}, \theta_{\mathsf{enc}} \cup \{x \mapsto \{u\}^{\mathsf{n}_r}_{\mathsf{pk}}\}, \lambda_{\mathsf{dec}}\big)$$
$$R^{\mathcal{K}}_{\mathsf{CCA2}^a} \big((\psi, \{v\}^{\mathsf{n}'_r}_{\mathsf{pk}}), \mathcal{X}_{\mathsf{enc}} \cup \{x\}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}} \cup \{x \mapsto \mathsf{n}'_r\}, \theta'_{\mathsf{enc}} \cup \{x \mapsto \{v\}^{\mathsf{n}'_r}_{\mathsf{pk}}\}, \lambda'_{\mathsf{dec}}\big)$$

   Since we have $\mathsf{fresh}(\mathsf{n}_r, \mathsf{n}'_r; \phi, u, \psi, v)$ we know that the top-level terms do not appear in the store. It is easy to check that $\mathcal{B}$ inductive definition is such that $\mathcal{B}$ store has a value associated with every variable in $\mathcal{X}_{\mathsf{enc}} \cup \mathcal{X}_{\mathsf{dec}}$ and that, if $x \in \mathcal{X}_{\mathsf{enc}}$, then the store value of x is either sampled from $[\![x\theta_{\mathsf{enc}}]\!]$ or from $[\![x\theta'_{\mathsf{enc}}]\!]$ (depending on the challenger internal bit), and that if $x \in \mathcal{X}_{\mathsf{dec}}$ then the store value of x is either sampled from $[\![x\lambda_{\mathsf{dec}}]\!]$ or from $[\![x\lambda'_{\mathsf{dec}}]\!]$ (depending on the challenger internal bit). We also observe that if the challenger internal bit is 0 then for all w:

$$\mathcal{O}_{\mathsf{LR}}(\mathsf{pk}, b)([\![u]\!], [\![v]\!]) = \mathcal{O}_{\mathsf{LR}}(\mathsf{pk}, b)([\![u]\!], w)$$

   Similarly if the challenger internal bit is 1 then for all w:

$$\mathcal{O}_{\mathsf{LR}}(\mathsf{pk}, b)([\![u]\!], [\![v]\!]) = \mathcal{O}_{\mathsf{LR}}(\mathsf{pk}, b)(w, [\![v]\!])$$

   $\mathcal{B}$ samples two values α, β such that if the challenger internal bit is 0 then α is sampled from $[\![u]\!]$ and if the challenger internal bit is 1 then β is sampled from $[\![v]\!]$. Therefore whatever the challenger internal is bit, $\mathcal{O}_{\mathsf{LR}}(\mathsf{pk}, b)(\alpha, \beta)$ is sampled from $\mathcal{O}_{\mathsf{LR}}(\mathsf{pk}, b)([\![u]\!], [\![v]\!])$:
   - α is sampled from $[\![u]\!]$ using the case 1 algorithm. Remark that when we encounter a decryption under $\mathsf{sk}' \in \mathcal{K}$, we know that it was already sampled and can therefore retrieve it from the store.
   - similarly, β is sampled from $[\![v]\!]$ using the case 1 algorithm.

   The condition $\mathsf{nodec}(\mathcal{K}; t, t')$ ensures that no secret key from $\mathcal{K}$ appears in u, v anywhere else than in decryption positions for already queried oracle calls (which can therefore be retrieved from the store), and the two conditions $\mathsf{fresh}(\mathsf{n}_r, \mathsf{n}'_r; \phi, u, \psi, v)$ and $\mathsf{hidden\text{-}rand}(\mathcal{X}_{\mathsf{enc}}\sigma_{\mathsf{rand}} \cup \mathcal{X}_{\mathsf{enc}}\sigma'_{\mathsf{rand}}; \phi, u, \psi, v)$ ensure that all randomness used by the challenger left-right oracles do not appear anywhere else than in encryption randomness position for the corresponding left-right oracle calls.

We store the result of the left-right oracle call at key x.

3) **Decryption Case**:

$$((\phi, \mathsf{else}^*(l, \mathsf{dec}(u, \mathsf{sk}))), \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}} \cup \{z\}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}} \cup \{z \mapsto \mathsf{else}^*(l, \mathsf{dec}(u, \mathsf{sk}))\})$$
$$R^{\mathcal{K}}_{\mathsf{CCA2}^a} ((\psi, \mathsf{else}^*(l', \mathsf{dec}(v, \mathsf{sk}))), \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}} \cup \{z\}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}} \cup \{z \mapsto \mathsf{else}^*(l', \mathsf{dec}(v, \mathsf{sk}))\})$$

We know that $u \equiv t\theta_{\mathsf{enc}}\lambda_{\mathsf{dec}}$ and $v \equiv t\theta'_{\mathsf{enc}}\lambda'_{\mathsf{dec}}$. $\mathcal{B}$ uses the case 1 algorithm to sample γ from $[\![t\theta_{\mathsf{enc}}\lambda_{\mathsf{dec}}]\!]$ or $[\![t\theta'_{\mathsf{enc}}\lambda'_{\mathsf{dec}}]\!]$ depending on the challenger internal bit. $\mathsf{nodec}(\mathcal{K}; t)$ ensures that no call to the decryption oracles are needed and $\mathsf{hidden\text{-}rand}(\mathcal{X}_{\mathsf{enc}}\sigma_{\mathsf{rand}} \cup \mathcal{X}_{\mathsf{enc}}\sigma'_{\mathsf{rand}}; \phi, u, \psi, v)$ guarantee that the randomness drawn by the challenger for LR oracle encryptions do not appear in t.

Observe that all calls to $\mathcal{O}_{\mathsf{LR}}(\mathsf{pk}, b)$ have already been stored. Let $x_1\theta_{\mathsf{enc}}, \dots, x_p\theta_{\mathsf{enc}}$ be the corresponding keys in the store. Hence if γ is equal to any of the values stored at keys $x_1\theta_{\mathsf{enc}}, \dots, x_p\theta_{\mathsf{enc}}$ then $\mathcal{B}$ return $[\![\mathbf{0}]\!](\gamma)$, otherwise $\mathcal{B}$ can call the decryption oracle $\mathcal{O}_{\mathsf{dec}}(\mathsf{sk})$ on γ.

As we observed in Remark 4, if the challenger internal bit is 0, checking whether γ is different from the values sampled from $[\![x_1\theta_{\mathsf{enc}}]\!], \dots, [\![x_p\theta_{\mathsf{enc}}]\!]$ amounts to checking whether γ is different from the values sampled from $[\![y_1]\!], \dots, [\![y_m]\!]$, except for a negligible number of samplings. Therefore we are sampling from the correct distribution (up to a negligible number of samplings).

Moreover, the set of variables $x \in \mathcal{X}_{\mathsf{enc}}$ such that the encryption binded to x in θ_{enc} appears directly in the *left decryption* u:

$$x\sigma_{\mathsf{rand}} \in u\big\{\{0\}^{\mathsf{n}_x}_{\mathsf{pk}} / \{\_\}^{\mathsf{n}_x}_{\mathsf{pk}} \mid \{\_\}^{\mathsf{n}_x}_{\mathsf{pk}} \in \mathsf{codom}(\theta_{\mathsf{enc}})\big\} \downarrow_R$$

is exactly the set of variables x such that the encryption binded to x in θ'_{enc} appears directly in the *right decryption* v:

$$x\sigma_{\mathsf{rand}} \in v\big\{\{0\}^{\mathsf{n}_x}_{\mathsf{pk}} / \{\_\}^{\mathsf{n}_x}_{\mathsf{pk}} \mid \{\_\}^{\mathsf{n}_x}_{\mathsf{pk}} \in \mathsf{codom}(\theta'_{\mathsf{enc}})\big\} \downarrow_R$$

Hence, if the internal bit is 1 then checking whether γ is different from the values sampled from $[\![x_1\theta'_{\mathsf{enc}}]\!], \dots, [\![x_p\theta'_{\mathsf{enc}}]\!]$ amounts to checking whether γ is different from the values sampled from $[\![y'_1]\!], \dots, [\![y'_m]\!]$, except for a negligible number of samplings.

We store the result at key z.

The attacker against the multi-user IND-CCA$_2$ game simply returns $\mathcal{A}(\mathcal{B})$. Since $\mathcal{B}$ samples either from $[\![\phi]\!]$ if $b = 0$ or from $[\![\psi]\!]$ if $b = 1$ (up to a negligible number of samplings), and since $\mathcal{A}$ has a non-negligible advantage of distinguishing $[\![\phi]\!]$ from $[\![\psi]\!]$ we know that the attacker has a non-negligible advantage against the multi-user IND-CCA$_2$ game. ■

A. *Closure Under* Restr

To close our logic under Restr, we need the unitary axioms to be closed. Therefore, we let CCA2 be the closure of CCA2^a under Restr.

Definition 16. CCA2 *is the set of formula $\phi \sim \psi$ such that we have the derivation:*

$$\dfrac{\dfrac{}{\phi' \sim \psi'}\ \mathsf{CCA2}^a}{\phi \sim \psi}\ \mathsf{Restr}$$

The main contribution of this sub-section, given below, states that any instance $\vec{u} \sim \vec{v}$ of CCA2 can be automatically extended into an instance $\vec{u}' \sim \vec{v}'$ of CCA2^a of, at most, polynomial size.

Proposition 9. *For every instance $\vec{u} \sim \vec{v}$ of* CCA2*, there exists $\vec{u}_1, \vec{v}_1$ such that $\vec{u}, \vec{u}_1 \sim \vec{v}, \vec{v}_1$ is an instance of* CCA2^a *(modulo* Perm*) and $|\vec{u}_1| + |\vec{v}_1|$ is of polynomial size in $|\vec{u}| + |\vec{v}|$. We let $\mathsf{completion}(\vec{u} \sim \vec{v})$ be the formula $\vec{u}, \vec{u}_1 \sim \vec{v}, \vec{v}_1$.*

Proof. We first show how to extend an instance of CCA2 into an instance of CCA2^a. Let $(u_i)_{i \in I} \sim (v_i)_{i \in I}$ be an instance of CCA2^a. Let $I' \subseteq I$, we want to extend $(u_i)_{i \in I'} \sim (v_i)_{i \in I'}$ into an instance of CCA2^a. Let $\phi \equiv (u_i)_{i \in I}$, $\psi \equiv (v_i)_{i \in I}$, since $(u_i)_{i \in I} \sim (v_i)_{i \in I}$ is an instance of CCA2^a we have:

$$(\phi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}}) R^{\mathcal{K}}_{\mathsf{CCA2}^a} (\psi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}})$$

For all $x \in \mathcal{X}_{\mathsf{enc}} \cup \mathcal{X}_{\mathsf{dec}}$, we let $i_x \in I$ be the index corresponding to $x\theta_{\mathsf{enc}}\lambda_{\mathsf{dec}} \sim x\theta'_{\mathsf{enc}}\lambda'_{\mathsf{dec}}$. Moreover, for all $x \in \mathcal{X}_{\mathsf{dec}}$, we let t_{i_x} be the context used for the decryption in the definition of $R^{\mathcal{K}}_{\mathsf{CCA2}^a}$ (hence we have $x\lambda_{\mathsf{dec}} \equiv \mathsf{else}^*(l, \mathsf{dec}(t_{i_x}\theta_{\mathsf{enc}}\lambda_{\mathsf{dec}}), \mathsf{sk})$).

a) Outline: We are going to define $I^{lr}, I^l, I^r \subseteq I$ and $(\tilde{u}_i)_{i\in J}$, $(\tilde{v}_i)_{i\in J}$ (where $J = I^{lr} \cup I^l \cup I^r$) such that:

- I^{lr}, I^l, I^r are pair-wise disjoints and $I' \subseteq I^{lr}$.
- $(\tilde{u}_i)_{i\in J} \sim (\tilde{v}_i)_{i\in J}$ is an instance of $\mathsf{CCA2}^a$ of polynomial size with respect to $\sum_{i\in I'} |u_i| + |v_i|$.

Intuitively, I^{lr} is the subset of indices of $I \backslash I'$ of the terms that are subterm of $(u_i)_{i\in I'} \sim (v_i)_{i\in I'}$ *on the left and on the right*, i.e. for all $i \in I^{lr}$, $u_i \in \mathsf{st}((u_i)_{i\in I'})$ *and* $v_i \in \mathsf{st}((v_i)_{i\in I'})$. The terms whose index is in I^{lr} are easy to handle, as they are immediately bounded by the terms whose indices is in I'.

Then, I^l is the subset of indices of $I \backslash I'$ of the terms that are subterms of $(u_i)_{i\in I'} \sim (v_i)_{i\in I'}$ *on the left only* (i.e. for every $i \in I^l$, we only know that $u_i \in \mathsf{st}((u_i)_{i\in I'})$). Terms with indices in I^l are easy to bound on the left, but not on the right. To bound the right terms, we introduce dummy messages (by replace encryptions by encryption of $g()$, where g is an adversarial function symbol in $\mathcal{G}$). Similarly I^r is the subset of indices of $I \backslash I'$ of the terms that are subterms of $(u_i)_{i\in I'} \sim (v_i)_{i\in I'}$ *on the right only*.

First, we define I^{lr}, I^l, I^r, and then we define the corresponding $\mathsf{CCA2}^a$ instance $(\tilde{u}_i)_{i\in J} \sim (\tilde{v}_i)_{i\in J}$.

b) Inductive Definition of the Left and Right Appearance Sets: We define by induction on $i \in I'$ the sets $I^l_i, I^r_i \subseteq I$. Intuitively, I^l_i is the set of indices of I needed so that u_i is well-defined (same for I^r_i and v_i). Let $i \in I'$, we do a case disjunction on the rule applied to u_i, v_i in $R^{\mathcal{K}}_{\mathsf{CCA2}^a}$:

- **No Call to the Oracles:** In that case we take $I^l_i = I^r_i = \{i\}$.
- **Encryption Case:** let $t, t' \in \mathcal{T}(\mathcal{F}\backslash\{\mathbf{0}\}, \mathcal{N}, \mathcal{X}_{\mathsf{dec}})$ such that $u_i \equiv \{t\lambda_{\mathsf{dec}}\}_{\_}^{\_}$ and $v_i \equiv \{t'\lambda'_{\mathsf{dec}}\}_{\_}^{\_}$. To have u_i well-defined, we need all the decryptions in u_i to be well-defined (same for v_i). Hence let:
$$I^l_i = \{i\} \cup \bigcup_{x\in\mathcal{X}_{\mathsf{dec}}\cap\mathsf{st}(t)} I^l_{i_x} \qquad\qquad I^r_i = \{i\} \cup \bigcup_{x\in\mathcal{X}_{\mathsf{dec}}\cap\mathsf{st}(t')} I^r_{i_x}$$
- **Decryption Case:** recall that $u_i \equiv \mathsf{else}^*(l, \mathsf{dec}(u, \mathsf{sk}))$ where $u \equiv t_i\theta_{\mathsf{enc}}\lambda_{\mathsf{dec}}$. Therefore we need all encryption in $\mathcal{X}_{\mathsf{enc}} \cap \mathsf{st}(t_i)$ and decryption in $\mathcal{X}_{\mathsf{dec}} \cap \mathsf{st}(t_i)$ to be defined, on the left and on the right. Hence we let:
$$I^l_i = \{i\} \cup \bigcup_{x\in(\mathcal{X}_{\mathsf{dec}}\cup\mathcal{X}_{\mathsf{enc}})\cap\mathsf{st}(t_i)} I^l_{i_x} \qquad\qquad I^r_i = \{i\} \cup \bigcup_{x\in(\mathcal{X}_{\mathsf{dec}}\cup\mathcal{X}_{\mathsf{enc}})\cap\mathsf{st}(t_i)} I^r_{i_x}$$

We let:
$$I^{lr} = \bigcup_{i\in I'} I^l_i \cap \bigcup_{i\in I'} I^r_i \qquad\qquad I^l = \bigcup_{i\in I'} I^l_i \cap \overline{\bigcup_{i\in I'} I^r_i} \qquad\qquad I^r = \overline{\bigcup_{i\in I'} I^l_i} \cap \bigcup_{i\in I'} I^r_i$$

These three sets are disjoint and form a partition of $\bigcup_{i\in I'} I^l_i \cup I^r_i$. Remark that for every $i \in I^l_j$, u_i is a subterm of u_j. Hence, for every $i \in I^{lr} \cup I^l$, there exists $j \in I'$ such that u_i is a subterm of u_j.

c) Building the New Instance: We define (by induction on i) the terms $(\tilde{u}_i)_{i\in J}$, by letting $\tilde{u}_i$ be:

- u_i when $i \in I^{lr} \cup I^l$.
- $\{g()\}^{\mathsf{n}}_{\mathsf{pk}}$ when $i \in I^r$ and u_i is an encryption, with $u_i \equiv \{\_\}^{\mathsf{n}}_{\mathsf{pk}}$.
- $\mathsf{else}^*(\tilde{l}, \mathsf{dec}(\tilde{u}, \mathsf{sk}))$ when $i \in I^r$ and u_i is a decryption, where $u_i \equiv \mathsf{else}^*(l, \mathsf{dec}(u, \mathsf{sk}))$, $u \equiv t_i\theta_{\mathsf{enc}}\lambda_{\mathsf{dec}}$, l is the sequence of guards $l \equiv (\mathsf{eq}(u, y_1), \dots, \mathsf{eq}(u, y_m))$ where $(y_1, \dots, y_m) = \mathsf{sort}(\mathcal{Y}_u\theta_{\mathsf{enc}})$. Then we take:
  - $\tilde{u} \equiv t_i\tilde{\theta}_{\mathsf{enc}}\tilde{\lambda}_{\mathsf{dec}}$, where $\tilde{\theta}_{\mathsf{enc}} = \{x \mapsto \tilde{u}_{i_x} \mid x \in \mathcal{X}_{\mathsf{enc}}\}$ and $\tilde{\lambda}_{\mathsf{dec}} = \{x \mapsto \tilde{u}_{i_x} \mid x \in \mathcal{X}_{\mathsf{dec}}\}$.
  - $\tilde{l} \equiv (\mathsf{eq}(\tilde{u}, \tilde{y}_1), \dots, \mathsf{eq}(\tilde{u}, \tilde{y}_m))$ where $(\tilde{y}_1, \dots, \tilde{y}_m) = \mathsf{sort}(\mathcal{Y}_u\tilde{\theta}_{\mathsf{enc}})$.

Similarly, we define $\tilde{v}_i$ for every $i \in J$.

d) Conclusion: Let $J = I^{lr} \cup I^l \cup I^r$. To conclude, we check that $(\tilde{u}_i)_{i\in J} \sim (\tilde{v}_i)_{i\in J}$:

- is a $\mathsf{CCA2}^a$ instance. This is done by induction on $i \in J$.
- is of polynomial size w.r.t. $(u_i)_{i\in I'} \sim (v_i)_{i\in I'}$.

We omit the details of the proof of the first point.

For the second point, we first show by induction on i that $|I^l_i| \leq |u_i|$ and $|I^r_i| \leq |v_i|$. We deduce that:
$$|J| = \big|\bigcup_{i\in I'} I^r_i \cup I^l_i\big| \leq \sum_{i\in I'} |I^r_i| + |I^l_i| \leq \sum_{i\in I'} |u_i| + |v_i|$$

Let $i \in I^{lr} \cup I^l$, we know that there exists $j \in I'$ such that u_i is a subterm of u_j. Since $\tilde{u}_i \equiv u_i$, we deduce that $|\tilde{u}_i| \leq |u_j| \leq \sum_{j\in I'} |u_j| + |v_j|$.

Let $i \in I^r$. If $\tilde{u}_i$ is an encryption then it is of constant size. Assume $\tilde{u}_i$ is a decryption. Then $\tilde{u}_i$ is the decryption v_i where any encryption whose index is in I^{lr} has been replaced by its left counterpart, and any encryption whose index is in I^r has been replaced by a dummy encryption (the case I^l cannot happen, since $i \in I^r$). Since there are at most $|v_i| - 1$ such

$$\mathsf{Length}(\mathsf{n}) = l_\eta \qquad\qquad \mathsf{Length}(0_{l_e}) = l_e$$

$$\mathsf{Length}(u) = \mathsf{Length}(u') \text{ if } u =_R u' \text{ and } \mathsf{Length}(u), \mathsf{Length}(u') \text{ are not } \mathsf{undefined}$$

$$\mathsf{Length}(\langle u\,,\,v\rangle) = \mathsf{Length}(u) + \mathsf{Length}(v) + l_{\langle\,,\,\rangle} \qquad\qquad \forall l_e.\mathsf{Length}(\mathsf{pad}_{l_e}(u)) = l_e$$

$$\forall k.\mathsf{Length}(\{u\}_{\mathsf{pk}}^{\mathsf{n}}) = k.\mathsf{l}_{\{\mathsf{block}\}} + l_{\{\}} \text{ if } \mathsf{Length}(u) = k.\mathsf{l}_{\mathsf{block}}$$

$$\forall k.\mathsf{Length}(\mathsf{dec}(u, \mathsf{sk})) = k.\mathsf{l}_{\mathsf{block}} \text{ if } \mathsf{Length}(u) = k.\mathsf{l}_{\{\mathsf{block}\}} + l_{\{\}}$$

$$\mathsf{Length}(\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v) = \begin{cases} \mathsf{Length}(u) & \text{if } \mathsf{Length}(u) = \mathsf{Length}(v) \\ \mathsf{undefined} & \text{otherwise} \end{cases}$$

Fig. 6. Definition of the Length partial function.

encryptions (as v_i contain at least one occurrence of the dec function symbol), and since any encryption with index in I^{lr} or I^r is upper-bounded by $\sum_{j \in I'} |u_j| + |v_j|$, we get that:

$$|\tilde{u}_i| \leq |v_i| + (|v_i| - 1). \sum_{j \in I'} |u_j| + |v_j| \leq |v_i|. \sum_{j \in I'} |u_j| + |v_j| \leq \big(\sum_{j \in I'} |u_j| + |v_j| \big)^2$$

We deduce that $(\tilde{u}_i)_{i \in J} \sim (\tilde{v}_i)_{i \in J}$ is of polynomial size in $\sum_{j \in I'} |u_j| + |v_j|$. ■

B. Length in the CCA2 *Axioms*

If we want the formula $\{t\}_{\mathsf{pk}}^{r} \sim \{t'\}_{\mathsf{pk}'}^{r'}$ to be a valid application of the CCA2 axioms, we need to make sure that t and t' are of the same length. Since the length of terms depend on implementation details (e.g. how is the pair $\langle \_\,,\,\_ \rangle$ implemented), we let the user supply implementation assumptions. We use a predicate symbol $\mathsf{EQL}(\_,\,\_)$ in the logic, together with some derivation rules $\mathcal{D}_{\mathsf{L}}$ (supplied by the user), and we require that they verify the following properties:

- **Complexity:** for every u, v, we can decide whether $\mathsf{EQL}(u,\,v)$ is a consequence of $\mathcal{D}_{\mathsf{L}}$ in polynomial time in $|u| + |v|$.
- **Branch Invariance:** for all term b, u, v, t, if $\mathsf{EQL}(\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v,\, t)$ is derivable using $\mathcal{D}_{\mathsf{L}}$ then $\mathsf{EQL}(u,\,t)$ and $\mathsf{EQL}(v,\,t)$ are derivable using $\mathcal{D}_{\mathsf{L}}$.

We add to all CCA2 instances the side condition $\mathsf{EQL}(m_l,\,m_r)$ for every encryption oracle call on (m_l, m_r). Then, we know that our CCA2 instances are valid in any computational model $\mathcal{M}_c$ where the encryption is interpreted as a IND-CCA$_2$ encryption scheme, and where the following property holds: for every ground terms u, v, if $\mathsf{EQL}(u,\,v)$ is derivable using $\mathcal{D}_{\mathsf{L}}$, then:

$$[\![\mathsf{length}(u)]\!]_{\mathcal{M}_c} = [\![\mathsf{length}(v)]\!]_{\mathcal{M}_c}$$

a) Example: Block Cipher: We give here an example of derivation rules $\mathcal{D}_{\mathsf{L}}$ that axiomatize the fact that the encryption function is built upon a block cipher, taking blocks of length $\mathsf{l}_{\mathsf{block}}$ and returning blocks of length $\mathsf{l}_{\{\mathsf{block}\}}$. The length constant $l_{\{\}}$ is used to represent the constant length used, e.g., for the IV and the HMAC.

We let $\mathcal{L}$ be a set of length constants, and we define a length expression to be an expression of the form $\sum_{l \in L} k_l.l$, where L is a finite subset of $\mathcal{L}$ and $(k_l)_{l \in L}$ are positive integers. We consider length expressions modulo commutativity (i.e. $3.l_1 + 4.l_2 \approx 4.l_2 + 3.l_1$), and we assume that for every length expression l_e, there exists a function symbol $\mathsf{pad}_{l_e} \in \mathcal{F}$. Intuitively pad_{l_e} is function padding messages to length l: if the message is too long it truncates it, and if the message is too short it pads it. Similarly, we assume that for every l_e, we have a function symbol $0_{l_e} \in \mathcal{F}$ or arity zero which, intuitively, returns l_e zeroes. Also, we assume that $\mathcal{L}$ contains the following length constants: $l_{\langle\,,\,\rangle}, l_{enc}, \mathsf{l}_{\mathsf{block}}, l_\eta$.

We define the Length (partial) function on terms in Figure 6. Then, we let $\mathcal{D}_{\mathsf{L}}$ be the (recursive) set of unitary axioms:

$$\frac{\mathsf{Length}(u) = \mathsf{Length}(v) \neq \mathsf{undefined}}{\mathsf{EQL}(u,\,v)}$$

Proposition 10. *The function Length is well defined, and the set of axioms $\mathcal{D}_{\mathsf{L}}$ satisfies the branch invariance properties.*

Proof. To check that Length is well defined, one just need to look at the critical pairs in the definition and check that they are joinable. Soundness is easy, as $[\![\mathsf{Length}]\!]_{\mathcal{M}_c}$ is just an under-approximation of $[\![\mathsf{length}]\!]_{\mathcal{M}_c}$ in every computational model $\mathcal{M}_c$ where the encryption is interpreted as a block cipher, the padding functions are interpreted as expected etc.

Finally, branch invariance follows directly from the definition of $\mathsf{Length}(\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v)$. ■

Remark 5. We can allow the user to add any set of length equations, as long as the branch invariance property holds and the Length function is well-defined. E.g one may wish to add equations like $\mathsf{Length}(A) = \mathsf{Length}(B) = \mathsf{Length}(C) = l_{\mathsf{agent}}$.

Appendix III
Rule Ordering and Freeze Strategy

In this section, we give the proofs of the Restr elimination lemma (Lemma 1). We then show the rule commutations used to obtain a complete ordered strategy (Lemma 3, Lemma 5). Finally we show the completeness of the freeze strategy (Lemma 7).

A. Tracking Relations Between Branches

We introduce the following erasure function, defined on if-free ground terms inductively as follows:

$$\mathsf{2erase}(t) \equiv \begin{cases} f(\mathsf{2erase}(t_1), \dots, \mathsf{2erase}(t_n)) & \text{if } t \equiv f(t_1, \dots, t_n) \wedge f \in \mathcal{F}_s \\ \mathsf{2erase}(b) & \text{if } t \equiv \boxed{b_1}\boxed{b_2}_{\,b} \\ \mathsf{n} & \text{if } t \equiv \mathsf{n} \wedge \mathsf{n} \in \mathcal{N} \end{cases}$$

This function is used to define the full (not simplified) versions of UnF and 2Box, which are given in Fig. 7, together with a summary of all the axioms introduced for the complete strategy.

Remark 6. We modify the definition of $\mathsf{cond\text{-}st}(t)$ as follows: for all t, $\mathsf{cond\text{-}st}(t) = \mathsf{cond\text{-}st}(\mathsf{2erase}(t))$.

B. Proof Ordering

We now show that all the rule commutations given in Fig. 8 are correct. Observe that this subsumes Lemma 3 and Lemma 5.

Lemma 11. *All the rule commutations in Fig. 8 are correct.*

Proof. We split the proof depending on the left-most rule we are commuting.

a) Delay Dup*:*

- If the R rules involves a term which is not duplicated then this is trivial. Assume the R rewriting involves a duplicated term, and that $t =_R s$ and $t' =_R s'$:

$$\dfrac{\dfrac{\vec{u}, \vec{v}, s \sim \vec{u}', \vec{v}', s'}{\vec{u}, \vec{v}, t \sim \vec{u}', \vec{v}', t'}\ R}{\vec{u}, \vec{v}, t, \vec{v}, t \sim \vec{u}', \vec{v}', t', \vec{v}', t'}\ \mathsf{Dup} \quad \Rightarrow \quad \dfrac{\dfrac{\vec{u}, \vec{v}, s \sim \vec{u}', \vec{v}', s'}{\vec{u}, \vec{v}, s, \vec{v}, s \sim \vec{u}', \vec{v}', s', \vec{v}', s'}\ \mathsf{Dup}}{\vec{u}, \vec{v}, t, \vec{v}, t \sim \vec{u}', \vec{v}', t', \vec{v}', t'}\ R$$

- Similarly if the FA rules does not involve a duplicated term then this is trivial. Otherwise:

$$\dfrac{\dfrac{\vec{u}, \vec{v}, \vec{w} \sim \vec{u}', \vec{v}', \vec{w}'}{\vec{u}, \vec{v}, f(\vec{w}) \sim \vec{u}', \vec{v}', f(\vec{w}')}\ \mathsf{FA}}{\vec{u}, \vec{v}, f(\vec{w}), \vec{v}, f(\vec{w}) \sim \vec{u}', \vec{v}', f(\vec{w}'), \vec{v}', f(\vec{w}')}\ \mathsf{Dup} \quad \Rightarrow \quad \dfrac{\dfrac{\dfrac{\vec{u} \sim \vec{u}'}{\vec{u}, \vec{v}, \vec{w}, \vec{v}, \vec{w} \sim \vec{u}', \vec{v}', \vec{w}', \vec{v}', \vec{w}'}\ \mathsf{Dup}}{\vec{u}, \vec{v}, f(\vec{w}), \vec{v}, \vec{w} \sim \vec{u}', \vec{v}', f(\vec{w}'), \vec{'v}, \vec{w}'}\ \mathsf{FA}}{\vec{u}, \vec{v}, f(\vec{w}), \vec{v}, f(\vec{w}) \sim \vec{u}', \vec{v}', f(\vec{w}'), \vec{v}', f(\vec{w}')}\ \mathsf{FA}$$

- Commutation of Dup with CS is easy.

b) Delay FA*:*

- For every $b, b' \in \mathcal{T}(\mathcal{F}_s, \mathcal{N})$:

$$\dfrac{\dfrac{\vec{w}_1, \vec{w}_2, b, (u_i)_{i \in I \cup J} \sim \vec{w}'_1, \vec{w}'_2, b', (u'_i)_{i \in I \cup J} \qquad \vec{w}_1, \vec{w}_2, b, (v_i)_{i \in I \cup J} \sim \vec{w}'_1, \vec{w}'_2, b', (v'_i)_{i \in I \cup J}}{\vec{w}_1, \vec{w}_2, (\text{if } b \text{ then } u_i \text{ else } v_i)_{i \in I \cup J} \sim \vec{w}'_1, \vec{w}'_2, (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_{i \in I \cup J}}\ \mathsf{CS}}{\begin{array}{c} \vec{w}_1, (\text{if } b \text{ then } u_i \text{ else } v_i)_{i \in I}, f(\vec{w}_2, (\text{if } b \text{ then } u_i \text{ else } v_i)_{i \in J}) \\ \sim \quad \vec{w}'_1, (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_{i \in I}, f(\vec{w}'_2, (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_{i \in J}) \end{array}}\ \mathsf{FA}$$

  Can be rewritten into:

$$\dfrac{\dfrac{\dfrac{\vec{w}_1, \vec{w}_2, b, (u_i)_{i \in I \cup J} \sim \vec{w}'_1, \vec{w}'_2, b', (u'_i)_{i \in I \cup J}}{\begin{array}{c}\vec{w}_1, b, (u_i)_{i \in I}, f(\vec{w}_2, (u_i)_{i \in J}) \\ \sim \quad \vec{w}'_1, b', (u'_i)_{i \in I}, f(\vec{w}'_2, (u'_i)_{i \in J})\end{array}}\ \mathsf{FA} \quad \dfrac{\vec{w}_1, \vec{w}_2, b, (v_i)_{i \in I \cup J} \sim \vec{w}'_1, \vec{w}'_2, b', (v'_i)_{i \in I \cup J}}{\begin{array}{c}\vec{w}_1, b, (v_i)_{i \in I}, f(\vec{w}_2, (v_i)_{i \in J}) \\ \sim \quad \vec{w}'_1, b', (v'_i)_{i \in I}, f(\vec{w}'_2, (v'_i)_{i \in J})\end{array}}\ \mathsf{FA}}{\begin{array}{c}\vec{w}_1, (\text{if } b \text{ then } u_i \text{ else } v_i)_{i \in I}, \text{if } b \text{ then } f(\vec{w}_2, (u_i)_{i \in J}) \text{ else } f(\vec{w}_2, (v_i)_{i \in J}) \\ \sim \quad \vec{w}'_1, (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_{i \in I}, \text{if } b' \text{ then } f(\vec{w}'_2, (u'_i)_{i \in J}) \text{ else } f(\vec{w}'_2, (v'_i)_{i \in J})\end{array}}\ \mathsf{CS}}{\begin{array}{c}\vec{w}_1, (\text{if } b \text{ then } u_i \text{ else } v_i)_{i \in I}, f(\vec{w}_2, (\text{if } b \text{ then } u_i \text{ else } v_i)_{i \in J}) \\ \sim \quad \vec{w}'_1, (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_{i \in I}, f(\vec{w}'_2, (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_{i \in J})\end{array}}\ R$$

- Assume that $\vec{u}, \vec{v}, \vec{u}', \vec{v}' =_R \vec{u}_1, \vec{v}_1, \vec{u}'_1, \vec{v}'_1$:

$$\dfrac{\dfrac{\vec{u}_1, \vec{v}_1 \sim \vec{u}'_1, \vec{v}'_1}{\vec{u}, \vec{v} \sim \vec{u}', \vec{v}'}\ R}{\vec{u}, f(\vec{v}) \sim \vec{u}', f(\vec{v}')}\ \mathsf{FA} \quad \Rightarrow \quad \dfrac{\dfrac{\vec{u}_1, \vec{v}_1 \sim \vec{u}'_1, \vec{v}'_1}{\vec{u}_1, f(\vec{v}_1) \sim \vec{u}'_1, f(\vec{v}'_1)}\ \mathsf{FA}}{\vec{u}, f(\vec{v}) \sim \vec{u}', f(\vec{v}')}\ R$$

- (Sym) : $\sim$ is symmetric.
- For any permutation π of $1, \dots, n$: $\dfrac{x_{\pi(1)}, \dots, x_{\pi(n)} \sim y_{\pi(1)}, \dots, y_{\pi(n)}}{x_1 \dots, x_n \sim y_1, \dots, y_n}\ Perm$
- $\dfrac{\vec{u}, t \sim \vec{v}, t'}{\vec{u}, t, t \sim \vec{v}, t', t'}\ \mathsf{Dup}$
- If $s =_R t$ and $\{\boxed{a \mid c}_b \in \mathsf{st}(\vec{u}, t)\} \subseteq \{\boxed{a \mid c}_b \in \mathsf{st}(\vec{u}, C[s])\}$ then: $\dfrac{\vec{u}, C[t] \sim \vec{v}}{\vec{u}, C[s] \sim \vec{v}}\ R_\square$
- For all $f \in \mathcal{F}$, $\dfrac{\vec{x}, \vec{y} \sim \vec{x'}, \vec{y'}}{f(\vec{x}), \vec{y} \sim f(\vec{x'}), \vec{y'}}\ \mathsf{FA}$
- For every $b, b' \in \mathcal{T}(\mathcal{F}_s, \mathcal{N})$:

$$\frac{\vec{w}, b_1, (u_i)_i \sim \vec{w}', b_1', (u_i')_i \qquad \vec{w}, b_2, (v_i)_i \sim \vec{w}', b_2', (v_i')_i}{\vec{w}, \left(\mathsf{if}\ \boxed{b_1 \mid b_2}_b\ \mathsf{then}\ u_i\ \mathsf{else}\ v_i\right)_i \sim \vec{w}', \left(\mathsf{if}\ \boxed{b_1' \mid b_2'}_{b'}\ \mathsf{then}\ u_i'\ \mathsf{else}\ v_i'\right)_i}\ \mathsf{CS}_\square$$

- UnF unfreezes all conditionals.
- For every $b, b' \in \mathcal{T}(\mathcal{F}_s \cup \mathcal{B}, \mathcal{N})$:

$$\frac{\vec{u}, C\left[\boxed{b \mid b}_{\mathsf{2erase}(b)\downarrow_R}\right] \sim \vec{u}', C'\left[\boxed{b' \mid b'}_{\mathsf{2erase}(b')\downarrow_R}\right]}{\vec{u}, C[b] \sim \vec{u}', C'[b']}\ \mathsf{2Box}$$

Fig. 7. Summary of the strategy axioms.

$\mathsf{Dup} \cdot R$	$\Rightarrow$	$R \cdot \mathsf{Dup}$
$\mathsf{Dup} \cdot \mathsf{FA}$	$\Rightarrow$	$\mathsf{FA}^* \cdot \mathsf{Dup}$
$\mathsf{Dup} \cdot \mathsf{CS}$	$\Rightarrow$	$\mathsf{CS} \cdot \mathsf{Dup}$
$\mathsf{FA} \cdot R$	$\Rightarrow$	$R \cdot \mathsf{FA}$
$\mathsf{FA} \cdot \mathsf{CS}$	$\Rightarrow$	$R \cdot \mathsf{CS} \cdot \mathsf{FA}$
$\mathsf{FA_s} \cdot \mathsf{FA}(b, b')$	$\Rightarrow$	$R \cdot \mathsf{FA}(b, b') \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}$
$\mathsf{CS}_\square \cdot R_\square$	$\Rightarrow$	$R_\square \cdot \mathsf{CS}_\square$
$\mathsf{CS}_\square \cdot \mathsf{2Box}$	$\Rightarrow$	$R_\square \cdot \mathsf{2Box} \cdot \mathsf{CS}_\square$

Explanation: Each entry $w \Rightarrow w'$ means that a derivation in w can be rewritten into a derivation in w'.

Fig. 8. Summary of all the rule commutations

- For all f, b, b', one can always apply FA_f after $\mathsf{FA}(b, b')$:

$$\cfrac{\cfrac{\vec{u}, \vec{v}, b, s, t \sim \vec{u}', \vec{v}', b', s', t'}{\vec{u}, \vec{v}, \mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ t \sim \vec{u}', \vec{v}', \mathsf{if}\ b'\ \mathsf{then}\ s'\ \mathsf{else}\ t'}\ \mathsf{FA}(b, b')}{\vec{u}, f(\vec{v}, \mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ t) \sim \vec{u}', f(\vec{v}', \mathsf{if}\ b'\ \mathsf{then}\ s'\ \mathsf{else}\ t')}\ \mathsf{FA}_f$$

Then we can rewrite this proof as follows:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\vec{u}, b, s, \vec{v}, t \sim \vec{u}', b', s', \vec{v}', t'}{\vec{u}, b, \vec{v}, s, \vec{v}, t \sim \vec{u}', b', \vec{v}', s', \vec{v}', t'}\ \mathsf{Dup}}{\vec{u}, b, \vec{v}, s, f(\vec{v}, t) \sim \vec{u}', b', \vec{v}', s', f(\vec{v}', t')}\ \mathsf{FA}_f}{\vec{u}, b, f(\vec{v}, s), f(\vec{v}, t) \sim \vec{u}', b', f(\vec{v}', s'), f(\vec{v}', t')}\ \mathsf{FA}_f}{\vec{u}, \mathsf{if}\ b\ \mathsf{then}\ f(\vec{v}, s)\ \mathsf{else}\ f(\vec{v}, t) \sim \vec{u}', \mathsf{if}\ b'\ \mathsf{then}\ f(\vec{v}', s')\ \mathsf{else}\ f(\vec{v}', t')}\ \mathsf{FA}(b, b')}{\vec{u}, f(\vec{v}, \mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ t) \sim \vec{u}', f(\vec{v}', \mathsf{if}\ b'\ \mathsf{then}\ s'\ \mathsf{else}\ t')}\ R$$

- $\mathsf{FA}(b,b') - \mathsf{FA}(a,a')$ commutation: assume that $u =_R \text{if } a \text{ then } s \text{ else } t$ and that $u' =_R \text{if } a' \text{ then } s' \text{ else } t'$.

$$\dfrac{\dfrac{\dfrac{\vec{w},b,a,s,t,v \sim \vec{w}',b',a',s',t',v'}{\vec{w},b,\text{if } a \text{ then } s \text{ else } t,v \sim \vec{w}',b',\text{if } a' \text{ then } s' \text{ else } t',v'}\;\mathsf{FA}(a,a')}{\vec{w},b,u,v \sim \vec{w}',b',u',v'}\;R}{\vec{w},\text{if } b \text{ then } u \text{ else } v \sim \vec{w}',\text{if } b' \text{ then } u' \text{ else } v'}\;\mathsf{FA}(b,b')$$

Then we can rewrite this proof as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vec{w},a,b,s,t,v \sim \vec{w}',a',b',s',t',v'}{\vec{w},a,b,s,v,b,t,v \sim \vec{w}',a',b',s',v',b',t',v'}\;\mathsf{Dup}}{\vec{w},a,b,s,v,\text{if } b \text{ then } t \text{ else } v \sim \vec{w}',a',b',s',v',\text{if } b' \text{ then } t' \text{ else } v'}\;\mathsf{FA}(b,b')}{\vec{w},a,\text{if } b \text{ then } s \text{ else } v,\text{if } b \text{ then } t \text{ else } v \sim \vec{w}',a',\text{if } b' \text{ then } s' \text{ else } v',\text{if } b' \text{ then } t' \text{ else } v'}\;\mathsf{FA}(b,b')}{\begin{array}{c}\vec{w},\ \text{if } a \text{ then if } b \text{ then } s \text{ else } v \\ \text{else if } b \text{ then } t \text{ else } v\end{array} \sim \begin{array}{c}\vec{w}',\ \text{if } a' \text{ then if } b' \text{ then } s' \text{ else } v' \\ \text{else if } b' \text{ then } t' \text{ else } v'\end{array}}\;\mathsf{FA}(a,a')}{\vec{w},\text{if } b \text{ then } u \text{ else } v \sim \vec{w}',\text{if } b' \text{ then } u' \text{ else } v'}\;R$$

c) Delay CS*:*

- For all $b, b' \in \mathcal{T}(\mathcal{F}_s, \mathcal{N})$, the rule application:

$$\dfrac{\dfrac{(w_n)^0_n, b^0, (u^0_i)_i \sim (w'^0_n)_n, b'^0, (u'^0_i)_i}{(w_n)_n, b, (u_i)_i \sim (w'_n)_n, b', (u'_i)_i}\;R \qquad (w_n)_n, b, (v_i)_i \sim (w'_n)_n, b', (v'_i)_i}{(w_n)_n, (\text{if } b \text{ then } u_i \text{ else } v_i)_i \sim (w'_n)_n, (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_i}\;\mathsf{CS}$$

can be rewritten into:

$$\dfrac{\dfrac{\dfrac{(w_n)^0_n, b^0, (u^0_i)_i \sim (w_n)'^0_n, b'^0, (u'^0_i)_i}{(w_n)^0_n, b, (u^0_i)_i \sim (w_n)'^0_n, b', (u'^0_i)_i}\;R^{\mathsf{free}} \qquad (w_n)_n, b, (v_i)_i \sim (w_n)'_n, b', (v'_i)_i}{(\text{if } b \text{ then } w^0_n \text{ else } w_n)_n, (\text{if } b \text{ then } u^0_i \text{ else } v_i)_i \sim (\text{if } b' \text{ then } w'^0_n \text{ else } w'_n)_n, (\text{if } b' \text{ then } u'^0_i \text{ else } v'_i)_i}\;\mathsf{CS}}{(w_n)_n, (\text{if } b \text{ then } u_i \text{ else } v_i)_i \sim (w'_n)_n, (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_i}\;R$$

d) Delay $\mathsf{CS}_\square$*:*

- The following proof:

$$\dfrac{\dfrac{(w^1_j)_j, b_1, (u^1_i)_i \sim (w'^1_j)_j, b'_1, (u'^1_i)_i}{(w_j)_j, a_1, (u_i)_i \sim (w'_j)_j, a'_1, (u'_i)_i}\;R_\square \quad \dfrac{(w^2_j)_j, b_2, (v^1_i)_i \sim (w'^2_j)_j, b'_2, (v'^1_i)_i}{(w_j)_j, a_2, (v_i)_i \sim (w'_j)_j, a'_2, (v'_i)_i}\;R_\square}{(w_j)_j, (\text{if } \boxed{a_1 \mid a_2}_b \text{ then } u_i \text{ else } v_i)_i \sim (w'_j)_j, (\text{if } \boxed{a'_1 \mid a'_2}_{b'} \text{ then } u'_i \text{ else } v'_i)_i}\;\mathsf{CS}_\square$$

can be rewritten into:

$$\dfrac{\dfrac{(w^1_j)_j, b_1, (u^1_i)_i \sim (w'^1_j)_j, b'_1, (u'^1_i)_i \quad (w^2_j)_j, b_2, (v^1_i)_i \sim (w'^1_j)_j, b'_2, (v'^1_i)_i}{\begin{array}{cc} & (\text{if } \boxed{b_1 \mid b_2}_b \text{ then } w^1_j \text{ else } w^2_j)_j, (\text{if } \boxed{b_1 \mid b_2}_b \text{ then } u^1_i \text{ else } v^1_i)_i \\ \sim & (\text{if } \boxed{b'_1 \mid b'_2}_{b'} \text{ then } w'^1_j \text{ else } w'^2_j)_j, (\text{if } \boxed{b'_1 \mid b'_2}_{b'} \text{ then } u'^1_i \text{ else } v'^1_i)_i\end{array}}\;\mathsf{CS}_\square}{(w_j)_j, (\text{if } \boxed{a_1 \mid a_2}_b \text{ then } u_i \text{ else } v_i)_i \sim (w'_j)_j, (\text{if } \boxed{a'_1 \mid a'_2}_{b'} \text{ then } u'_i \text{ else } v'_i)_i}\;R_\square$$

- Similarly we can commute $\mathsf{CS}_\square$ with $\mathsf{2Box}$. Let $b, b' \in \mathcal{T}(\mathcal{F}_s \cup \mathcal{B}, \mathcal{N})$, and let:

$$b_\square \equiv \boxed{b \mid b}_{\mathsf{2erase}(b)\downarrow_R} \qquad \wedge \qquad b'_\square \equiv \boxed{b' \mid b'}_{\mathsf{2erase}(b')\downarrow_R}$$

Then the following proof:

$$\dfrac{\dfrac{\begin{array}{cc} & (w_j[b_\square])_j, a_1[b_\square], (u_i[b_\square])_i \\ \sim & (w'_j[b'_\square])_j, a'_1[b'_\square], (u'_i[b'_\square])_i\end{array}}{\begin{array}{cc} & (w_j[b])_j, a_1[b], (u_i[b])_i \\ \sim & (w'_j[b'])_j, a'_1[b'], (u'_i[b'])_i\end{array}}\;\mathsf{2Box} \qquad \begin{array}{cc} & (w_j[b])_j, a_2[b], (v_i[b])_i \\ \sim & (w'_j[b'])_j, a'_2[b'], (v'_i[b'])_i\end{array}}{\begin{array}{cc} & (w_j[b])_j, \left(\text{if } \boxed{a_1[b] \mid a_2[b]}_a \text{ then } u_i[b] \text{ else } v_i[b]\right)_i \\ \sim & (w'_j[b'])_j, \left(\text{if } \boxed{a'_1[b'] \mid a'_2[b']}_{a'} \text{ then } u'_i[b'] \text{ else } v'_i[b']\right)_i\end{array}}\;\mathsf{CS}_\square$$

can be rewritten into:

$$\dfrac{\dfrac{\dfrac{\begin{array}{rl} & (w_j[b_\square])_j, a_1[b_\square], (u_i[b_\square])_i \\ \sim & (w'_j[b'_\square])_j, a'_1[b'_\square], (u'_i[b'_\square])_i \end{array} \qquad \begin{array}{rl} & (w_j[b])_j, a_2[b], (v_i[b])_i \\ \sim & (w'_j[b'])_j, a'_2[b'], (v'_i[b'])_i \end{array}}{\begin{array}{rl} & \Big(\mathsf{if}\ \boxed{a_1[b_\square]}\boxed{a_2[b]}_a\ \mathsf{then}\ w_j[b_\square]\ \mathsf{else}\ w_j[b]\Big)_j, \Big(\mathsf{if}\ \boxed{a_1[b_\square]}\boxed{a_2[b]}_a\ \mathsf{then}\ u_i[b_\square]\ \mathsf{else}\ v_i[b]\Big)_i \\ \sim & \Big(\mathsf{if}\ \boxed{a'_1[b'_\square]}\boxed{a'_2[b']}_{a'}\ \mathsf{then}\ w'_j[b'_\square]\ \mathsf{else}\ w'_j[b']\Big)_j, \Big(\mathsf{if}\ \boxed{a'_1[b'_\square]}\boxed{a'_2[b']}_{a'}\ \mathsf{then}\ u'_i[b'_\square]\ \mathsf{else}\ v'_i[b']\Big)_i \end{array}}\,\mathsf{CS}_\square}{\begin{array}{rl} & \Big(\mathsf{if}\ \boxed{a_1[b]}\boxed{a_2[b]}_a\ \mathsf{then}\ w_j[b]\ \mathsf{else}\ w_j[b]\Big)_j, \Big(\mathsf{if}\ \boxed{a_1[b]}\boxed{a_2[b]}_a\ \mathsf{then}\ u_i[b]\ \mathsf{else}\ v_i[b]\Big)_i \\ \sim & \Big(\mathsf{if}\ \boxed{a'_1[b']}\boxed{a'_2[b']}_{a'}\ \mathsf{then}\ w'_j[b']\ \mathsf{else}\ w'_j[b']\Big)_j, \Big(\mathsf{if}\ \boxed{a'_1[b']}\boxed{a'_2[b']}_{a'}\ \mathsf{then}\ u'_i[b']\ \mathsf{else}\ v'_i[b']\Big)_i \end{array}}\,\mathsf{2Box}}{\begin{array}{rl} & (w_j[b])_j, \Big(\mathsf{if}\ \boxed{a_1[b]}\boxed{a_2[b]}_a\ \mathsf{then}\ u_i[b]\ \mathsf{else}\ v_i[b]\Big)_i \\ \sim & (w'_j[b'])_j, \Big(\mathsf{if}\ \boxed{a'_1[b']}\boxed{a'_2[b']}_{a'}\ \mathsf{then}\ u'_i[b']\ \mathsf{else}\ v'_i[b']\Big)_i \end{array}}\,R_\square$$

The commutation with an application of 2Box in the right branch is exactly the same. ■

C. Restr Elimination

We show in the following lemma that any proof using Restr can be rewritten into a (no larger) proof without the Restr rule. In other word, the Restr rule is admissible in our logic. Remark that this Restr elimination result subsumes Lemma 1.

Lemma 12 (Restr Elimination)**.** *If $P \vdash \vec{u} \sim \vec{v}$ with P in $(\mathsf{CS}_\square + R + \mathsf{2Box} + \mathsf{FA} + \mathsf{Dup} + \mathsf{CCA2} + \mathsf{Restr})^*$ then there exists P' such that $P' \vdash \vec{u} \sim \vec{v}$ and P' contains no* Restr *applications. Moreover the height of P' is no larger than the height of P.*

Proof. We do a proof by induction on the height of the derivation P of $\vec{u} \sim \vec{v}$. For the inductive case, assume that we have a derivation P of $\vec{u} \sim \vec{v}$ where the last rule applied is Restr:

$$\dfrac{\vec{u}, \vec{t} \sim \vec{v}, \vec{t}}{\vec{u} \sim \vec{v}}\,\mathsf{Restr}$$

We discriminate on the second last rule applied:

- If it is a unitary axiom we conclude easily using the fact that unitary axioms are closed under Restr.
- If it is a FA axiom and $\vec{t}$ is not involved in this function application then P is of the form:

$$\dfrac{\dfrac{P_0}{f(\vec{u}), \vec{u}', \vec{t} \sim f(\vec{v}), \vec{v}', \vec{t}}\,\mathsf{FA}}{f(\vec{u}), \vec{u}' \sim f(\vec{v}), \vec{v}'}\,\mathsf{Restr} \qquad \wedge \qquad P_0 \vdash \vec{u}, \vec{u}', \vec{t} \sim \vec{v}, \vec{v}', \vec{t}$$

  By applying the induction hypothesis on the following derivation:

$$\dfrac{P_0}{\vec{u}, \vec{u}' \sim \vec{v}, \vec{v}'}\,\mathsf{Restr}$$

  we have a derivation $P' \vdash \vec{u}, \vec{u}' \sim \vec{v}, \vec{v}'$ in the wanted fragment. We conclude by applying the FA rule: $\dfrac{P'}{f(\vec{u}), \vec{u}' \sim f(\vec{v}), \vec{v}'}\,\mathsf{FA}$.
- If it is a FA axiom and $\vec{t}$ is involved in this function application then P is of the form:

$$\dfrac{\dfrac{P_0}{\vec{u}, \vec{u}', f(\vec{u}'') \sim \vec{v}, \vec{v}', f(\vec{v}'')}\,\mathsf{FA}}{\vec{u} \sim \vec{v}}\,\mathsf{Restr} \qquad \wedge \qquad P_0 \vdash \vec{u}, \vec{u}', \vec{u}'' \sim \vec{v}, \vec{v}', \vec{v}''$$

  By applying the induction hypothesis on the following derivation:

$$\dfrac{P_0}{\vec{u} \sim \vec{v}}\,\mathsf{Restr}$$

  We get a derivation $P' \vdash \vec{u} \sim \vec{v}$ in the wanted fragment.
- The $\mathsf{CS}_\square$ axiom is handled similarly to FA.
- The Dup, 2Box and R axioms are trivial to handle. ■

a) Sub-Proof Extraction Functions $\mathsf{extract_l}$ *and* $\mathsf{extract_r}$*:* It follows that, given a proof $P \vdash \vec{u} \sim \vec{v}$ and a position h in the proof P such that:

$$P_{|h} = \dfrac{\vec{w}, b_1, (u_i)_i \sim \vec{w}', b_1', (u_i')_i \qquad \vec{w}, b_2, (v_i)_i \sim \vec{w}', b_2', (v_i')_i}{\vec{w}, \left(\mathsf{if}\ \boxed{b_1 \mid b_2}_b\ \mathsf{then}\ u_i\ \mathsf{else}\ v_i\right)_i \sim \vec{w}', \left(\mathsf{if}\ \boxed{b_1' \mid b_2'}_{b'}\ \mathsf{then}\ u_i'\ \mathsf{else}\ v_i'\right)_i}\ \mathsf{CS}_\square$$

we can extract from P the left (resp. right) proof of $b_1 \sim b_1'$ (resp. $b_2 \sim b_2'$) using the Restr elimination procedure described in the proof of Lemma 12. We let $\mathsf{extract_l}(h, P)$ be proof of $b_1 \sim b_1'$ extracted from $P_{|h}$, and $\mathsf{extract_r}(h, P)$ be proof of $b_2 \sim b_2'$ extracted from $P_{|h}$.

D. Completeness of the Freeze Strategy

We give here a proof of Lemma 7, which we recall below.

Lemma (7)**.** *Let* U *be a set of unitary axioms closed under* Restr*. Then the following strategy:*

$$\mathfrak{F}\big((\mathsf{2Box} + R_\square)^* \cdot \mathsf{CS}_\square^* \cdot \{\overline{\mathsf{BFA}}(b, b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{U}\big)$$

is complete for $\mathfrak{F}(\mathsf{CS} + \mathsf{FA} + R + \mathsf{Dup} + \mathsf{U})$.

Before starting the proof, we need to define the induction ordering.

a) Proof ordering: Let us consider the following well-founded order on proofs: a proof is interpreted by the multi-set of pair (b, b') appearing as (potentially frozen) labels of BFA applications where we erased the function symbol $\overline{\ }$. We then order these multi-set using the multi-set ordering $\succ_{\text{mult}}$, which is induced by the product ordering $\succ_\times$, which itself is built upon an arbitrary total rewrite ordering on ground terms without boxes $\succ$ (e.g a LPO for some arbitrary precedence over function symbols).

b) Example: Assume that $b_1 \equiv \mathsf{if}\ b\ \mathsf{then}\ a\ \mathsf{else}\ c$ and $b_2 \equiv \mathsf{if}\ b'\ \mathsf{then}\ a'\ \mathsf{else}\ c'$. We let P_1 be the derivation:

$$\dfrac{\dfrac{\overline{b}, a, c, u_1, v_1 \sim \vec{x}', \overline{b'}, a', c', u_1, v_1}{\widetilde{b_1}, u_1, v_1 \sim \widetilde{b_2}, u_2, v_2}\ \mathsf{BFA}(\overline{b}, \overline{b'})}{\mathsf{if}\ b_1\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \sim \mathsf{if}\ b_2\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2}\ \mathsf{BFA}(b_1, b_2)$$

And P_2 be the derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\widetilde{b}, \widetilde{a}, \widetilde{c}, u_1, v_1 \sim \widetilde{b'}, \widetilde{a'}, \widetilde{c'}, u_2, v_2}{\widetilde{b}, \widetilde{a}, u_1, v_1, \widetilde{c}, u_1, v_1 \sim \widetilde{b'}, \widetilde{a'}, u_2, u_2, \widetilde{c'}, u_2, v_2}\ \mathsf{Dup}}{\widetilde{b}, \widetilde{a}, u_1, v_1, \mathsf{if}\ c\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \sim \widetilde{b'}, \widetilde{a'}, u_2, u_2, \mathsf{if}\ c'\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2}\ \mathsf{BFA}(c, c')}{\widetilde{b}, \mathsf{if}\ a\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1, \mathsf{if}\ c\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \sim \widetilde{b'}, \mathsf{if}\ a'\ \mathsf{then}\ u_2\ \mathsf{else}\ u_2, \mathsf{if}\ c'\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2}\ \mathsf{BFA}(a, a')}{\mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1)\ \mathsf{else}\ (\mathsf{if}\ c\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1) \sim \mathsf{if}\ b'\ \mathsf{then}\ (\mathsf{if}\ a'\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2)\ \mathsf{else}\ (\mathsf{if}\ c'\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2)}\ \mathsf{BFA}(b, b')}{\mathsf{if}\ b_1\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \sim \mathsf{if}\ b_2\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2}\ R$$

P_1 and P_2 are respectively interpreted as the multi-sets $\{(b_1, b_2), (b, b')\}$ and $\{(b, b'), (a, a'), (c, c')\}$ (observe that we unfroze the conditionals). b, a, c (resp. b', a', c') are strict subterms of b_1 (resp. b_2), therefore we have $(b_1, b_2) \succ_\times (b, b')$, $(b_1, b_2) \succ_\times (a, a')$ and $(b_1, b_2) \succ_\times (c, c')$. Therefore we have:

$$\{(b_1, b_2), (b, b')\} \succ_{\text{mult}} \{(b, b'), (a, a'), (c, c')\}$$

By consequence P_2 is a smaller proof of $\mathsf{if}\ b_1\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \sim \mathsf{if}\ b_2\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2$ than P_1.

Proof of Lemma 7. First we are going to show a cut elimination strategy to get rid of the deconstruction of frozen conditionals introduced by:

$$\dfrac{\vec{w}_1, \widetilde{b_1}, u_1', v_1' \sim \vec{w}_2, \widetilde{b_2}, u_2', v_2'}{\vec{w}_1, \mathsf{if}\ b_1\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \sim \vec{w}_2, \mathsf{if}\ b_2\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2}\ \mathsf{BFA}(b_1, b_2)$$

Assume now that $u \sim v$ is not provable without deconstructing frozen conditionals introduced as described above. We consider a proof P_1 of $u \sim v$ that we suppose minimal for $\succ_{\text{mult}}$. We are going to consider the first conditionals (b_1, b_2)

(starting from the bottom) which are deconstructed. We let $b_1 \equiv \mathsf{if}\ b\ \mathsf{then}\ a\ \mathsf{else}\ c$ and $b_2 \equiv \mathsf{if}\ b'\ \mathsf{then}\ a'\ \mathsf{else}\ c'$, we know that our proof has the following shape:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\ (A_3)\\ \vec{x}, \overline{b}, a, c, \vec{y} \sim \vec{x}', \overline{b'}, a', c', \vec{y}'\end{array}}{\vec{x}, \widetilde{b_1}, \vec{y} \sim \vec{x}', \widetilde{b_2}, \vec{y}'}\ \mathsf{BFA}(\overline{b}, \overline{b'})}{\begin{array}{c}\vdots\ (A_2)\\ \vec{w}_1, \widetilde{b_1}, u_1, v_1 \sim \vec{w}_2, \widetilde{b_2}, u_2, v_2\end{array}}}{\vec{w}_1, \mathsf{if}\ b_1\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \sim \vec{w}_2, \mathsf{if}\ b_2\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2}\ \mathsf{BFA}(b_1, b_2)}{\begin{array}{c}\vdots\ (A_1)\\ \dfrac{C[\mathsf{if}\ b_1\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1] \sim C[\mathsf{if}\ b_2\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2]}{u \sim v}\ R\end{array}}
$$

Where C is a one-hole context. Since (b_1, b_2) are the first conditionals deconstructed in this proof we know that C is such that the hole does not appear in a conditional branch. This proof can be rewritten as the following proof P_2:

$$
\begin{array}{c}
\vdots\ (A_3)\\
\vec{x}, \widetilde{b}, \widetilde{a}, \widetilde{c}, \vec{y} \sim \vec{x}', \widetilde{b'}, \widetilde{a'}, \widetilde{c'}, \vec{y}'\\
\vdots\ (A_2)\\
\dfrac{\dfrac{\dfrac{\dfrac{\vec{w}_1, \widetilde{b}, \widetilde{a}, \widetilde{c}, u_1, v_1 \sim \vec{w}_2, \widetilde{b'}, \widetilde{a'}, \widetilde{c'}, u_2, v_2}{\vec{w}_1, \widetilde{b}, \widetilde{a}, u_1, v_1, \widetilde{c}, u_1, v_1 \sim \vec{w}_2, \widetilde{b'}, \widetilde{a'}, u_2, u_2, \widetilde{c'}, u_2, v_2}\ \mathsf{Dup}}{\vec{w}_1, \widetilde{b}, \widetilde{a}, u_1, v_1, \mathsf{if}\ c\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \sim \vec{w}_2, \widetilde{b'}, \widetilde{a'}, u_2, u_2, \mathsf{if}\ c'\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2}\ \mathsf{BFA}(c, c')}{\vec{w}_1, \widetilde{b}, \mathsf{if}\ a\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1, \mathsf{if}\ c\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \sim \vec{w}_2, \widetilde{b'}, \mathsf{if}\ a'\ \mathsf{then}\ u_2\ \mathsf{else}\ u_2, \mathsf{if}\ c'\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2}\ \mathsf{BFA}(a, a')}{\vec{w}_1, \mathsf{if}\ b\ \begin{array}{l}\mathsf{then\ if}\ a\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1\\ \mathsf{else\ if}\ c\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1\end{array} \sim \vec{w}_2, \mathsf{if}\ b'\ \begin{array}{l}\mathsf{then\ if}\ a'\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2\\ \mathsf{else\ if}\ c'\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2\end{array}}\ \mathsf{BFA}(b, b')\\
\vdots\ (A_1)\\
\dfrac{\dfrac{C[\mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1)\ \mathsf{else}\ (\mathsf{if}\ c\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1)] \sim C[\mathsf{if}\ b'\ \mathsf{then}\ (\mathsf{if}\ a'\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2)\ \mathsf{else}\ (\mathsf{if}\ c'\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2)]}{C[\mathsf{if}\ b_1\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1] \sim C[\mathsf{if}\ b_2\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2]}\ R}{u \sim v}\ R
\end{array}
$$

One can check that A_1 remains the same in the second proof tree since the hole in C is not in a conditional branch.

The A_1, A_2, A_3 parts are the same in both proofs, so let M be the interpretation of A_1, A_2, A_3 as a multi-set. Then the interpretation of P_1 (resp. P_2) is $M \cup \{(b_1, b_2), (b, b')\}$ (resp. $M \cup \{(b, b'), (a, a'), (c, c')\}$). Therefore P_2 is a strictly smaller proof of $u \sim v$ than P_1 (this is almost the same multi-sets than in the example above). Absurd. ■

APPENDIX IV
PROOF FORM

In this section, we define what are the early proof form and the normal proof form. This is rather technical and lengthy, as the definition of normal proof form relies on four mutually recursive definitions: $\mathcal{S}_l$-*encryption oracle calls* are well-formed encryptions; $\mathcal{S}_l$-*decryption oracle calls* are well-formed decryptions; $\mathcal{S}_l$-*normalized basic terms* are terms built using well-formed encryptions and decryptions as well as function symbols different from $\mathsf{if\_then\_else\_}$; and $\mathcal{S}_l$-*normalized simple terms* are combinations of normalized basic terms using $\mathsf{if\_then\_else\_}$.

We then show Lemma 8, which is a weak normalization result: it describes a procedure that, given a proof P of $\vec{u} \sim \vec{v}$ following the ordered freeze strategy of Lemma 7, computes a proof P' of $\vec{u} \sim \vec{v}$ such that P' is in normal proof form. This procedure is a careful bottom-up rewriting of all the sub-terms appearing in P.

We also give a proof of Lemma 2.

A. Early Proof Form

We showed in Lemma 7 that:

$$(\mathsf{2Box} + R_\Box)^* \cdot \mathsf{CS}_\Box^* \cdot \{\overline{\mathsf{BFA}}(b, b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2} \qquad (\mathcal{A}_\succ)$$

is complete for $\mathsf{CS} + \mathsf{FA} + R + \mathsf{Dup} + \mathsf{CCA2}$. Let us consider a proof P following this ordering. From now on we will use $\mathcal{A}_\succ$ to denote this fragment. Moreover we let $\mathcal{A}_{\mathsf{CS}_\Box}$ and $\mathcal{A}_{\overline{\mathsf{BFA}}}$ be, respectively, the fragments:

$$\mathsf{CS}_\Box^* \cdot \{\overline{\mathsf{BFA}}(b, b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2} \qquad (\mathcal{A}_{\mathsf{CS}_\Box})$$

$$\{\overline{\mathsf{BFA}}(b, b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2} \qquad (\mathcal{A}_{\overline{\mathsf{BFA}}})$$

The only branching rule is the $\mathsf{CS}_\Box$ rule, which has two premises. Hence after having completed all the $\mathsf{CS}_\Box$ applications we know that the proof will be non-branching and in $\mathcal{A}_{\overline{\mathsf{BFA}}}$. We want to name each branch of the proof tree, and its corresponding instance of the $\mathsf{CCA2}$ axiom. To do so, we index each branch of the proof tree P by some $l \in L$ where L is a set of labels, and we let $\vdash^{\mathsf{b}}$ be the proof system $\vdash$ with branch annotations. When $P \vdash^{\mathsf{b}} t \sim t'$, we let $\mathsf{label}(P)$ be the set of labels L annotating the branches in P, and for all $l \in L$, we let $\mathsf{instance}(P, l)$ be the instance of $\mathsf{CCA2}$ obtained using Proposition 9 from the instance of $\mathsf{CCA2}$ used in branch l:

$$\mathsf{instance}(P, l) = \frac{}{\vec{w}, (\alpha_i)_{i \in I}, (\mathsf{dec}_j)_{j \in J} \sim \vec{w}, (\alpha'_i)_{i \in I}, (\mathsf{dec}'_j)_{j \in J}}\ \mathsf{CCA2}$$

We also define $\mathcal{E}_l^P = \{\alpha_i \mid i\}$, $\mathcal{D}_l^P = \{\mathsf{dec}_j \mid j \in J\}$ and $\mathcal{K}_l^P$ to be the sets of, respectively, encryptions, decryptions and keys used in the $\mathsf{CCA2}$ application of the branch l of proof P, on the left side. Similarly we define $\mathcal{E}_l'^P$, $\mathcal{D}_l'^P$ and $\mathcal{K}_l'^P$ for the right side.

Definition 17. *For all terms t, t' and proofs P such that $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\Box}} t \sim t'$, we say that P proof in* early proof form *if t and t' are of the following form:*

$$t \equiv C\left[\left(\boxed{b^{h_l}}\boxed{b^{h_r}}_{\,b^h}\right)_{h \in H} \diamond (u_l)_{l \in \mathsf{label}(P)}\right] \qquad \wedge \qquad t' \equiv C\left[\left(\boxed{b'^{h_l}}\boxed{b'^{h_r}}_{\,b'^h}\right)_{h \in H} \diamond (u'_l)_{l \in \mathsf{label}(P)}\right]$$

where H is a set of positions in P (we let $\mathsf{cs\text{-}pos}(P) \equiv H$) such that:

- *for all $h \in H$, the rule applied at position h in P is a $\mathsf{CS}_\Box$ rule on the conditionals:*

$$\left(\boxed{b^{h_l}}\boxed{b^{h_r}}_{\,b^h}, \boxed{b'^{h_l}}\boxed{b'^{h_r}}_{\,b'^h}\right)$$

- *$(b^h)_{h \in H}$ are if-free conditionals in R-normal form and for all $h \in H, b^{h_l} =_R b^{h_r} =_R b^h$ (same for b'^{h_l}, b'^{h_r}, b^h).*
- *Let $P^{h_l} = \mathsf{extract_l}(h, P)$ and $P^{h_r} = \mathsf{extract_r}(h, P)$, then:*

$$P^{h_l} \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\Box}} b^{h_l} \sim b'^{h_l} \qquad \wedge \qquad P^{h_r} \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\Box}} b^{h_r} \sim b'^{h_r}$$

  and these two proofs are in early proof form.
- *$\mathsf{label}(P^{h_l}) \subseteq \mathsf{label}(P)$, and for all $l \in \mathsf{label}(P^{h_l})$, $\mathsf{instance}(P^{h_l}, l)$ is subsumed by $\mathsf{instance}(P, l)$ (same for $\mathsf{label}(P^{h_r})$).*
- *For all $l \in \mathsf{label}(P)$, we know that the extraction from P of the sub-proof of $u_l \sim u'_l$ is in the fragment $\mathcal{A}_{\overline{\mathsf{BFA}}}$.*

Proposition 11. *For all terms t, t' and proofs P such that $P \vdash_{\mathcal{A}_{\mathsf{CS}_\Box}} t \sim t'$, there exists a labelling P' of P such that $P' \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\Box}} t \sim t'$ and P' is in early proof form.*

Proof. We can check that the proof P has the wanted shape and is properly labelled by induction on the size of the proof, by observing that for all $h \in \mathsf{cs\text{-}pos}(P)$ and $\mathsf{x} \in \{\mathsf{l}, \mathsf{r}\}$, $\mathsf{extract_x}(h, P)$ is of size strictly smaller that P. We only need to perform some α-renaming to have the labelling of the sub-proofs coincide. $\square$

Finally we can check that the resulting proof Q is such that for all $h \in \mathsf{cs\text{-}pos}(Q), \mathsf{x} \in \{\mathsf{l}, \mathsf{r}\}$, for all $l \in \mathsf{label}(\mathsf{extract}_{\mathsf{x}}(h, P))$, the CCA2 instance $\mathsf{instance}(\mathsf{extract}_{\mathsf{x}}(h, P), l)$ is subsumed by $\mathsf{instance}(P, l)$. This follows from the fact that $\mathsf{extract}_{\mathsf{x}}(h, P)$ is obtained through the Restr elimination procedure from P. ■

We define below the set $\mathsf{index}(P)$ of positions of P, which is the set of *all* positions of P where a $\mathsf{CS}_{\Box}$ rule is applied. This set is naturally ordered using the prefix ordering on positions. Moreover we can define the "depth" of a position h in P to be, intuitively, the number of nested applications of the $\mathsf{CS}_{\Box}$ rule.

Definition 18. *Let $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_{\Box}}} t \sim t'$ in early proof form.*

- *We let $\mathsf{index}(P)$ be the set of indices where $\mathsf{CS}_{\Box}$ rules occur in the proof P:*

$$\mathsf{index}(P) = \mathsf{cs\text{-}pos}(P) \cup \left(\bigcup_{h \in \mathsf{cs\text{-}pos}(P)} \mathsf{index}\left(\mathsf{extract}_{\mathsf{l}}(h, P)\right) \cup \mathsf{index}\left(\mathsf{extract}_{\mathsf{r}}(h, P)\right) \right)$$

- *For all $h, h' \in \mathsf{index}(t, P)$, we let $<$ be the ancestor relation, defined by $h < h'$ if and only if h is a prefix of h'.*
- *For all $h \in \mathsf{index}(P)$, we let $\mathsf{if\text{-}depth}_P(h)$ be the depth of h in P, defined as follows:*

$$\mathsf{if\text{-}depth}_P(h) = \begin{cases} 0 & \text{if } h \in \mathsf{cs\text{-}pos}(P) \\ 1 + \mathsf{if\text{-}depth}_{P^{\mathsf{l}}}(h) & \text{if } \exists g \in \mathsf{cs\text{-}pos}(P) \text{ such that } h \in \mathsf{index}(\mathsf{extract}_{\mathsf{l}}(g, P)) \\ 1 + \mathsf{if\text{-}depth}_{P^{\mathsf{r}}}(h) & \text{if } \exists g \in \mathsf{cs\text{-}pos}(P) \text{ such that } h \in \mathsf{index}(\mathsf{extract}_{\mathsf{r}}(g, P)) \end{cases}$$

For all $\mathsf{h} = h_{\mathsf{x}}$, where $h \in \mathsf{index}(P)$ and $\mathsf{x} \in \{\mathsf{l}, \mathsf{r}\}$, we let $\mathsf{cs\text{-}pos}_P(\mathsf{h}) = \mathsf{cs\text{-}pos}(\mathsf{extract}_{\mathsf{x}}(h, P))$. When there is no ambiguity on the proof P, we write $\mathsf{cs\text{-}pos}(\mathsf{h})$ instead of $\mathsf{cs\text{-}pos}_P(\mathsf{h})$.

Definition 19. *Let $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_{\Box}}} t \sim t'$ in early proof form. For all $l \in \mathsf{label}(P)$, we define:*

$$\mathsf{h\text{-}branch}(l) = \{h_{\mathsf{x}} \mid h \in \mathsf{index}(P) \wedge \mathsf{x} \in \{\mathsf{l}, \mathsf{r}\} \wedge l \in \mathsf{label}(\mathsf{extract}_{\mathsf{x}}(h, P))\} \cup \{\epsilon\}$$

We abuse the notation and say that $h \in \mathsf{h\text{-}branch}(l)$ if there exists $\mathsf{x} \in \{\mathsf{l}, \mathsf{r}\}$ such that $h_{\mathsf{x}} \in \mathsf{h\text{-}branch}(l)$. In that case, we say that x is the direction taken at h in l.

Morally, $\mathsf{h\text{-}branch}(l)$ is the set of positions of P where a $\mathsf{CS}_{\Box}$ rule is applied on a given branch. Of course for all $l \in \mathsf{label}(P)$, $\epsilon \in \mathsf{h\text{-}branch}(l)$ since ϵ is the index of the toplevel proof P.

B. Shape of the Terms

For all proofs in $\mathcal{A}_{\succ}$, all R rewritings are done at the beginning of the proofs in the $(\mathsf{2Box} + R_{\Box})^*$ part, and, afterwards, all rules (apart from Dup) only "peel off" terms by removing the top-most function symbol. Therefore the terms just after $(\mathsf{2Box} + R_{\Box})^*$ characterize the shape of the subsequent proof. This observation is illustrated in Fig. 9. Recall that for all $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_{\Box}}} t \sim t'$ in early proof form, we have:

$$t \equiv C\left[\left(\boxed{b^{h_{\mathsf{l}}}}\boxed{b^{h_{\mathsf{r}}}}_{\,b^h}\right)_{h \in H} \diamond (u_l)_{l \in \mathsf{label}(P)}\right] \qquad \wedge \qquad t' \equiv C\left[\left(\boxed{b'^{h_{\mathsf{l}}}}\boxed{b'^{h_{\mathsf{r}}}}_{\,b'^h}\right)_{h \in H} \diamond (u'_l)_{l \in \mathsf{label}(P)}\right]$$

where for all $l \in \mathsf{label}(P)$, the extraction from P of the sub-proof of $u_l \sim u'_l$ is in the fragment $\mathcal{A}_{\overline{\mathsf{BFA}}}$. This means that for all l:

$$u_l \equiv D_l\left[\left(B_{i,l}[\vec{w}_{i,l}, (\alpha^j_{i,l})_{j \in J^0_{i,l}}, (\mathsf{dec}^k_{i,l})_{K^0_{i,l}}]\right)_{i \in I} \diamond \left(U_{m,l}[\vec{w}_{m,l}, (\alpha^j_{m,l})_{j \in J^1_{i,l}}, (\mathsf{dec}^k_{m,l})_{k \in K^1_{i,l}}]\right)_{m \in M}\right]$$
$$u'_l \equiv D_l\left[\left(B_{i,l}[\vec{w}_{i,l}, (\alpha'^j_{i,l})_{j \in J^0_{i,l}}, (\mathsf{dec}'^k_{i,l})_{K^0_{i,l}}]\right)_{i \in I} \diamond \left(U_{m,l}[\vec{w}_{m,l}, (\alpha'^j_{m,l})_{j \in J^1_{i,l}}, (\mathsf{dec}'^k_{m,l})_{k \in K^1_{i,l}}]\right)_{m \in M}\right]$$

where D_l is an if-context, $(B_{i,l})_i$ and $(U_{m,l})_m$ are if-free contexts, the encryptions appear in $\mathcal{E}^P_l$:

$$\left\{\alpha^j_{i,l} \mid i \in I, j \in J^0_{i,l}\right\} \cup \left\{\alpha^j_{m,l} \mid m \in M, j \in J^0_{m,l}\right\} \subseteq \mathcal{E}^P_l$$
$$\left\{\alpha'^j_{i,l} \mid i \in I, j \in J^0_{i,l}\right\} \cup \left\{\alpha'^j_{m,l} \mid m \in M, j \in J^0_{m,l}\right\} \subseteq \mathcal{E}'^P_l$$

and the decryptions appear in $\mathcal{D}^P_l$:

$$\left\{\mathsf{dec}^k_{i,l} \mid i \in I, k \in K^0_{i,l}\right\} \cup \left\{\mathsf{dec}^k_{m,l} \mid m \in M, k \in K^0_{m,l}\right\} \subseteq \mathcal{D}^P_l$$
$$\left\{\mathsf{dec}'^k_{i,l} \mid i \in I, k \in K^0_{i,l}\right\} \cup \left\{\mathsf{dec}'^k_{m,l} \mid m \in M, k \in K^0_{m,l}\right\} \subseteq \mathcal{D}'^P_l$$

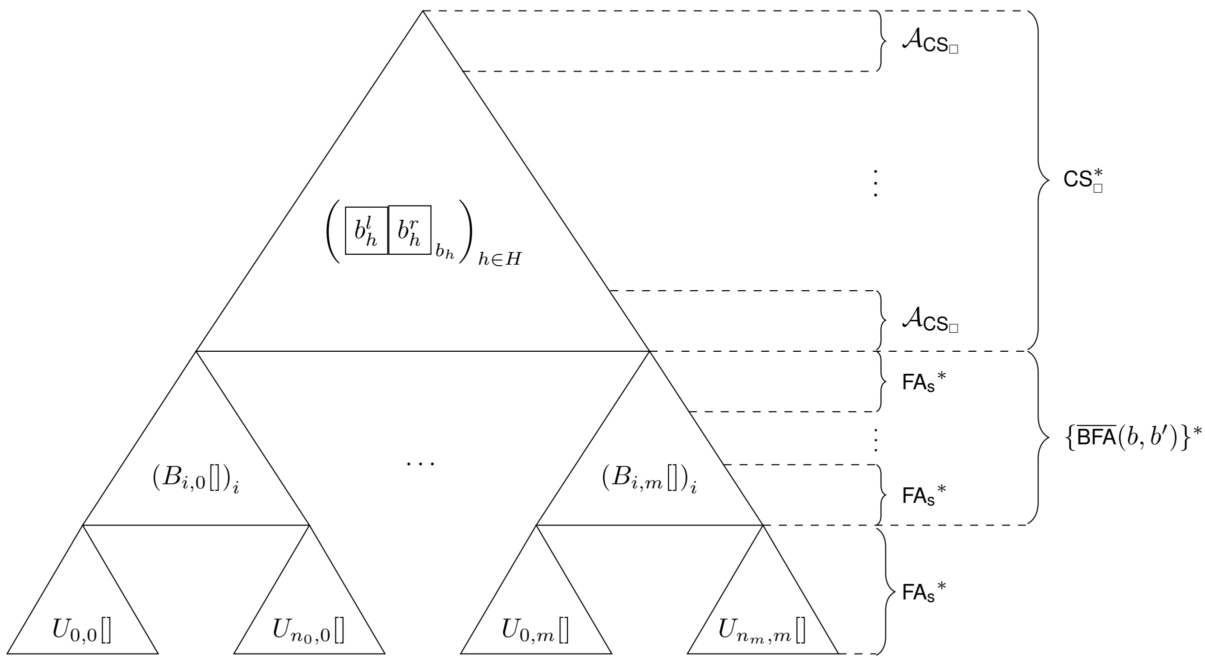


Fig. 9. The shape of the term is determined by the proof.

Using these notation, we give some definitions:

Definition 20. *Let $P \vdash^{b}_{\mathcal{A}_{\mathsf{CS}_\square}} t \sim t'$. Then for all $l \in \mathsf{label}(P)$, we define the following relations:*

- $(b, b') \leq^{\epsilon,l}_{\mathsf{cs}\sim\mathsf{cs}} (t \sim t', P)$ *(resp. $b \leq^{\epsilon,l}_{\mathsf{cs}} (t, P)$, $b' \leq^{\epsilon,l}_{\mathsf{cs}} (t', P)$) if and only if there exists $h_0 \in \mathsf{index}(P)$ such that:*

$$b \equiv b^{h_0} \quad \wedge \quad b' \equiv b'^{h_0}$$

- $(\beta, \beta') \leq^{\epsilon,l}_{\mathsf{c}\sim\mathsf{c}} (t \sim t', P)$ *(resp. $\beta \leq^{\epsilon,l}_{\mathsf{c}} (t, P)$, $\beta' \leq^{\epsilon,l}_{\mathsf{c}} (t', P)$) if and only if there exists $i \in I$ such that:*

$$\beta \equiv B_{i,l}[\vec{w}_{i,l}, (\alpha^j_{i,l})_{j\in J^0_{i,l}}, (\mathsf{dec}^k_{i,l})_{k\in K^0_{i,l}}] \quad \wedge \quad \beta' \equiv B_{i,l}[\vec{w}_{i,l}, (\alpha'^j_{i,l})_{j\in J^0_{i,l}}, (\mathsf{dec}'^k_{i,l})_{k\in K^0_{i,l}}]$$

- $(\gamma, \gamma') \leq^{\epsilon,l}_{\mathsf{l}\sim\mathsf{l}} (t \sim t', P)$ *(resp. $\gamma \leq^{\epsilon,l}_{\mathsf{l}} (t, P)$, $\gamma' \leq^{\epsilon,l}_{\mathsf{l}} (t', P)$) if and only if there exists $m \in M$ such that:*

$$\gamma \equiv U_{m,l}[\vec{w}_{m,l}, (\alpha^j_{m,l})_{j\in J^1_{i,l}}, (\mathsf{dec}^k_{m,l})_{k\in K^1_{i,l}}] \quad \wedge \quad \gamma' \equiv U_{m,l}[\vec{w}_{m,l}, (\alpha'^j_{m,l})_{j\in J^1_{i,l}}, (\mathsf{dec}'^k_{m,l})_{k\in K^1_{i,l}}]$$

Definition 21. *Let $P \vdash^{b}_{\mathcal{A}_{\mathsf{CS}_\square}} t \sim t'$ in early proof form. For all $h \in \mathsf{index}(P)$, $\mathsf{x} \in \{\mathsf{l}, \mathsf{r}\}$:*

- *For all $\Delta \in \{\mathsf{c}\sim\mathsf{c}, \mathsf{l}\sim\mathsf{l}, \mathsf{cs}\sim\mathsf{cs}\}$, we define $\leq^{h_{\mathsf{x}},l}_{\Delta} (t \sim t', P)$ as follows:*

$$\forall s, s'.\, (s, s') \leq^{h_{\mathsf{x}},l}_{\Delta} (t \sim t', P) \qquad \textit{if and only if} \qquad (s, s') \leq^{\epsilon,l}_{\Delta} (b \sim b', \mathsf{extract}_{\mathsf{x}}(h, P))$$

where $\mathsf{extract}_{\mathsf{x}}(h, P)$ is a proof of $b \sim b'$.

- *For all $\Delta \in \{\mathsf{c}, \mathsf{l}, \mathsf{cs}\}$, we define $\leq^{h_{\mathsf{x}},l}_{\Delta} (t, P)$ as follows:*

$$\forall s.\, s \leq^{h_{\mathsf{x}},l}_{\Delta} (t, P) \qquad \textit{if and only if} \qquad s \leq^{\epsilon,l}_{\Delta} (b, \mathsf{extract}_{\mathsf{x}}(h, P))$$

where $\mathsf{extract}_{\mathsf{x}}(h, P)$ is a proof of $b \sim b'$.

Remark 7. We extend these notations to proofs P such that $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\succ}} t \sim t'$. Let P' be such that:

$$P \equiv \frac{P'}{t \sim t'}\ (\mathsf{2Box} + R_\square)^*$$

and $P' \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\square}} t_0 \sim t'_0$, then $(s, s') \leq^{\mathsf{h},l}_{\Delta} (t \sim t', P)$ if and only if $(s, s') \leq^{\mathsf{h},l}_{\Delta} (t_0 \sim t'_0, P')$ where $\Delta \in \{\mathsf{c}\sim\mathsf{c}, \mathsf{l}\sim\mathsf{l}, \mathsf{cs}\sim\mathsf{cs}\}$. Similarly $s \leq^{\mathsf{h},l}_{\Delta} (t, P)$ if and only if $s \leq^{\mathsf{h},l}_{\Delta} (t_0, P')$ where $\Delta \in \{\mathsf{c}, \mathsf{l}, \mathsf{cs}\}$.

Extending these notations to $B^{\mathsf{h}}_l[], U^{\mathsf{h}}_l \dots$, we describe the shape of a complete proof in Fig. 10.

$$
\frac{}{\left(\left(\vec{w}^{\mathsf{h}}_{i,l}, (\alpha^{\mathsf{h},j}_{i,l})_j, (\mathsf{dec}^{\mathsf{h},k}_{i,l})_k\right)_i, \left(\vec{w}^{\mathsf{h}}_{m,l}, (\alpha^{\mathsf{h},j}_{m,l})_j, (\mathsf{dec}^{\mathsf{h},k}_{m,l})_k\right)_m\right)_{\mathsf{h}\in\mathsf{h\text{-}branch}(l)} \sim \left(\left(\vec{w}^{\mathsf{h}}_{i,l}, (\alpha'^{\mathsf{h},j}_{i,l})_j, (\mathsf{dec}'^{\mathsf{h},k}_{i,l})_k\right)_i, \left(\vec{w}^{\mathsf{h}}_{m,l}, (\alpha'^{\mathsf{h},j}_{m,l})_j, (\mathsf{dec}'^{\mathsf{h},k}_{m,l})_k\right)_m\right)_{\mathsf{h}\in\mathsf{h\text{-}branch}(l)}}\ \overline{\mathsf{CCA2}}
$$

$$
\vdots\ \mathsf{FA_s}^* \cdot \mathsf{Dup}^*
$$

$$
\left(\left(B^{\mathsf{h}}_{i,l}[\vec{w}^{\mathsf{h}}_{i,l}, (\alpha^{\mathsf{h},j}_{i,l})_j, (\mathsf{dec}^{\mathsf{h},k}_{i,l})_k]\right)_i, \left(U^{\mathsf{h}}_{m,l}[\vec{w}^{\mathsf{h}}_{m,l}, (\alpha^{\mathsf{h},j}_{m,l})_j, (\mathsf{dec}^{\mathsf{h},k}_{m,l})_k]\right)_m\right)_{\mathsf{h}\in\mathsf{h\text{-}branch}(l)} \sim \left(\left(B^{\mathsf{h}}_{i,l}[\vec{w}^{\mathsf{h}}_{i,l}, (\alpha'^{\mathsf{h},j}_{i,l})_j, (\mathsf{dec}'^{\mathsf{h},k}_{i,l})_k]\right)_i, \left(U^{\mathsf{h}}_{m,l}[\vec{w}^{\mathsf{h}}_{m,l}, (\alpha'^{\mathsf{h},j}_{m,l})_j, (\mathsf{dec}'^{\mathsf{h},k}_{m,l})_k]\right)_m\right)_{\mathsf{h}\in\mathsf{h\text{-}branch}(l)}
$$

$$
\vdots\ \{\overline{\mathsf{BFA}}(b,b')\}^*
$$

$$
\left(D^{\mathsf{h}}_l\left[\left(B^{\mathsf{h}}_{i,l}[\vec{w}^{\mathsf{h}}_{i,l}, (\alpha^{\mathsf{h},j}_{i,l})_j, (\mathsf{dec}^{\mathsf{h},k}_{i,l})_k]\right)_i \diamond \left(U^{\mathsf{h}}_{m,l}[\vec{w}^{\mathsf{h}}_{m,l}, (\alpha^{\mathsf{h},j}_{m,l})_j, (\mathsf{dec}^{\mathsf{h},k}_{m,l})_k]\right)_m\right]\right)_{\mathsf{h}\in\mathsf{h\text{-}branch}(l)} \sim \left(D^{\mathsf{h}}_l\left[\left(B^{\mathsf{h}}_{i,l}[\vec{w}^{\mathsf{h}}_{i,l}, (\alpha'^{\mathsf{h},j}_{i,l})_j, (\mathsf{dec}'^{\mathsf{h},k}_{i,l})_k]\right)_i \diamond \left(U^{\mathsf{h}}_{m,l}[\vec{w}^{\mathsf{h}}_{m,l}, (\alpha'^{\mathsf{h},j}_{m,l})_j, (\mathsf{dec}'^{\mathsf{h},k}_{m,l})_k]\right)_m\right]\right)_{\mathsf{h}\in\mathsf{h\text{-}branch}(l)}
$$

$$
\forall l \in L,\quad \vdots\ \mathsf{CS}^*_{\square}
$$

$$
\frac{C\left[\left(\boxed{b^l_h}\,\boxed{b^r_h}_{\,b_h}\right)_h \diamond \left(D_l\left[\left(B_{i,l}[\vec{w}_{i,l}, (\alpha^j_{i,l})_j, (\mathsf{dec}^k_{i,l})_k]\right)_i \diamond \left(U_{m,l}[\vec{w}_{m,l}, (\alpha^j_{m,l})_j, (\mathsf{dec}^k_{m,l})_k]\right)_m\right]\right)_l\right] \sim C\left[\left(\boxed{b'^l_h}\,\boxed{b'^r_h}_{\,b'_h}\right)_h \diamond \left(D_l\left[\left(B_{i,l}[\vec{w}_{i,l}, (\alpha'^j_{i,l})_j, (\mathsf{dec}'^k_{i,l})_k]\right)_i \diamond \left(U_{m,l}[\vec{w}_{m,l}, (\alpha'^j_{m,l})_j, (\mathsf{dec}'^k_{m,l})_k]\right)_m\right]\right)_l\right]}{t \sim t'}
$$

Fig. 10. Shape of a full proof (for simplicity, we omitted the boxes in terms and related rules).

C. Simple Terms

A public/private key pair is valid if the same name has been used to generate the keys.

Definition 22. *A valid public/private key pair is a pair of terms $(\mathsf{pk}(\mathsf{n}), \mathsf{sk}(\mathsf{n}))$ where n is a name.*

We will now formally define the normal form for terms used in the strategy. This is done through four mutually inductive definitions: the normal forms of well-formed encryptions and of well-formed decryptions; the normal form of basic terms built using well-formed encryptions and decryptions, as well as function symbols different from if_then_else_; and finally the normal form of terms with conditionals.

The next step will be to prove that all intermediate terms in the proofs can be assumed to be in these normal forms. To keep the proof tractable, this will be done in two steps. Therefore we introduce two versions of some forms, e.g. we will define *simple terms* to be terms having a particular form, and *normalized simple terms* to be *simple terms* satisfying some further properties. Consider a instance of $\mathsf{CCA2}_a$:

$$
(\phi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}}) R^{\mathcal{K}}_{\mathsf{CCA2}_a}(\_,\_,\_,\_,\_,\_)
$$

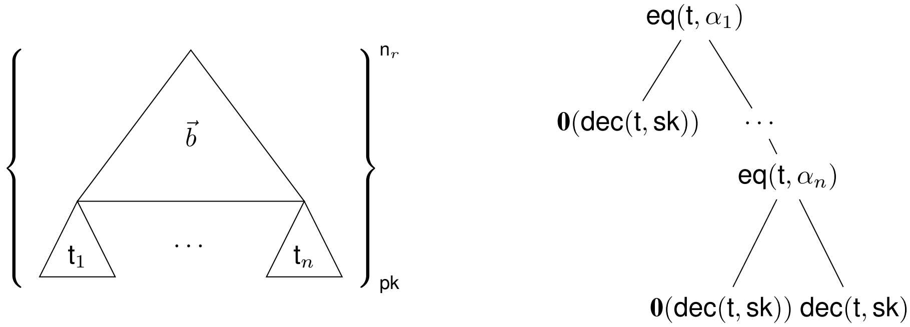
Let $\mathcal{E} = \mathcal{X}_{\mathsf{enc}}\theta_{\mathsf{enc}}$ be the set of encryptions, $\mathcal{D} = \mathcal{X}_{\mathsf{dec}}\lambda_{\mathsf{dec}}$ be set of decryptions and $\mathcal{R} = \mathcal{X}_{\mathsf{enc}}\sigma_{\mathsf{rand}}$ the set of encryption randomness used. We also let $\mathcal{S} = (\mathcal{K}, \mathcal{R}, \mathcal{E}, \mathcal{D})$.

Definition 23. *A $\mathcal{S}$-*encryption oracle call *is a term t of the form $\{u\}^r_{\mathsf{pk}}$ where:*

- $\{u\}^r_{\mathsf{pk}} \in \mathcal{E}$, $r \in \mathcal{R}$, *$(\mathsf{pk}, \mathsf{sk})$ is a valid public/private key pair and with $\mathsf{sk} \in \mathcal{K}$.*
- *u is a $\mathcal{S}$-*normalized simple terms.

Similarly, a *$\mathcal{S}$-decryption oracle calls* t is valid decryption in $\mathcal{D}$ under secret key $\mathsf{sk} \in \mathcal{K}$ such that all other encryptions and decryptions appearing directly in t, either in guards or in the decrypted term, are themselves *$\mathcal{S}$-encryption oracle calls* and *$\mathcal{S}$-decryption oracle calls*.

Definition 24. *A $\mathcal{S}$-*decryption oracle call *is a term of the form $C\left[\vec{g} \diamond (s_i)_{i\le p}\right]$ in $\mathcal{D}$ where:*



Encryption Oracle Call **Decryption Oracle Call**

Convention: $\alpha_1, \dots, \alpha_n$ are the encryptions of $\mathcal{E}$ under pk appearing in t.

Fig. 11. Shapes of Encryption and Decryption Oracle Calls

- *$(\mathsf{pk}, \mathsf{sk})$ is valid public/private key pair and $\mathsf{sk} \in \mathcal{K}$.*
- *There exists a context u if-free and in R-normal form, and a term t such that:*

$$t \equiv u[(\alpha_j)_j, (\mathsf{dec}_k)_k] \qquad \forall i < p,\ s_i \equiv \mathbf{0}(\mathsf{dec}(t, \mathsf{sk})) \qquad s_p \equiv \mathsf{dec}(t, \mathsf{sk}) \qquad \forall g \in \vec{g},\ g \equiv \mathsf{eq}(t, \alpha_j)$$

- *For all j, α_j is a $\mathcal{S}$-*encryption oracle call*.*
- *For all k, dec_k is a $\mathcal{S}$-*decryption oracle call*.*

$(\alpha_j)_j$ are called u's encryptions. We often write $(\mathsf{dec}_k)_k$ to denote a vector of decryption oracle calls.

Figure 11 gives a visual representation of the shapes of encryption and decryption oracle calls.

A *$\mathcal{S}$-basic term* is a term build using *$\mathcal{S}$-encryption oracle calls*, *$\mathcal{S}$-decryption oracle calls*, function symbols in $\mathcal{F} \setminus \{\mathsf{if\_then\_else\_}, \mathbf{0}\}$ and names in $\mathcal{N}$, with some restrictions. More precisely, we require that:

- We do not use names in $\mathcal{R}$, as this would contradict CCA2 randomness side-conditions.
- We do not decrypt terms using secret keys in $\mathcal{K}$.

Definition 25. *A $\mathcal{S}$-*basic term *is a term of the form $U[\vec{w}, (\alpha_j)_j, (\mathsf{dec}_k)_k]$ where:*

- *U and $\vec{w}$ are if-free, U does not contain $\mathbf{0}(\_)$, $\mathsf{fresh}(\mathcal{R}; \vec{w})$ and $\mathsf{nodec}(\mathcal{K}, \vec{w})$.*
- *$(\alpha_j)_j$ are $\mathcal{S}$-*encryption oracle calls*.*
- *$(\mathsf{dec}_k)_k$ are $\mathcal{S}$-*decryption oracle calls*.*

*A $\mathcal{S}$-*basic conditional *is a $\mathcal{S}$-*basic term *of sort $\mathcal{S}_{\mathsf{b}}$.*

A *$\mathcal{S}$-normalized basic term* is a a *$\mathcal{S}$-basic term* that has been built without introducing any R-redex.

Definition 26. *A $\mathcal{S}$-*normalized basic term *is a $\mathcal{S}$-*basic term *of the form $U[\vec{w}, (\alpha_j)_j, (\mathsf{dec}_k)_k]$ where:*

- *$(\alpha_j)_j$ are encryptions under $(\mathsf{pk}_j, \mathsf{sk}_j)_j$, and $(\mathsf{dec}_k)_k$ are decryptions under $(\mathsf{pk}_k, \mathsf{sk}_k)_k$.*
- *$U[\vec{w}, (\{[]_j\}^0_{\mathsf{pk}_j})_j, (\mathsf{dec}([]_k, \mathsf{sk}_k))_k]$ is in R-normal form.*

*A $\mathcal{S}$-*normalized basic conditional *is a $\mathcal{S}$-*normalized basic term *of sort $\mathcal{S}_{\mathsf{b}}$.*

Finally, a *$\mathcal{S}$-simple term* is a term build using only *$\mathcal{S}$-basic term* and the $\mathsf{if\_then\_else\_}$ function symbols. Moreover, if we use only *$\mathcal{S}$-normalized basic term*, then we get an a *$\mathcal{S}$-normalized simple term*.

Definition 27. *A $\mathcal{S}$-*simple term *(resp. $\mathcal{S}$-*normalized simple term*) is a term of the form $C[\vec{b} \diamond \vec{u}]$ where:*

- *C is an if-context.*
- *$\vec{b}$ are $\mathcal{S}$-*basic conditionals *(resp. $\mathcal{S}$-*normalized basic conditionals*).*
- *$\vec{u}$ are $\mathcal{S}$-*basic terms *(resp. $\mathcal{S}$-*normalized basic terms*).*

Remark 8. For all term u, the guards of a $\mathcal{S}_l$-decryption oracle calls are $\mathcal{S}_l$-normalized basic terms. But the *leaves* of $\mathcal{S}$-decryption oracle calls *are not* $\mathcal{S}$-normalized basic terms, because they do not satisfy the condition $\mathsf{nodec}(\mathcal{K}, \cdot)$.

The inductive definition of $\mathcal{S}$-normalized basic terms naturally gives us a relation $<^{\mathcal{S}}_{\mathsf{ind}}$ between $\mathcal{S}$-normalized basic terms, $\mathcal{S}$-normalized simple terms, $\mathcal{S}$-decryption oracle calls and $\mathcal{S}$-encryption oracle calls.

Definition 28. *$<^{\mathcal{S}}_{\mathsf{ind}}$ is the reflexive and transitive closure of the relation $<^{\mathcal{S}}$ defined as:*

- *For all $\mathcal{S}$-encryption oracle call $t \equiv \{u\}_{pk}^r$, $u <^{\mathcal{S}} t$.*
- *For all $\mathcal{S}$-decryption oracle call:*

$$t \equiv C\left[\vec{g}[(\alpha_j)_j, (\mathsf{dec}_k)_k] \diamond (s_i[(\alpha_j)_j, (\mathsf{dec}_k)_k])_{i \le p}\right]$$

  for all j, $\alpha_j <^{\mathcal{S}} t$ and for all k, $\mathsf{dec}_k <^{\mathcal{S}} t$.
- *For all $\mathcal{S}$-normalized basic term $t \equiv U[\vec{w}, (\alpha_j)_j, (\mathsf{dec}_k)_k]$ where: for all j, $\alpha_j <^{\mathcal{S}} t$ and for all k, $\mathsf{dec}_k <^{\mathcal{S}} t$.*
- *For all $\mathcal{S}$normalized simple term $t \equiv C[\vec{b} \diamond \vec{u}]$, $\forall b \in \vec{b}, b <^{\mathcal{S}} t$ and $\forall u \in \vec{u}, u <^{\mathcal{S}} t$.*

We let $\le_{\mathsf{bt}}^{\mathcal{S}}$ be union of the restriction of $<_{\mathsf{ind}}^{\mathcal{S}}$ to the instances where the left term is a $\mathcal{S}$-normalized basic term, and the set of guards appearing in the right-term. Formally:

Definition 29. *Let $<_{\mathit{ind}}^{\prime\mathcal{S}}$ be the reflexive and transitive closure of the order $<^{\prime\mathcal{S}}$, which has the same definition than $<^{\mathcal{S}}$, apart for the $\mathcal{S}$-decryption oracle call:*

- *For all $\mathcal{S}$-decryption oracle call:*

$$t \equiv C\left[\vec{g} \diamond (s_i[(\alpha_j)_j, (\mathsf{dec}_k)_k])_{i \le p}\right]$$

  for all j, $\alpha_j <^{\prime\mathcal{S}} t$; for all k, $\mathsf{dec}_k <^{\prime\mathcal{S}} t$; and for all $b \in \vec{g}$, $b <^{\prime\mathcal{S}} t$.

We finally define $\le_{\mathit{bt}}^{\mathcal{S}}$: for every terms u, v:

$$u \le_{\mathit{bt}}^{\mathcal{S}} v \quad \text{iff} \quad u <_{\mathit{ind}}^{\prime\mathcal{S}} v \ \wedge\ u \text{ is a } \mathcal{S}\text{-normalized basic term}$$

D. Proof Form and Normalized Proof Form

Definition 30. *Let $P \vdash_{\mathcal{A}_{\mathsf{CS}_\square}}^{b} t \sim t'$ in early proof form. We say that this proof is in* proof form *(resp.* normalized proof form*) if:*

$$t \equiv C\left[\left(\boxed{b^{h_l}}\boxed{b^{h_r}}_{b^h}\right)_{h \in H} \diamond \left(D_l\left[(\beta)_{\beta \le_c^{\epsilon,l}(t,P)} \diamond (\gamma)_{\gamma \le_l^{\epsilon,l}(t,P)}\right]\right)_{l \in L}\right]$$

$$t' \equiv C\left[\left(\boxed{b'^{h_l}}\boxed{b'^{h_r}}_{b'^h}\right)_{h \in H} \diamond \left(D_l\left[(\beta')_{\beta' \le_c^{\epsilon,l}(t',P)} \diamond (\gamma')_{\gamma' \le_l^{\epsilon,l}(t',P)}\right]\right)_{l \in L}\right]$$

and it satisfies the following properties:

- *$(b^{h_l})_{h \in H}, (b^{h_r})_{h \in H}$ are terms in* proof forms *(resp.* normalized proof forms*).*
- *For all l, $D_l\left[(\beta)_{\beta \le_c^{\epsilon,l}(t,P)} \diamond (\gamma)_{\gamma \le_l^{\epsilon,l}(t,P)}\right]$ is a $(\mathcal{K}_l^P, \mathcal{E}_l^P)$*-simple term *(resp. $(\mathcal{K}_l^P, \mathcal{E}_l^P)$*-normalized simple term*).*
- *For all l, $D_l\left[(\beta')_{\beta' \le_c^{\epsilon,l}(t',P)} \diamond (\gamma')_{\gamma' \le_l^{\epsilon,l}(t',P)}\right]$ is a $(\mathcal{K}_l'^P, \mathcal{E}_l'^P)$*-simple term *(resp. $(\mathcal{K}_l'^P, \mathcal{E}_l'^P)$*-normalized simple term*).*

We let $P \vdash^{\mathsf{npf}} t \sim t'$ if and only if $P \vdash_{\mathcal{A}_{\mathsf{CS}_\square}} t \sim t'$ and the proof is in normalized proof form*.*

Let $P \vdash^{\mathsf{npf}} t \sim t'$, we already defined the set of conditionals $\le_{\mathsf{c}}^{\mathsf{h},l}(t, P)$ used in the $\overline{\mathsf{BFA}}$ rules in the sub-proof P of at index h and branch l. In the case of proof in normalized proof form, these conditionals are normalized basic conditional. Similarly the set of leave terms $\le_{\mathsf{l}}^{\mathsf{h},l}(t, P)$ in the sub-proof of P of at index h and branch l is a set of normalized basic terms. Recall that a basic term may contain other basic terms in its subterm. Hence we can define the set of all normalized basic terms appearing in the subterms of $\le_{\mathsf{c}}^{\mathsf{h},l}(t, P) \cup \le_{\mathsf{l}}^{\mathsf{h},l}(t, P)$.

Definition 31. *For all $P \vdash^{\mathit{npf}} t \sim t'$, we define $\le_{\mathit{bt}}^{\mathit{h},l}(t, P)$ as follows: for all term s, $s \le_{\mathit{bt}}^{\mathit{h},l}(t, P)$ if and only if there exists $u(\le_{\mathit{c}}^{\mathit{h},l} \cup \le_{\mathit{l}}^{\mathit{h},l})(t, P)$ such that $s \le_{\mathit{bt}}^{\mathcal{S}_l} u$.*

E. Eager Reduction for $\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2}$

Before proving that we can restrict ourselves to term in proof forms we need several auxiliary results about the $\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2}$ fragment, which we state and prove here.

Proposition 12. *For all $b, b' \in \mathcal{T}(\mathcal{F}, \mathcal{N})$, if $b \sim b'$ is derivable in $\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2}$ then $b \equiv C[\vec{w}, (\alpha_i)_i, (\mathsf{dec}_j)_j]$, $b' \equiv C[\vec{w}, (\alpha_i')_i, (\mathsf{dec}_j')_j]$ and the applied $\mathsf{CCA2}$ axiom is:*

$$\vec{w}, (\alpha_i)_i, (\mathsf{dec}_j)_j \sim \vec{w}, (\alpha_i')_i, (\mathsf{dec}_j')_j$$

Proof. This is easy to show by induction on the proof derivation. ∎

We now give the proof of Lemma 2, which we recall below:

Lemma (2)**.** *For all b, b', b'', if $b, b \sim b', b''$ is in the fragment $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2})$ then $b' \equiv b''$.*

Proof. From Proposition 12 we have:

$$b \equiv C^l[\vec{w}^l, (\alpha_i^l)_{i\in I^l}, (\mathsf{dec}_j^l)_{j\in J^l}] \qquad\qquad b' \equiv C^l[\vec{w}^l, (\alpha_i'^l)_{i\in I^l}, (\mathsf{dec}_j'^l)_{j\in J^l}]$$
$$b \equiv C^r[\vec{w}^r, (\alpha_i^r)_{i\in I^r}, (\mathsf{dec}_j^r)_{j\in J^r}] \qquad\qquad b'' \equiv C^r[\vec{w}^r, (\alpha_i'^r)_{i\in I^r}, (\mathsf{dec}_j'^r)_{j\in J^r}]$$

Assume that $C^l \not\equiv C^r$. Let p be the position of a hole of C^l such that p is a valid position but not a hole position in C^r (if this is not the case, invert b' and b''). Then we have three cases:

- If the hole at $b_{|p}$ is mapped to a term $u \in \vec{w}^l$, then we can rewrite the proof such that p is an hole position in both terms.
- If the hole at $b_{|p}$ is mapped to an encryption oracle call $\{m\}^r_{\mathsf{pk(n)}}$ in b and $\{m'\}^r_{\mathsf{pk(n)}}$ in b'. Since $\{m\}^r_{\mathsf{pk(n)}}$ is an encryption in the $\mathsf{CCA2}$ application we know from the freshness side-condition that r does not appear in $\vec{w}^r$.
  Then there exists a context A such that A is not a hole, $m \equiv A[\vec{w}^r, (\alpha_i^r)_{i\in I^r}, (\mathsf{dec}_j^r)_{j\in J^r}]$ and $C^r_{|p} \equiv A$. By consequence we know that $r \in \vec{w}^r$. Absurd.
- If the hole at $b_{|p}$ is mapped to a decryption oracle call $\mathsf{dec}^l_{i_0}$ in b. We let $\mathsf{dec}(m, \mathsf{sk(n)})$ be such that $\mathsf{dec}(m, \mathsf{sk(n)})$ is well-guarded in $\mathsf{dec}^l_{i_0}$. Since $\mathsf{dec}^l_{i_0}$ is a decryption in the $\mathsf{CCA2}$ application we know from the key-usability side-condition that $\mathsf{sk(n)}$ appears only in decryption position in $\vec{w}^r$. Then there exists a context A such that A is not a hole, $b'_{|p} \equiv A[\vec{w}^r, (\alpha_i^r)_{i\in I^r}, (\mathsf{dec}_j^r)_{j\in J^r}]$ and A is if-free. By consequence we know that $\mathsf{FA_{dec}}$ is applied on the right-side, which implies that either $\mathsf{n} \in \vec{w}^r$ or $\mathsf{sk(n)} \in \vec{w}^r$. Absurd. ∎

a) Eager Reduction: We state here a key result about the $\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2}$ fragment, which deals with the following problem: when trying to prove that $u \sim u'$ holds, one may rewrite u and u' into $\langle u\,,\,v\rangle$ and $\langle u'\,,\,v'\rangle$ using R. The problem here is that v and v' are arbitrary large terms, which makes the proof space unbounded. E.g. this is the case in the following proof:

$$\dfrac{\dfrac{\dfrac{\vdots\;(P)}{u, v \sim u', v'}}{\pi_1(\langle u\,,\,v\rangle) \sim \pi_1(\langle u'\,,\,v'\rangle)}\;\mathsf{FA}_{\langle\,,\,\rangle}}{u \sim u'}\;R$$

Of course there is a shortcut here: since (P) is a proof of $u, v \sim u', v'$ using the Restr rule we have a proof of $u \sim u'$. Moreover the Restr elimination Lemma 12 allows us to get rid of v and v', and to get a (no larger) proof P_{cut} of $u \sim u'$.

One may wish to generalize this, and to prove that we can restrict ourselves to proofs where all intermediate terms are in R-normal form. As we saw this is not possible (terms in proof form are not necessarily in R-normal form). Therefore we prove a slightly different result. For all basic terms $C[\vec{w}, (\alpha_i)_{i\in I}, (\mathsf{dec}_j)_{j\in J}]$ and $C'[\vec{w}, (\alpha'_i)_{i\in I}, (\mathsf{dec}'_j)_{j\in J}]$, for all proof:

$$\dfrac{\dfrac{}{\vec{w}, (\alpha_i)_{i\in I}, (\mathsf{dec}_j)_{j\in J} \sim \vec{w}, (\alpha'_i)_{i\in I}, (\mathsf{dec}'_j)_{j\in J}}\;\mathsf{CCA2}}{\vdots \atop C[\vec{w}, (\alpha_i)_{i\in I}, (\mathsf{dec}_j)_{j\in J}] \sim C[\vec{w}, (\alpha'_i)_{i\in I}, (\mathsf{dec}'_j)_{j\in J}]}\;\mathsf{FA_s}^* \cdot \mathsf{Dup}^*$$

we are going to prove that we can assume that there are no redexes in C. This shows that we can assume the basic terms $C[\vec{w}, (\alpha_i)_{i\in I}, (\mathsf{dec}_j)_{j\in J}]$ and $C'[\vec{w}, (\alpha'_i)_{i\in I}, (\mathsf{dec}'_j)_{j\in J}]$ to be *normalized* basic terms.

b) Formal Statement: We are going to prove that we can guarantee that C does not contain any redexes and that some further technical properties holds. These properties (that we discuss below) are used to deal with the fact that $\sim$ is not a congruence: they allow to compose applications of the cut-elimination lemma. We start by discussing the properties needed to compose these cut-eliminations, then give the composition proposition and finally we will state the cut-elimination lemma.

Let $\left(C^k[\vec{w}^k, (\alpha_i^k)_{i\in I^k}, (\mathsf{dec}_j^k)_{j\in J^k}]\right)_k$ and $\left(C'^k[\vec{w}^k, (\alpha_i'^k)_{i\in I^k}, (\mathsf{dec}_j'^k)_{j\in J^k}]\right)_k$ be basic terms, and assume that we have the proof:

$$\dfrac{\dfrac{}{\left(\vec{w}^k, (\alpha_i^k)_{i\in I^k}, (\mathsf{dec}_j^k)_{j\in J^k}\right)_k \sim \left(\vec{w}^k, (\alpha_i'^k)_{i\in I^k}, (\mathsf{dec}_j'^k)_{j\in J^k}\right)_k}\;\mathsf{CCA2}}{\vdots \atop \left(C^k[\vec{w}^k, (\alpha_i^k)_{i\in I^k}, (\mathsf{dec}_j^k)_{j\in J^k}]\right)_k \sim \left(C'^k[\vec{w}^k, (\alpha_i'^k)_{i\in I^k}, (\mathsf{dec}_j'^k)_{j\in J^k}]\right)_k}\;\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \tag{5}$$

Moreover assume that, for all k, there exists basic terms $\tilde{C}^k$, $\tilde{\vec{w}}^k$ and $\tilde{I}^k$, $\tilde{J}^k$ such that we can rewrite the sub-proof of:

$$\tilde{C}^k[\vec{w}^k, (\alpha_i^k)_{i\in I^k}, (\mathsf{dec}_j^k)_{j\in J^k}] \sim \tilde{C}'^k[\vec{w}^k, (\alpha_i'^k)_{i\in I^k}, (\mathsf{dec}_j'^k)_{j\in J^k}]$$

into the following proof:

$$\dfrac{\dfrac{\dfrac{}{\tilde{\vec{w}}^k, (\alpha_i^k)_{i\in\tilde{I}^k}, (\mathsf{dec}_j^k)_{j\in\tilde{J}^k} \sim \tilde{\vec{w}}^k, (\alpha_i'^k)_{i\in\tilde{I}^k}, (\mathsf{dec}_j'^k)_{j\in\tilde{J}^k}}\ \mathsf{CCA2}}{\vdots}}{\dfrac{\tilde{C}^k[\tilde{\vec{w}}^k, (\alpha_i^k)_{i\in\tilde{I}^k}, (\mathsf{dec}_j^k)_{j\in\tilde{J}^k}] \sim \tilde{C}'^k[\tilde{\vec{w}}^k, (\alpha_i'^k)_{i\in\tilde{I}^k}, (\mathsf{dec}_j'^k)_{j\in\tilde{J}^k}]}{C^k[\vec{w}^k, (\alpha_i^k)_{i\in I^k}, (\mathsf{dec}_j^k)_{j\in J^k}] \sim C'^k[\vec{w}^k, (\alpha_i'^k)_{i\in I^k}, (\mathsf{dec}_j'^k)_{j\in J^k}]}\ R}\ \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \tag{6}$$

Then we can recombine the instances of the $\mathsf{CCA2}$ axiom into one instance, as long as we did not introduce new encryptions and new decryptions (i.e. $\tilde{I}^k \subseteq I^k$ and $\tilde{J}^k \subseteq J^k$), and as long as $\tilde{\vec{w}}^k$ does not contain new encryptions randomness or secret keys etc ... A sufficient condition to that ensure the latter property holds is to require that $\tilde{\vec{w}}^k \subseteq \mathsf{st}(\vec{w}^k)$. Putting everything together one get the following proposition:

Proposition 13. *For all basic terms* $\left(C^k[\vec{w}^k, (\alpha_i^k)_{i\in I^k}, (\mathsf{dec}_j^k)_{j\in J^k}]\right)_k$ *and* $\left(C'^k[\vec{w}^k, (\alpha_i'^k)_{i\in I^k}, (\mathsf{dec}_j'^k)_{j\in J^k}]\right)_k$ *such that the proof displayed in Equation* (5) *is valid, if for all* k*, there exists basic terms* $\tilde{C}^k$*,* $\tilde{\vec{w}}^k$ *and* $\tilde{I}^k$*,* $\tilde{J}^k$ *such that:*

- $\mathsf{st}(\tilde{\vec{w}}^k) \subseteq \mathsf{st}(\vec{w}^k)$.
- $\tilde{I}^k \subseteq I^k$ *and* $\tilde{J}^k \subseteq J^k$
- *The derivation in* (6) *is valid.*

Then we have:

$$\dfrac{\dfrac{\dfrac{}{\left(\tilde{\vec{w}}^k, (\alpha_i^k)_{i\in\tilde{I}^k}, (\mathsf{dec}_j^k)_{j\in\tilde{J}^k}\right)_k \sim \left(\tilde{\vec{w}}^k, (\alpha_i'^k)_{i\in\tilde{I}^k}, (\mathsf{dec}_j'^k)_{j\in\tilde{J}^k}\right)_k}\ \mathsf{CCA2}}{\vdots}}{\dfrac{\left(\tilde{C}^k[\tilde{\vec{w}}^k, (\alpha_i^k)_{i\in\tilde{I}^k}, (\mathsf{dec}_j^k)_{j\in\tilde{J}^k}]\right)_k \sim \left(\tilde{C}'^k[\tilde{\vec{w}}^k, (\alpha_i'^k)_{i\in\tilde{I}^k}, (\mathsf{dec}_j'^k)_{j\in\tilde{J}^k}]\right)_k}{\left(C^k[\vec{w}^k, (\alpha_i^k)_{i\in I^k}, (\mathsf{dec}_j^k)_{j\in J^k}]\right)_k \sim \left(C'^k[\vec{w}^k, (\alpha_i'^k)_{i\in I^k}, (\mathsf{dec}_j'^k)_{j\in J^k}]\right)_k}\ R}\ \mathsf{FA_s}^* \cdot \mathsf{Dup}^*$$

Proof. Axioms $\mathsf{FA_s}$ and Dup verify a kind of frame property. If we have the derivation:

$$\frac{\vec{u}' \sim \vec{v}'}{\vec{u} \sim \vec{v}}\ Ax$$

then for all $\vec{w}, \vec{w}'$ of same length, the following derivation is valid:

$$\frac{\vec{w}, \vec{u}' \sim \vec{w}', \vec{v}'}{\vec{w}, \vec{u} \sim \vec{w}', \vec{v}}\ Ax$$

The only problem comes from the $\mathsf{CCA2}$ axiom, which does not verify the frame property. But thanks to the hypothesis $\mathsf{st}(\tilde{\vec{w}}^k) \subseteq \mathsf{st}(\vec{w}^k)$ and $\tilde{I}^k \subseteq I^k$, $\tilde{J}^k \subseteq J^k$, we know that the $\mathsf{CCA2}$ application:

$$\frac{}{\tilde{\vec{w}}^k, (\alpha_i^k)_{i\in\tilde{I}^k}, (\mathsf{dec}_j^k)_{j\in\tilde{J}^k} \sim \tilde{\vec{w}}^k, (\alpha_i'^k)_{i\in\tilde{I}^k}, (\mathsf{dec}_j'^k)_{j\in\tilde{J}^k}}\ \mathsf{CCA2}$$

is "included" in the application:

$$\frac{}{\left(\vec{w}^k, (\alpha_i^k)_{i\in I^k}, (\mathsf{dec}_j^k)_{j\in J^k}\right)_k \sim \left(\vec{w}^k, (\alpha_i'^k)_{i\in I^k}, (\mathsf{dec}_j'^k)_{j\in J^k}\right)_k}\ \mathsf{CCA2}$$

Therefore we can combine all proofs, using Dup to remove duplicates, to get the wanted proof. ■

Lemma 13. *For all basic terms* $C[\vec{w}, (\alpha_i)_{i\in I}, (\mathsf{dec}_j)_{j\in J}]$ *and* $C'[\vec{w}, (\alpha_i')_{i\in I}, (\mathsf{dec}_j')_{j\in J}]$*, if we have a derivation:*

$$\dfrac{\dfrac{\dfrac{}{\vec{w}, (\alpha_i)_{i\in I}, (\mathsf{dec}_j)_{j\in J} \sim \vec{w}, (\alpha_i')_{i\in I}, (\mathsf{dec}_j')_{j\in J}}\ \mathsf{CCA2}}{\vdots}}{C[\vec{w}, (\alpha_i)_{i\in I}, (\mathsf{dec}_j)_{j\in J}] \sim C[\vec{w}, (\alpha_i')_{i\in I}, (\mathsf{dec}_j')_{j\in J}]}\ \mathsf{FA_s}^* \cdot \mathsf{Dup}^*$$

then there exists $\tilde{C}$*,* $\tilde{\vec{w}}$ *and* $\tilde{I}$*,* $\tilde{J}$ *such that:*

- $\mathsf{st}(\tilde{\vec{w}}) \subseteq \mathsf{st}(\vec{w})$.
- $\tilde{I} \subseteq I$*,* $\tilde{J} \subseteq J$
- $\tilde{C}[\tilde{\vec{w}}, (\alpha_i)_{i\in\tilde{I}}, (\mathsf{dec}_j)_{j\in\tilde{J}}]$ *and* $\tilde{C}[\tilde{\vec{w}}, (\alpha_i')_{i\in\tilde{I}}, (\mathsf{dec}_j')_{j\in\tilde{J}}]$ *are normalized basic terms.*

- *We have the following derivation:*

$$\dfrac{\dfrac{\dfrac{}{\tilde{\vec{w}}, (\alpha_i)_{i\in\tilde{I}}, (\textit{dec}_j)_{j\in\tilde{J}} \sim \tilde{\vec{w}}, (\alpha'_i)_{i\in\tilde{I}}, (\textit{dec}'_j)_{j\in\tilde{J}}}\ \textsf{CCA2}}{\vdots}}{\dfrac{\tilde{C}[\tilde{\vec{w}}, (\alpha_i)_{i\in\tilde{I}}, (\textit{dec}_j)_{j\in\tilde{J}}] \sim \tilde{C}[\tilde{\vec{w}}, (\alpha'_i)_{i\in\tilde{I}}, (\textit{dec}'_j)_{j\in\tilde{J}}]}{C[\vec{w}, (\alpha_i)_{i\in I}, (\textit{dec}_j)_{j\in J}] \sim C[\vec{w}, (\alpha'_i)_{i\in I}, (\textit{dec}'_j)_{j\in J}]}\ R}\ \textsf{FA}_\textsf{s}{}^* \cdot \textsf{Dup}^*$$

Proof. We start by observing that if we have a derivation:

$$\dfrac{\dfrac{\dfrac{}{(\vec{w}^k, (\alpha_i^k)_{i\in I}, (\textsf{dec}_j^k)_{j\in J})_k \sim (\vec{w}^k, (\alpha_i'^k)_{i\in I}, (\textsf{dec}_j'^k)_{j\in J})_k}\ \textsf{CCA2}}{\vdots}}{(C[\vec{w}^k, (\alpha_i^k)_{i\in I}, (\textsf{dec}_j^k)_{j\in J}])_k \sim (C[\vec{w}^k, (\alpha_i'^k)_{i\in I}, (\textsf{dec}_j'^k)_{j\in J}])_k}\ \textsf{FA}_\textsf{s}{}^* \cdot \textsf{Dup}^*$$

Then we can apply the lemma for each k and recombine the proofs together using Proposition 13.

We prove the lemma by induction on the context C. Each time we say we have a shortcut we use Lemma 12 to get ride of the $\textsf{Restr}$ application introduced by the shortcut.

- Both left and right side can be reduced by $\pi_i(\langle x_1\ ,\ x_2\rangle) \to x_i$. W.l.o.g we assume $i = 1$, therefore we have:

$$\dfrac{\begin{array}{c}\left\langle C^1[\vec{w}^1, (\alpha_i^1)_{i\in I}, (\textsf{dec}_j^1)_{j\in J}]\ ,\ C^2[\vec{w}^2, (\alpha_i^2)_{i\in I}, (\textsf{dec}_j^2)_{j\in J}]\right\rangle \\ \sim\quad \left\langle C^1[\vec{w}^1, (\alpha_i'^1)_{i\in I}, (\textsf{dec}_j'^1)_{j\in J}]'\ ,\ C^2[\vec{w}^2, (\alpha_i'^2)_{i\in I}, (\textsf{dec}_j'^2)_{j\in J}]\right\rangle\end{array}}{\begin{array}{c}\pi_1\left(\left\langle C^1[\vec{w}^1, (\alpha_i^1)_{i\in I}, (\textsf{dec}_j^1)_{j\in J}]\ ,\ C^2[\vec{w}^2, (\alpha_i^2)_{i\in I}, (\textsf{dec}_j^2)_{j\in J}]\right\rangle\right) \\ \sim\quad \pi_1\left(\left\langle C^1[\vec{w}^1, (\alpha_i'^1)_{i\in I}, (\textsf{dec}_j'^1)_{j\in J}]'\ ,\ C^2[\vec{w}^2, (\alpha_i'^2)_{i\in I}, (\textsf{dec}_j'^2)_{j\in J}]\right\rangle\right)\end{array}}\ \textsf{FA}_{\pi_1}$$

  By induction hypothesis we have a derivation of the premise in which terms are normalized basic terms. Observe that this implies that the normalized basic terms start with a pair symbol, therefore we have:

$$\dfrac{\dfrac{\begin{array}{c}\left\langle \tilde{C}^1[\tilde{\vec{w}}^1, (\tilde{\alpha}_i^1)_{i\in I}, (\tilde{\textsf{dec}}_j^1)_{j\in J}]\ ,\ \tilde{C}^2[\tilde{\vec{w}}^2, (\tilde{\alpha}_i^2)_{i\in I}, (\tilde{\textsf{dec}}_j^2)_{j\in J}]\right\rangle \\ \sim\quad \left\langle \tilde{C}^1[\tilde{\vec{w}}^1, (\tilde{\alpha}_i'^1)_{i\in I}, (\tilde{\textsf{dec}}_j'^1)_{j\in J}]'\ ,\ \tilde{C}^2[\tilde{\vec{w}}^2, (\tilde{\alpha}_i'^2)_{i\in I}, (\tilde{\textsf{dec}}_j'^2)_{j\in J}]\right\rangle\end{array}}{\begin{array}{c}\pi_1\left(\left\langle \tilde{C}^1[\tilde{\vec{w}}^1, (\tilde{\alpha}_i^1)_{i\in I}, (\tilde{\textsf{dec}}_j^1)_{j\in J}]\ ,\ \tilde{C}^2[\tilde{\vec{w}}^2, (\tilde{\alpha}_i^2)_{i\in I}, (\tilde{\textsf{dec}}_j^2)_{j\in J}]\right\rangle\right) \\ \sim\quad \pi_1\left(\left\langle \tilde{C}^1[\tilde{\vec{w}}^1, (\tilde{\alpha}_i'^1)_{i\in I}, (\tilde{\textsf{dec}}_j'^1)_{j\in J}]'\ ,\ \tilde{C}^2[\tilde{\vec{w}}^2, (\tilde{\alpha}_i'^2)_{i\in I}, (\tilde{\textsf{dec}}_j'^2)_{j\in J}]\right\rangle\right)\end{array}}\ \textsf{FA}_{\pi_1}}{\begin{array}{c}\pi_1\left(\left\langle C^1[\vec{w}^1, (\alpha_i^1)_{i\in I}, (\textsf{dec}_j^1)_{j\in J}]\ ,\ C^2[\vec{w}^2, (\alpha_i^2)_{i\in I}, (\textsf{dec}_j^2)_{j\in J}]\right\rangle\right) \\ \sim\quad \pi_1\left(\left\langle C^1[\vec{w}^1, (\alpha_i'^1)_{i\in I}, (\textsf{dec}_j'^1)_{j\in J}]'\ ,\ C^2[\vec{w}^2, (\alpha_i'^2)_{i\in I}, (\textsf{dec}_j'^2)_{j\in J}]\right\rangle\right)\end{array}}\ R$$

  We look at the next rule:

  - Either it is an is a unitary axioms and both terms are the same, in which case we can construct directly a derivation (by induction over P) of:

$$\dfrac{\dfrac{\tilde{C}^1[\tilde{\vec{w}}^1, (\tilde{\alpha}_i^1)_{i\in I}, (\tilde{\textsf{dec}}_j^1)_{j\in J}] \sim \tilde{C}^1[\tilde{\vec{w}}^1, (\tilde{\alpha}_i'^1)_{i\in I}, (\tilde{\textsf{dec}}_j'^1)_{j\in J}]'}{\tilde{C}^1[\tilde{\vec{w}}^1, (\tilde{\alpha}_i^1)_{i\in I}, (\tilde{\textsf{dec}}_j^1)_{j\in J}] \sim \tilde{C}^1[\tilde{\vec{w}}^1, (\tilde{\alpha}_i'^1)_{i\in I}, (\tilde{\textsf{dec}}_j'^1)_{j\in J}]'}\ \textsf{FA}_{\pi_1}}{\begin{array}{c}\pi_1\left(\left\langle C^1[\vec{w}^1, (\alpha_i^1)_{i\in I}, (\textsf{dec}_j^1)_{j\in J}]\ ,\ C^2[\vec{w}^2, (\alpha_i^2)_{i\in I}, (\textsf{dec}_j^2)_{j\in J}]\right\rangle\right) \\ \sim\quad \pi_1\left(\left\langle C^1[\vec{w}^1, (\alpha_i'^1)_{i\in I}, (\textsf{dec}_j'^1)_{j\in J}]'\ ,\ C^2[\vec{w}^2, (\alpha_i'^2)_{i\in I}, (\textsf{dec}_j'^2)_{j\in J}]\right\rangle\right)\end{array}}\ R$$

  - Or it is a function application: it can only be a function application on the pair symbol, hence we have a shortcut.

- Only one side can be reduced by $\pi_i(\langle x_1\ ,\ x_2\rangle) \to x_i$. This is impossible since, at all positions in the proof, corresponding terms start with the same function symbol. Absurd.
- Both sides can be reduced by $\textsf{dec}(\{x\}_{\textsf{pk}(n)}^{\textsf{n}_r}, \textsf{sk}(n)) \to x$:

$$\dfrac{\{u\}_{\textsf{pk}(n)}^r, \textsf{sk}(n) \sim \{u'\}_{\textsf{pk}(n')}^{r'}, \textsf{sk}(n')}{\textsf{dec}(\{u\}_{\textsf{pk}(n)}^r, \textsf{sk}(n)) \sim \textsf{dec}(\{u'\}_{\textsf{pk}(n')}^{r'}, \textsf{sk}(n'))}$$

Using the induction hypothesis we know that we have a derivation of $\{u\}^r_{\mathsf{pk}(n)}, \mathsf{sk}(n) \sim \{u'\}^{r'}_{\mathsf{pk}(n')}, \mathsf{sk}(n')$ where intermediate terms are normalized basic conditionals. We look at the next rule applied on $\{u\}^r_{\mathsf{pk}(n)}, \_ \sim \{u'\}^{r'}_{\mathsf{pk}(n')}, \_$ which is not Dup. If it is a function application then we have a shortcuts, if it is a unitary axioms then we have two cases:

- $\{u\}^r_{\mathsf{pk}(n)}$ is a renaming of $\{u'\}^{r'}_{\mathsf{pk}(n')}$, in which case we can build by induction a proof of $u \sim u'$ whose intermediate terms are normalized basic conditionals.
- $\{u\}^r_{\mathsf{pk}(n)}$ is a not renaming of $\{u'\}^{r'}_{\mathsf{pk}(n')}$, in which case the IND-CCA2 axiom is used. This means that at the root of the proof tree we know that sk appears only in decryption position. By induction we show that this is not the case. Absurd.

- Only one side can be reduced by $\mathsf{dec}(\{x\}^r_{\mathsf{pk}(n)}, \mathsf{sk}(n)) \to x$. Observe that it is necessarily of the form:

$$\frac{\{t\}^r_{\mathsf{pk}(n)}, \mathsf{sk}(n) \sim \{t'\}^{r'}_{p'}, \mathsf{sk}'(n')}{\mathsf{dec}(\{t\}^r_{\mathsf{pk}(n)}, \mathsf{sk}(n)) \sim \mathsf{dec}(\{t'\}^{r'}_{p'}, \mathsf{sk}'(n'))}$$

We look at the next rule applied to $\{t\}^r_{\mathsf{pk}(n)}$ and $\{t'\}^{r'}_{p'}$ it which is not Dup:

- If it is a unitary axiom, then necessarily $p' \equiv \mathsf{pk}(n)$ and $n' \equiv n$. Therefore the right side can be reduced by $\mathsf{dec}(\{x\}^r_{\mathsf{pk}(n)}, \mathsf{sk}(n)) \to x$. Absurd.
- If it is $\mathsf{FA}_{\{\}}$ then there is a proof of $\mathsf{pk}(n), \mathsf{sk}(n) \sim p', \mathsf{sk}(n')$, which implies that $p' \equiv \mathsf{pk}(n)$ and $n' \equiv n$. Therefore the right side can be reduced by $\mathsf{dec}(\{x\}^r_{\mathsf{pk}(n)}, \mathsf{sk}(n)) \to x$. Absurd.

- Both side can be reduced by $\mathsf{eq}(x, x) \to \mathsf{true}$. In this case the cut is trivial.
- Only one side can be reduced by $\mathsf{eq}(x, x) \to \mathsf{true}$. Therefore we have a proof of the form:

$$\frac{t, t \sim t', t''}{\mathsf{eq}(t, t) \sim \mathsf{eq}(t', t'')}\ \mathsf{FA}_{\mathsf{eq}(,)}$$

Using Lemma 2 we know that $t' \equiv t''$, therefore both side can be reduced by $\mathsf{eq}(x, x) \to \mathsf{true}$. Absurd. ■

F. Restriction to Proofs in Normalized Proof Form

Definition 32. *We let $\overline{\mathsf{CCA2}}$ be the restriction of $\mathsf{CCA2}$ to cases $\vec{w}, (\alpha_i)_i, (\mathsf{dec}_j)_j \sim \vec{w}', (\alpha'_i)_i, (\mathsf{dec}'_j)_j$ where:*

- *$(\alpha_j)_j, (\alpha'_j)_j$ are encryption oracle calls.*
- *$(\mathsf{dec}_j)_j, (\mathsf{dec}'_j)_j$ are decryption oracle calls.*

Lemma 14. *The following strategy is complete for $\mathfrak{F}((\mathsf{CS} + \mathsf{FA} + R + \mathsf{Dup} + \mathsf{CCA2})^*)$:*

$$\mathfrak{F}((\mathsf{2Box} + R_\square)^* \cdot \mathsf{CS}^*_\square \cdot \{\overline{\mathsf{BFA}}(b, b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \overline{\mathsf{CCA2}})$$

Proof. We showed in Lemma 7 that the following strategy:

$$\mathfrak{F}((\mathsf{2Box} + R_\square)^* \cdot \mathsf{CS}^*_\square \cdot \{\overline{\mathsf{BFA}}(b, b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2})$$

is complete for $\mathsf{CS} + \mathsf{FA} + R + \mathsf{Dup} + \mathsf{CCA2}$. Let P be a proof of $t \sim t'$ in this fragment. Let $L^P = \mathsf{label}(P)$ the set of indices of the branch of the proof tree. Recall that $\mathcal{E}^P_l, \mathcal{D}^P_l$ and $\mathcal{K}^P_l$ are the sets of, respectively, encryptions, decryptions and keys used in the $\mathsf{CCA2}$ instance of branch l, and that $\mathcal{S}^P_l = (\mathcal{K}^P_l, \mathcal{R}^P_l, \mathcal{E}^P_l, \mathcal{D}^P_l)$. We define the order $<$ as follows: for all $u, u' \in \mathcal{E}^P_l \cup \mathcal{D}^P_l$, we let $u < u'$ hold if u is a strict subterm of u'.

We show that for all proof P of $t \sim t'$ in the above fragment, there is a proof Q of $t \sim t'$ where for all $l \in \mathsf{label}(Q)$, all $u \in \mathcal{E}^Q_l$ are $\mathcal{S}^Q_l$-encryption oracle calls, and all $u \in \mathcal{D}^Q_l$ are $\mathcal{S}^Q_l$-decryption oracle calls (the same holds for $\mathcal{E}'^Q_l$ and $\mathcal{D}'^Q_l$). We prove this by induction on the number of elements of $\bigcup_l \mathcal{E}^P_l \cup \mathcal{D}^P_l$ that are not $\mathcal{S}^P_l$-encryption oracle calls or $\mathcal{S}^P_l$-decryption oracle calls, plus the number of elements of $\bigcup_l \mathcal{E}'^P_l \cup \mathcal{D}'^P_l$ that are not $\mathcal{S}'^P_l$-encryption oracle calls or $\mathcal{S}'^P_l$-decryption oracle calls.

Let P be a proof of $t \sim t'$, $l \in L^P$ and let u maximal for $<$ which is not a $\mathcal{S}^P_l$-encryption oracle call or a $\mathcal{S}^P_l$-decryption oracle call.

- If $u \in \mathcal{E}^P_l$ is an encryption. We know that $u \equiv \{m\}^{\mathsf{n}_r}_{\mathsf{pk}}$ where the corresponding secret key sk is in $\mathcal{K}^P_l$. Let $(\alpha_k)_k$ be the set of elements of $\mathcal{E}^P_l \cap \mathsf{st}(m)$, and let $(\mathsf{dec}_n)_n$ be the set of elements of $\mathcal{D}^P_l \cap \mathsf{st}(m)$. We know that there exists a context C such that:

$$m \equiv C[(\alpha_k)_k, (\mathsf{dec}_n)_n]$$

We let A be an if-context and $(B_i[])_i$, $(U_m[])_m$ be if-free contexts in R-normal form such that $C[] =_R A[(B_i[])_i \diamond (U_m[])_m]$. Let m_0 be the term:

$$m_0 \equiv A[(B_i[(\alpha_k)_k, (\mathsf{dec}_n)_n])_i \diamond (U_m[(\alpha_k)_k, (\mathsf{dec}_n)_n])_m]$$

We know that $m_0 =_R m$. By maximality of u we know that the $(\alpha_k)_k$ are $\mathcal{S}_l^P$-encryption oracle calls, and the $(\mathsf{dec}_n)_n$ are $\mathcal{S}_l^P$-decryption oracle calls. For all k we know that $\alpha_k \equiv \{\_\}_{\mathsf{pk}_k}^{\mathsf{n}_k}$, and for all l let sk_n be the secret key of dec_n. Assume that there is some i such that:

$$\tilde{m} \equiv B_i[(\{[]_k\}_{\mathsf{pk}_k}^{\mathsf{n}_k})_k, (\mathsf{dec}([]_n, \mathsf{sk}_n))_n]$$

is not in R-normal form. Since $B_i[]$ is in R-normal form, this means that there exists some k such that $\mathsf{dec}(\{[]_k\}_{\mathsf{pk}_k}^{\mathsf{n}_k}, \mathsf{sk}_k)$ is a subterm of $\tilde{m}$. This implies that sk_k is a subterm of $B_i[]$. But $\mathsf{sk}_k \in \mathcal{K}_l^P$, and therefore B_i cannot contain sk_k as a subterm. Absurd. The same me reasoning applies to $U_m[(\alpha_k)_k, (\mathsf{dec}_n)_n]$.
Therefore for all k (resp. for all m), $B_i[(\alpha_k)_k, (\mathsf{dec}_n)_n]$ (resp. $U_m[(\alpha_k)_k, (\mathsf{dec}_n)_n]$) is an $\mathcal{S}_l^P$-normalized basic term. Hence m_0 is a $\mathcal{S}_l^P$-normalized simple term. We then rewrite, using R, all occurrences of $\{m\}_{\mathsf{sk}}^{\mathsf{n}_r}$ by $\{m'\}_{\mathsf{sk}}^{\mathsf{n}_r}$ in branch l, i.e in every:

$$D_l^{\mathsf{h}}\left[\left(B_{i,l}^{\mathsf{h}}[\vec{w}_{i,l}^{\mathsf{h}}, (\alpha_{i,l}^{\mathsf{h},j})_j, (\mathsf{dec}_{i,l}^{\mathsf{h},k})_k]\right)_i \diamond \left(U_{m,l}^{\mathsf{h}}[\vec{w}_{m,l}^{\mathsf{h}}, (\alpha_{m,l}^{\mathsf{h},j})_j, (\mathsf{dec}_{m,l}^{\mathsf{h},k})_k]\right)_m\right]$$

with $\mathsf{h} \in \mathsf{h\text{-}branch}(l)$. We can check that this yields a new proof Q of $t \sim t'$ with a smaller number of terms in $\mathcal{E}_l^Q \cup \mathcal{D}_l^Q$ which are not $\mathcal{S}_l^Q$-encryption oracle calls or $\mathcal{S}_l^Q$-decryption oracle calls: the only difficulty lies in making sure that the side-conditions of the decryptions still holds. This is the case, for example look at one of the conditions under which a encryption $\alpha_0 \equiv \{m_0\}_{\mathsf{pk}}^{\mathsf{n}_0}$ must be guarded in $\mathsf{dec}(u_0, \mathsf{sk})$: we require that $\mathsf{n}_0 \in \mathsf{st}(u_0 \downarrow_R)$, which is indeed stable under any R rewriting (hence in particular the rewriting of $\{m\}_{\mathsf{sk}}^{\mathsf{n}_r}$ into $\{m'\}_{\mathsf{sk}}^{\mathsf{n}_r}$).
Since all other branches $l' \in L_P \backslash \{l\}$ are left unchanged, and since the right part of the proof (corresponding to t') is also left unchanged we can conclude using the induction hypothesis.
- One can easily check that the case where $u \equiv C[(g_e)_e \diamond (s_a)_{a \leq p}] \in \mathcal{D}_l^P$ is a decryption cannot happen. ■

We are now ready to give the proof of Lemma 8, which we recall below.

Lemma (8)**.** *The restriction of the fragment $\mathcal{A}_{\succ}$ to formulas provable in $\vdash^{\mathsf{npf}}$ is complete for $\mathfrak{F}((\mathsf{CS} + \mathsf{FA} + R + \mathsf{Dup} + \mathsf{CCA2})^*)$.*

Proof. Using Lemma 14 we know that the strategy:

$$\mathfrak{F}((\mathsf{2Box} + R_\square)^* \cdot \mathsf{CS}_\square^* \cdot \{\overline{\mathsf{BFA}}(b, b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \overline{\mathsf{CCA2}})$$

is complete for $\mathfrak{F}((\mathsf{CS} + \mathsf{FA} + R + \mathsf{Dup} + \overline{\mathsf{CCA2}})^*)$.

First we show that this strategy remains complete even if with restrict it to proofs such that the terms after $(\mathsf{2Box} + R_\square)^*$ are in proof form. Let P be such that $P \vdash_{\mathcal{A}_{\mathsf{CS}_\square}} t \sim t'$, we want to find $t_0 =_R t, t_0' =_R t'$ and P' such that $P' \vdash_{\mathcal{A}_{\mathsf{CS}_\square}}^{\mathsf{b}} t \sim t'$ is in proof form.

Assume that $P_0 \vdash_{\mathcal{A}_{\mathsf{CS}_\square}} t \sim t'$, using Proposition 11 we know that there exists P such that $P \vdash_{\mathcal{A}_{\mathsf{CS}_\square}}^{\mathsf{b}} t \sim t'$. Let $h \in \mathsf{index}(P), \mathsf{x} \in \{\mathsf{l}, \mathsf{r}\}$, $\mathsf{h} = h_{\mathsf{x}}$. We know that there exists $b^{\mathsf{h}}, b'^{\mathsf{h}}$ such that $\mathsf{extract}_{\mathsf{x}}(h, P) \vdash_{\mathcal{A}_{\mathsf{CS}_\square}}^{\mathsf{b}} b^{\mathsf{h}} \sim b'^{\mathsf{h}}$. To get a proof with terms in proof form, we need to show that for all h, l, for all $(\beta, \beta')(\leq_{\mathsf{c}\sim\mathsf{c}}^{\mathsf{h},l} \cup \leq_{\mathsf{|}\sim\mathsf{|}}^{\mathsf{h},l})(t \sim t', P)$, β, β' are of the form:

$$\beta \equiv B[\vec{w}, (\alpha_j)_j, (\mathsf{dec}_k)_k] \quad \wedge \quad \beta' \equiv B[\vec{w}, (\alpha_j')_j, (\mathsf{dec}_k')_k]$$

the contexts B is if-free. Assume that this is not the case. Then there exists contexts B_e, B_c, B_0, B_1 such that:

$$B \equiv B_e[\mathsf{if}\ B_c\ \mathsf{then}\ B_0\ \mathsf{else}\ B_1]] =_R \mathsf{if}\ B_c\ \mathsf{then}\ B_e[B_0]\ \mathsf{else}\ B_e[B_1]$$

Let t_0 be the term obtained from t by replacing this occurrence of β by:

$$\mathsf{if}\ B_c[\vec{w}, (\alpha_j)_j, (\mathsf{dec}_k)_k]\ \mathsf{then}\ (B_e[B_0])[\vec{w}, (\alpha_j)_j, (\mathsf{dec}_k)_k]\ \mathsf{else}\ (B_e[B_1])[\vec{w}, (\alpha_j)_j, (\mathsf{dec}_k)_k]$$

Similarly we define t_0' by replacing β' by the corresponding term. Then $t_0 =_R t$ and $t_0' =_R t'$. Moreover it is easy to check that the formula $t_0 \sim t_0'$ is provable in $\vdash_{\mathcal{A}_{\mathsf{CS}_\square}}^{\mathsf{b}}$, as we replaced one $\overline{\mathsf{BFA}}$ application by three $\overline{\mathsf{BFA}}$ applications (without changing the encryptions, decryptions or branches of the proof etc ...).

Moreover we replaced B by three terms $B_c, B_e[B_0], B_e[B_1]$ containing strictly less $\mathsf{if\ then\ else}$ applications. Therefore we can show by induction that we can ensure that all the contexts B are if-free by repeating the proof rewriting above.

To show that there a proof of $t \sim t'$ such that the terms after $(\mathsf{2Box} + R_\square)^*$ are in *normalized* proof form, we only have to apply the Lemma 13 to all branches l, and to commute the new R rewriting to the bottom of the proof. ■

APPENDIX V
RESTRICTIONS ON THE BASIC CONDITIONALS PART

In this section, we give the proof of Proposition 5.

A. Properties of Normalized Basic Terms

Definition 33. *We call a* conditional context *a context $C[]_{\vec{x}}$ such that all holes appear in the conditional part of an* $\textsf{\textit{if then else}}$. *Formally, for all position p, if $C_{|p}$ is a hole $[]_x$ then $p = p'.0$ and there exists u, v such that:*

$$C_{|p'} \equiv \textsf{\textit{if}}\ []_x\ \textsf{\textit{then}}\ u\ \textsf{\textit{else}}\ v$$

We say that u is an almost conditional context *if u a conditional context or a hole.*

The main goal of this subsection is to show the following lemma.

Lemma 15. *For all $P \vdash^{\textsf{npf}} t \sim t'$, for all $\textsf{h}, l$ and $\beta, \beta' \leq_{\textsf{bt}}^{\textsf{h},l} (t, P)$, there exists an almost conditional context $\tilde{\beta}'[]$ such that:*

$$\beta' \equiv \tilde{\beta}'[\beta] \quad \wedge \quad \textsf{leave-st}(\beta \downarrow_R) \cap \textsf{cond-st}\left(\tilde{\beta}'[] \downarrow_R\right) = \emptyset$$

Before delving in the proof, we would like to remark that the above lemma is not entirely satisfactory. Consider the following example:

$$\beta_0 \equiv \textsf{eq}(\{\textsf{if}\ b\ \textsf{then}\ s\ \textsf{else}\ t\}_{\textsf{pk(n)}}^{\textsf{n}_r}, 0) =_R \textsf{if}\ b\ \textsf{then}\ \underbrace{\textsf{eq}(\{s\}_{\textsf{pk(n)}}^{\textsf{n}_r}, 0)}_{\beta_0^0}\ \textsf{else}\ \underbrace{\textsf{eq}(\{t\}_{\textsf{pk(n)}}^{\textsf{n}_r}, 0)}_{\beta_0^1}$$

$$\beta_1 \equiv \textsf{eq}(\{\textsf{if}\ \beta_0^0\ \textsf{then}\ u\ \textsf{else}\ u\}_{\textsf{pk(n)}}^{\textsf{n}_r'}, 0)$$

where $\beta_0^0, \beta_0^1 \not\in \textsf{cond-st}(u \downarrow_R)$ and $s \neq_R t$. Then $\beta_0^0, \beta_0^1 \not\in \textsf{cond-st}(\beta_1 \downarrow_R)$, because β_0^0 disappear using the rule $\textsf{if}\ x\ \textsf{then}\ y\ \textsf{else}\ y \to y$ in R. Hence, Lemma 15 could choose $\tilde{\beta}_1 \equiv \beta_1$. Of course this situation cannot occur, as we cannot have β_0^0 be a subterm of β_1 (this contradicts the freshness side-condition of encryptions' randomnesses in the $\textsf{CCA2}$ axiom). But we cannot rule this situation out simply by applying the lemma, we have to make a more in-depth analysis. We would like to a stronger version of this lemma that somehow directly "includes" the above observation.

To do this we introduce over-approximations of $\textsf{cond-st}(\cdot \downarrow_R)$ and $\textsf{leave-st}(\cdot \downarrow_R)$, show that Lemma 15 holds for the over-approximations of $\textsf{cond-st}(\cdot \downarrow_R)$ and $\textsf{leave-st}(\cdot \downarrow_R)$.

Definition 34. *We define the function $\overline{\textsf{leave-st}}$ from the set of terms to the set of if-free terms in R-normal form:*

$$\overline{\textsf{leave-st}}(u_0, \dots, u_n) = \cup_{i \leq n} \overline{\textsf{leave-st}}(u_i) \qquad \overline{\textsf{leave-st}}(\textsf{\textit{if}}\ b\ \textsf{\textit{then}}\ u\ \textsf{\textit{else}}\ v) = \overline{\textsf{leave-st}}(u, v)$$

$$\overline{\textsf{leave-st}}(f(u_0, \dots, u_n)) = \left\{ f(v_0, \dots, v_n) \downarrow_R \mid \forall i \leq n, v_i \in \overline{\textsf{leave-st}}(u_i) \right\} \quad (\forall f \in \mathcal{F}_s \cup \mathcal{N})$$

We define the function $\overline{\textsf{cond-st}}$ from the set of terms to the set of if-free conditionals in R-normal form:

$$\overline{\textsf{cond-st}}(u_0, \dots, u_n) = \cup_{i \leq n} \overline{\textsf{cond-st}}(u_i) \qquad \overline{\textsf{cond-st}}(f(\vec{u})) = \overline{\textsf{cond-st}}(\vec{u}) \quad (\forall f \in \mathcal{F}_s \cup \mathcal{N})$$

$$\overline{\textsf{cond-st}}(\textsf{\textit{if}}\ b\ \textsf{\textit{then}}\ u\ \textsf{\textit{else}}\ v) = \overline{\textsf{cond-st}}(b) \cup \overline{\textsf{leave-st}}(b) \cup \overline{\textsf{cond-st}}(u, v)$$

Remark 9. The over-approximation is two-fold: for $\overline{\textsf{leave-st}}()$ there is a first over-approximation, and for $\overline{\textsf{cond-st}}()$ there is an over-approximation, plus the over-approximation of $\overline{\textsf{leave-st}}()$.

Proposition 14. $\overline{\textsf{leave-st}}$ *and* $\overline{\textsf{cond-st}}$ *are sound over-approximations:*

- *For all $u \to_R^* u'$, $\overline{\textsf{leave-st}}(u) \supseteq \overline{\textsf{leave-st}}(u')$. Moreover $\overline{\textsf{leave-st}}(u \downarrow_R) = \textsf{leave-st}(u \downarrow_R)$.*
- *For all $u \to_R^* u'$, $\overline{\textsf{cond-st}}(u) \supseteq \overline{\textsf{cond-st}}(u')$. Moreover $\overline{\textsf{cond-st}}(u \downarrow_R) = \textsf{cond-st}(u \downarrow_R)$.*

Proof. The facts that $\overline{\textsf{leave-st}}(u \downarrow_R) = \textsf{leave-st}(u \downarrow_R)$ and $\overline{\textsf{cond-st}}(u \downarrow_R) = \textsf{cond-st}(u \downarrow_R)$ are straightforward to show. Let us prove by induction on $\to_R^*$ that for all $u \to_R^* u'$, $\overline{\textsf{leave-st}}(u) \supseteq \overline{\textsf{leave-st}}(u')$. If $u \equiv u'$ this is immediate, assume that $u \to_R v \to_R^* u'$. By induction hypothesis we know that $\overline{\textsf{leave-st}}(v) \supseteq \overline{\textsf{leave-st}}(u')$. We then have a case disjunction (we omit the redundant or obvious cases):

- $u \equiv \textsf{if}\ b\ \textsf{then}\ (\textsf{if}\ b\ \textsf{then}\ s\ \textsf{else}\ t)\ \textsf{else}\ w$ and $v \equiv \textsf{if}\ b\ \textsf{then}\ s\ \textsf{else}\ w$ then:

$$\begin{aligned}\overline{\textsf{leave-st}}(u) &= \overline{\textsf{leave-st}}(s) \cup \overline{\textsf{leave-st}}(t) \cup \overline{\textsf{leave-st}}(w) \\ &\supseteq \overline{\textsf{leave-st}}(s) \cup \overline{\textsf{leave-st}}(w) = \overline{\textsf{leave-st}}(v) \\ &\supseteq \overline{\textsf{leave-st}}(u')\end{aligned}$$

- $u \equiv \text{if } b \text{ then } s \text{ else } s$ and $v \equiv s$ then:

$$\overline{\text{leave-st}}(u) = \overline{\text{leave-st}}(s) = \overline{\text{leave-st}}(v)$$

- $u \equiv \text{if } (\text{if } b \text{ then } a \text{ else } c) \text{ then } s \text{ else } t$ and $v \equiv \text{if } b \text{ then } (\text{if } a \text{ then } s \text{ else } t) \text{ else } (\text{if } c \text{ then } s \text{ else } t)$ then:

$$\overline{\text{leave-st}}(u) = \overline{\text{leave-st}}(s) \cup \overline{\text{leave-st}}(t) = \overline{\text{leave-st}}(v)$$

- $u \equiv \text{if } b \text{ then } (\text{if } a \text{ then } s \text{ else } t) \text{ else } w$ and $v \equiv \text{if } a \text{ then } (\text{if } b \text{ then } s \text{ else } w) \text{ else } (\text{if } b \text{ then } t \text{ else } w)$ then:

$$\overline{\text{leave-st}}(u) = \overline{\text{leave-st}}(s) \cup \overline{\text{leave-st}}(t) \cup \overline{\text{leave-st}}(w) = \overline{\text{leave-st}}(v)$$

- $u \equiv f(\vec{w}, \text{if } b \text{ then } \vec{s} \text{ else } \vec{t})$ and $v \equiv \text{if } b \text{ then } f(\vec{w}, \vec{s}) \text{ else } f(\vec{w}, \vec{t})$ then:

$$\begin{aligned}
\overline{\text{leave-st}}(u) \;&=\; \{f(\vec{w}', \vec{w}'') \downarrow_R | \, \forall i, w'_i \in \overline{\text{leave-st}}(w_i) \wedge \forall j, w''_j \in \overline{\text{leave-st}}(s_j) \cup \overline{\text{leave-st}}(t_j)\} \\
&\supseteq\; \{f(\vec{w}', \vec{w}'') \downarrow_R | \, \forall i, w'_i \in \overline{\text{leave-st}}(w_i) \wedge \forall j, w''_j \in \overline{\text{leave-st}}(s_j)\} \\
&\quad \cup \{f(\vec{w}', \vec{w}'') \downarrow_R | \, \forall i, w'_i \in \overline{\text{leave-st}}(w_i) \wedge \forall j, w''_j \in \overline{\text{leave-st}}(t_j)\} \\
&\supseteq\; \overline{\text{leave-st}}(f(\vec{w}, \vec{s})) \cup \overline{\text{leave-st}}(f(\vec{w}, \vec{t})) \\
&\supseteq\; \overline{\text{leave-st}}(v)
\end{aligned}$$

- $(u \equiv \pi_i(\langle s_1 \, , \, s_2 \rangle), \, v \equiv s_i)$ and $(u \equiv \text{dec}(\{m\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}, \mathsf{sk(n)}), \, v \equiv m)$ are trivial.

a) : Similarly, we show by induction on $\to^*_R$ that for all $u \to^*_R u'$, $\overline{\text{cond-st}}(u) \supseteq \overline{\text{cond-st}}(u')$. If $u \equiv u'$ this is immediate, assume that $u \to_R v \to^*_R u'$. By induction hypothesis we know that $\overline{\text{leave-st}}(v) \supseteq \overline{\text{leave-st}}(u')$. We then have a case disjunction (we omit the redundant or obvious cases):

- $u \equiv \text{if } b \text{ then } (\text{if } b \text{ then } s \text{ else } t) \text{ else } w$ and $v \equiv \text{if } b \text{ then } s \text{ else } w$ then:

$$\begin{aligned}
\overline{\text{cond-st}}(u) \;&= \overline{\text{cond-st}}(s,t,w) \cup \overline{\text{cond-st}}(b) \cup \overline{\text{leave-st}}(b) \\
&\supseteq \overline{\text{cond-st}}(s,w) \cup \overline{\text{cond-st}}(b) \cup \overline{\text{leave-st}}(b) \\
&\supseteq \overline{\text{cond-st}}(v)
\end{aligned}$$

- $(u \equiv \text{if } b \text{ then } (\text{if } a \text{ then } s \text{ else } t) \text{ else } w, \; v \equiv \text{if } a \text{ then } (\text{if } b \text{ then } s \text{ else } w) \text{ else } (\text{if } b \text{ then } t \text{ else } w))$ and $(u \equiv \text{if } b \text{ then } s \text{ else } s, \, v \equiv s)$ are simple.
- $u \equiv \text{if } (\text{if } b \text{ then } a \text{ else } c) \text{ then } s \text{ else } t$ and $v \equiv \text{if } b \text{ then } (\text{if } a \text{ then } s \text{ else } t) \text{ else } (\text{if } c \text{ then } s \text{ else } t)$ then:

$$\overline{\text{cond-st}}(u) = \overline{\text{cond-st}}(b,a,c,s,t) \cup \overline{\text{leave-st}}(b,a,c) = \overline{\text{cond-st}}(v)$$

- $u \equiv f(\vec{w}, \text{if } b \text{ then } \vec{s} \text{ else } \vec{t})$ and $v \equiv \text{if } b \text{ then } f(\vec{w}, \vec{s}) \text{ else } f(\vec{w}, \vec{t})$ then:

$$\overline{\text{cond-st}}(u) = \overline{\text{cond-st}}(b, \vec{w}, \vec{s}, \vec{t}) \cup \overline{\text{leave-st}}(b) = \overline{\text{cond-st}}(v)$$

- $(u \equiv \pi_i(\langle s_1 \, , \, s_2 \rangle), \, v \equiv s_i)$ and $(u \equiv \text{dec}(\{m\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}, \mathsf{sk(n)}), \, v \equiv m)$ are trivial. ■

Let us show the following helpful propositions:

Proposition 15. *For all $\mathcal{S}_l$-normalized basic terms β, β' if:*

$$\overline{\text{leave-st}}(\beta) \cap \overline{\text{leave-st}}(\beta') \neq \emptyset$$

then we have $\mathcal{S}_l$-normalized basic terms $B[\vec{w}, (\alpha^j)_j, (\delta^k)_k], B[\vec{w}, (\alpha'^j)_j, (\delta'^k)_k]$ such that:

$$\begin{gathered}
\beta \equiv B[\vec{w}, (\alpha^j)_j, (\delta^k)_k] \quad \wedge \quad \beta' \equiv B[\vec{w}, (\alpha'^j)_j, (\delta'^k)_k] \\
\forall j, \; \overline{\text{leave-st}}(\alpha^j) \cap \overline{\text{leave-st}}(\alpha'^j) \neq \emptyset \quad \wedge \quad \forall k, \; \overline{\text{leave-st}}(\delta^k) \cap \overline{\text{leave-st}}(\delta'^k) \neq \emptyset
\end{gathered}$$

Proof. We have $\mathcal{S}_l$-normalized basic terms $B[\vec{w}, (\alpha^j)_j, (\delta^k)_k], B'[\vec{w}', (\alpha'^j)_j, (\delta'^k)_k]$ such that:

$$\beta \equiv B[\vec{w}, (\alpha^j)_j, (\delta^k)_k] \quad \wedge \quad \beta' \equiv B'[\vec{w}', (\alpha'^j)_j, (\delta'^k)_k]$$

Since β, β' are $\mathcal{S}_l$-normalized basic terms, we know that:

$$B[\vec{w}, (\{0\}^{\_}_{\_})_j, (\text{dec}(0, \_))_k] \quad \wedge \quad B'[\vec{w}', (\{[]_j\}^{\_}_{\_})_j, (\text{dec}([]_j, \_))_k]$$

are in R-normal form, and that $B, B', \vec{w}, \vec{w}'$ are if-free. Hence:

$$\begin{aligned}
\overline{\text{leave-st}}(\beta) &= \left\{ B[\vec{w}, (a^j)_j, (d^k)_k] \mid \forall j, a^j \in \overline{\text{leave-st}}(\alpha^j) \;\wedge\; \forall k, d^k \in \overline{\text{leave-st}}(\delta^k) \right\} \\
\overline{\text{leave-st}}(\beta') &= \left\{ B'[\vec{w}', (a'^j)_j, (d'^k)_k] \mid \forall j, a'^j \in \overline{\text{leave-st}}(\alpha'^j) \;\wedge\; \forall k, d'^k \in \overline{\text{leave-st}}(\delta'^k) \right\}
\end{aligned}$$

Similarly to what we did in the proof of Lemma 2, we prove that we can assume that $B \equiv B'$ by induction on the number of hole positions in B or B' such that $(B)_{|p}$ differs from $(B')_{|p}$ (modulo hole renaming). Knowing that $B \equiv B'$, it is then straightforward to show that:

$$\vec{w} \equiv \vec{w}' \quad \wedge \quad \forall j,\ \overline{\mathsf{leave\text{-}st}}(\alpha^j) \cap \overline{\mathsf{leave\text{-}st}}(\alpha'^j) \neq \emptyset \quad \wedge \quad \forall k,\ \overline{\mathsf{leave\text{-}st}}(\delta^k) \cap \overline{\mathsf{leave\text{-}st}}(\delta'^k) \neq \emptyset$$

The base case is trivial, let us prove the inductive case. We let p be the position of a hole in B such that p is a valid position in B', but not a hole (if p is not valid in B', invert B and B'). Let $B[\vec{w}, (a^j)_j, (d^k)_k]$ and $B'[\vec{w}', (a'^j)_j, (d'^k)_k]$ be such that:

$$\forall j, k.\, a^j \in \overline{\mathsf{leave\text{-}st}}(\alpha^j) \wedge d^k \in \overline{\mathsf{leave\text{-}st}}(\delta^k) \quad \wedge \quad \forall j, k.\, a'^j \in \overline{\mathsf{leave\text{-}st}}(\alpha'^j) \wedge d'^k \in \overline{\mathsf{leave\text{-}st}}(\delta'^k)$$

and:

$$B[\vec{w}, (a^j)_j, (d^k)_k] \equiv B'[\vec{w}', (a'^j)_j, (d'^k)_k] \in \overline{\mathsf{leave\text{-}st}}(\beta') \cap \overline{\mathsf{leave\text{-}st}}(\beta)$$

We then have three cases depending on $\beta_{|p}$:

- B contains a hole $[]_x$ at position p such that $\beta_{|p} \in \vec{w}$. Then let $\tilde{B}'$ be the context B' in which we replaced the term at position p by $[]_y$ (where y is a fresh hole variable) and let $\tilde{\vec{w}}'$ be the terms $\vec{w}'$ extended by $\beta_{|p}$ (binded to $[]_y$). Then B differs $\tilde{B}'$ on a smaller number of hole position, therefore we can conclude by induction hypothesis.
- B contains a hole $[]_x$ at position p such that $\beta_{|p}$ is an encryption oracle call $\{m\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n}_p)}$. Since $\{m\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n}_p)} \in \mathcal{E}_l$ is an encryption in an instance of a $\mathsf{CCA2}$ application, we know from the freshness side-condition that n_r does not appear in $\vec{w}$ and that $\mathsf{n}_r \in \mathcal{R}_l$.
  Moreover since β' is a $\mathcal{S}_l$-normalized basic term, we know that $\mathsf{fresh}(\mathcal{R}_l; \vec{w}')$. But since p is a valid non-hole position in B', we have $\mathsf{n}_r \in \vec{w}'$. Absurd.
- Similarly if B contains a hole $[]_x$ at position p such that $\beta_{|p}$ is a decryption oracle call $\mathsf{dec}(m, \mathsf{sk}(\mathsf{n}))$. Since $\mathsf{dec}(m, \mathsf{sk}(\mathsf{n}))$ is a decryption oracle call we know that $\mathsf{sk}(\mathsf{n}) \in \mathcal{K}_l$. Moreover since β' is a $\mathcal{S}_l$-normalized basic term, we know that $\mathsf{nodec}(\mathcal{K}_l, \vec{w}')$. But since p is a valid non-hole position in B', we know that either $\mathsf{sk}(\mathsf{n}) \in \vec{w}'$ or $\mathsf{n} \in \vec{w}'$. Absurd. ■

We can now state the following proposition, which subsumes Proposition 5.

Proposition 16. *For all $\mathcal{S}_l$-normalized basic terms β, β', if:*

$$\overline{\mathsf{leave\text{-}st}}(\beta) \cap \overline{\mathsf{leave\text{-}st}}(\beta') \neq \emptyset$$

then $\beta \equiv \beta'$.

Proof. We show this by induction on $|\beta| + |\beta'|$. Using Proposition 15 we know that we have $\mathcal{S}_l$-normalized basic terms $B[\vec{w}, (\alpha^j)_j, (\delta^k)_k], B[\vec{w}, (\alpha'^j)_j, (\delta'^k)_k]$ such that:

$$\beta \equiv B[\vec{w}, (\alpha^j)_j, (\delta^k)_k] \quad \wedge \quad \beta' \equiv B[\vec{w}, (\alpha'^j)_j, (\delta'^k)_k]$$
$$\forall j,\ \overline{\mathsf{leave\text{-}st}}(\alpha^j) \cap \overline{\mathsf{leave\text{-}st}}(\alpha'^j) \neq \emptyset \quad \wedge \quad \forall k,\ \overline{\mathsf{leave\text{-}st}}(\delta^k) \cap \overline{\mathsf{leave\text{-}st}}(\delta'^k) \neq \emptyset$$

To conclude we only need to show that for all j, $\overline{\mathsf{leave\text{-}st}}(\alpha^j) \cap \overline{\mathsf{leave\text{-}st}}(\alpha'^j) \neq \emptyset$ implies that $\alpha^j \equiv \alpha'^j$ and that $\overline{\mathsf{leave\text{-}st}}(\delta^k) \cap \overline{\mathsf{leave\text{-}st}}(\delta'^k) \neq \emptyset$ implies that $\delta^k \equiv \delta'^k$. The former is immediate, as $\overline{\mathsf{leave\text{-}st}}(\alpha^j) \cap \overline{\mathsf{leave\text{-}st}}(\alpha'^j) \neq \emptyset$ implies that $\alpha^j \equiv \{m\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}$ and $\alpha'^j \equiv \{m'\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n})}$. Since $\alpha^j, \alpha'^j \in \mathcal{E}_l$ and since there is *as most one* $\mathcal{S}_l$-encryption oracle call with the same randomness, we have $m \equiv m'$. It only remains to show that for all k, $\delta^k \equiv \delta'^k$. Since δ^k, δ'^k are $\mathcal{S}_l$-decryption oracle calls we know that

$$\delta^k \equiv C\left[\vec{g} \diamond (s_i)_{i \leq p}\right] \qquad \wedge \qquad \delta'^k \equiv C'\left[\vec{g}' \diamond (s'_i)_{i \leq p'}\right]$$

where:

- There exists contexts u, u', if-free and in R-normal form, such that:

$$\forall i < p,\ s_i \equiv \mathbf{0}(\mathsf{dec}(u[(\alpha_j)_j, (\mathsf{dec}_k)_k], \mathsf{sk})) \qquad\qquad s_p \equiv \mathsf{dec}(u[(\alpha_j)_j, (\mathsf{dec}_k)_k], \mathsf{sk})$$
$$\forall g \in \vec{g},\ g \equiv \mathsf{eq}(u[(\alpha_j)_j, (\mathsf{dec}_k)_k], \alpha_j)$$
$$\forall i < p',\ s'_i \equiv \mathbf{0}(\mathsf{dec}(u'[(\alpha'_j)_j, (\mathsf{dec}'_k)_k], \mathsf{sk}')) \qquad\qquad s'_p \equiv \mathsf{dec}(u'[(\alpha'_j)_j, (\mathsf{dec}'_k)_k], \mathsf{sk}')$$
$$\forall g \in \vec{g}',\ g \equiv \mathsf{eq}(u'[(\alpha'_j)_j, (\mathsf{dec}'_k)_k], \alpha'_j)$$

- $(\alpha_j)_j, (\alpha'_j)_j$ are *$\mathcal{S}_l$-encryption oracle calls.*
- $(\mathsf{dec}_k)_k, (\mathsf{dec}'_k)_k$ are *$\mathcal{S}_l$-decryption oracle call.*

Since $\overline{\mathsf{leave\text{-}st}}(\delta^k) \cap \overline{\mathsf{leave\text{-}st}}(\delta'^k) \neq \emptyset$, and since u, u' are if-free and in R-normal form we know that $u \equiv u'$, for all j, $\overline{\mathsf{leave\text{-}st}}(\alpha_j) \cap \overline{\mathsf{leave\text{-}st}}(\alpha'_j)$ and for all k, $\overline{\mathsf{leave\text{-}st}}(\mathsf{dec}_k) \cap \overline{\mathsf{leave\text{-}st}}(\mathsf{dec}'_k)$. It follows, by induction hypothesis, that for all j,

$\alpha_j \equiv \alpha'_j$ and for all k, $\mathsf{dec}_k \equiv \mathsf{dec}'_k$. We only have to check that the guards are the same. Since $\delta^k, \delta'^k \in \mathcal{D}_l$, we know from the definition of the $\mathsf{CCA2}$ axioms that δ^k (resp. δ'^k) has one guard for every encryption $\alpha \in \mathcal{E}_l$ such that $\alpha \equiv \{\_\}^{\mathsf{n}}_{\mathsf{pk}}$ and $\mathsf{n} \in \mathsf{st}(s_p \downarrow_R)\}$ (resp. $\mathsf{n} \in \mathsf{st}(s'_p \downarrow_R)\}$). Since we showed that $s_p \equiv s'_p$, we deduce that δ^k, δ'^k have the same guards. Since guards are sorted according to an arbitrary but fixed order (the sort function in the definition of $R^{\mathcal{K}}_{\mathsf{CCA2}_a}$), we know that $\delta^k \equiv \delta'^k$. ■

Corollary 1. *For all $P \vdash^{\mathsf{npf}} t \sim t'$, for all h, l:*

(i) for all $\beta, \beta' \leq_{\mathsf{c}}^{\mathsf{h},l} (t, P)$ if $\mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \mathsf{leave\text{-}st}(\beta' \downarrow_R) \neq \emptyset$ then $\beta \equiv \beta'$.
(ii) for all $\gamma, \gamma' \leq_{\mathsf{l}}^{\mathsf{h},l} (t, P)$ if $\mathsf{leave\text{-}st}(\gamma \downarrow_R) \cap \mathsf{leave\text{-}st}(\gamma' \downarrow_R) \neq \emptyset$ then $\gamma \equiv \gamma'$.
(iii) for all $\beta \leq_{\mathsf{c}}^{\mathsf{h},l} (t, P)$, $\gamma \leq_{\mathsf{l}}^{\mathsf{h},l} (t, P)$ if $\mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \mathsf{leave\text{-}st}(\gamma \downarrow_R) \neq \emptyset$ then $\beta \equiv \gamma$.

Lemma 16. *For all $P \vdash^{\mathsf{npf}} t \sim t'$, for all h, l and $\beta, \beta' \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P)$, there exists an almost conditional context $\tilde{\beta}'[]$ such that:*

$$\beta' \equiv \tilde{\beta}'[\beta] \quad \wedge \quad \mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \overline{\mathsf{cond\text{-}st}}\left(\tilde{\beta}'[]\right) = \emptyset$$

Proof. Let $l \in \mathsf{label}(P)$. We prove by mutual induction on the definition of $\mathcal{S}_l$-normalized simple terms, $\mathcal{S}_l$-normalized basic terms, $\mathcal{S}_l$-encryption oracle calls and $\mathcal{S}_l$-decryption oracle calls that for every $u \in \mathsf{st}(\beta')$ such that u is in one of the four above cases, there exists a conditional context $u_c[]$ such that:

$$u \equiv u_c[\beta] \quad \wedge \quad \mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \overline{\mathsf{cond\text{-}st}}(u_c[]) = \emptyset \quad \wedge \quad \overline{\mathsf{leave\text{-}st}}(\vec{u}_c) = \overline{\mathsf{leave\text{-}st}}(\vec{u})$$

Moreover if u is a $\mathcal{S}_l$-normalized basic term then there exists an almost conditional context $u_d[]$ such that:

$$u \equiv u_d[\beta] \quad \wedge \quad \mathsf{leave\text{-}st}(\beta \downarrow_R) \notin \overline{\mathsf{cond\text{-}st}}(u_d[]) \cup \overline{\mathsf{leave\text{-}st}}(u_d[])$$

- **Normalized Simple Term:** Let $u \equiv C[\vec{b} \diamond \vec{s}]$, where $\vec{b}$ are $\mathcal{S}_l$-normalized basic conditionals and $\vec{s}$ are $\mathcal{S}_l$-normalized basic terms. Let $\vec{b}_d[]$ and $\vec{s}_c[]$ be contexts obtained from $\vec{b}, \vec{s}$ by induction hypothesis such that $\vec{b}, \vec{s} \equiv \vec{b}_d[\beta], \vec{s}_c[\beta]$ and:

$$\mathsf{leave\text{-}st}(\vec{s}_c[]) = \mathsf{leave\text{-}st}(\vec{s}) \quad \wedge \quad \mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \left(\overline{\mathsf{cond\text{-}st}}\left(\vec{b}_d[], \vec{s}_c[]\right) \cup \overline{\mathsf{leave\text{-}st}}\left(\vec{b}_d[]\right)\right) = \emptyset$$

  Moreover:

$$\overline{\mathsf{cond\text{-}st}}(C[\vec{b}_d[] \diamond \vec{s}_c[]]) = \overline{\mathsf{cond\text{-}st}}(\vec{b}_d[], \vec{s}_c[]) \cup \overline{\mathsf{leave\text{-}st}}(\vec{b}_d[]) = \overline{\mathsf{cond\text{-}st}}(C[\vec{b} \diamond \vec{s}])$$
$$\overline{\mathsf{leave\text{-}st}}(C[\vec{b}_d[] \diamond \vec{s}_c[]]) = \overline{\mathsf{leave\text{-}st}}(\vec{s}_c[]) = \overline{\mathsf{leave\text{-}st}}(\vec{s}) = \overline{\mathsf{leave\text{-}st}}(C[\vec{b} \diamond \vec{s}])$$

  Hence we can take $\vec{u}_c \equiv C[\vec{b}_d[] \diamond \vec{s}_c[]]$.
- **Normalized Basic Term:** Let $u \equiv B[\vec{w}, (\alpha^i)_i, (\mathsf{dec}^j)_j]$ be a $\mathcal{S}_l$-normalized basic term. Let $(\alpha^i_c)_i, (\alpha^i_d)_i$ and $(\mathsf{dec}^j_c)_j, (\mathsf{dec}^j_d)_j$ be the contexts obtained by applying the induction hypothesis to $(\alpha^i)_i$ and $(\mathsf{dec}^j)_j$. Using the fact that:

$$\overline{\mathsf{leave\text{-}st}}\left((\alpha^i_c)_i, (\mathsf{dec}^j_c)_i\right) = \overline{\mathsf{leave\text{-}st}}\left((\alpha^i)_i, (\mathsf{dec}^j)_i\right)$$

  and since B and $\vec{w}$ are if-free, one can check that:

$$\overline{\mathsf{leave\text{-}st}}\left(B[\vec{w}, (\alpha^i_c)_i, (\mathsf{dec}^j_c)_j]\right) = \overline{\mathsf{leave\text{-}st}}\left(B[\vec{w}, (\alpha^i)_i, (\mathsf{dec}^j)_j]\right)$$

  It is then immediate to check that $u_c \equiv \underline{B[\vec{w}, (\alpha^i_c)_i}, (\mathsf{dec}^j_c)_j]$ satisfies the wanted properties. It remains to construct the context $u_d[]$: if for all, $\mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \overline{\mathsf{leave\text{-}st}}(u) = \emptyset$ then $u_d \equiv u_c$ satisfies the wanted properties. Otherwise using Proposition 16 we know that $\beta \equiv u$, hence we can take $u_d \equiv []$.
- **Encryption Oracle Call:** The proof done for the normalized basic term case applies here.
- **Decryption Oracle Call:** The proof done for the normalized simple term case applies here. ■

Observe that this lemma subsumes Lemma 15.

B. Well-nestedness

Definition 35. *A simple term $C[\vec{a} \diamond \vec{b}]$ is said to be* flat *if $\vec{a}, \vec{b}$ are if-free terms in R-normal forms.*

Definition 36. *We let* well-nested *be the smallest relation between sets $(\mathcal{C}, \mathcal{D})$ of flat simple terms such that:*

(a) *$(\mathcal{C}, \mathcal{D})$ is well-nested if for every $C_0[\vec{a}_0 \diamond \vec{b}_0] \in \mathcal{C}$:*

$$\begin{aligned} &\forall C[\vec{a} \diamond \vec{b}] \in \mathcal{C}, \quad \vec{b}_0 \cap \vec{a} = \emptyset \\ \textit{and} \quad &\forall D[\vec{c} \diamond \vec{t}] \in \mathcal{D}, \quad \vec{b}_0 \cap \vec{c} = \emptyset \end{aligned}$$

(b) *$(\mathcal{C}, \mathcal{D})$ is well-nested if for every $C_0[\vec{a}_0 \diamond \vec{b}_0] \in \mathcal{C}$:*

(i) *For all $C[\vec{a} \diamond \vec{b}] \in \mathcal{C}$, there exist two if-contexts C'^i, C''^i such that:*

$$C[\vec{a} \diamond \vec{b}] =_R \textsf{if}\; C_0[\vec{a}_0 \diamond \vec{b}_0] \;\textsf{then}\; C'^i[\vec{a}' \diamond \vec{b}'] \;\textsf{else}\; C''^i[\vec{a}'' \diamond \vec{b}'']$$

where $\vec{a}', \vec{a}'' \subseteq \vec{a} \backslash \vec{b}_0$ and $\vec{b}', \vec{b}'' \subseteq \vec{b}$.

(ii) *For every $D[\vec{c} \diamond \vec{t}] \in \mathcal{D}$, there exist two if-contexts D'^i, D'^i such that:*

$$D[\vec{c} \diamond \vec{t}] =_R \textsf{if}\; C_0[\vec{a}_0 \diamond \vec{b}_0] \;\textsf{then}\; D'^i[\vec{c}' \diamond \vec{t}'] \;\textsf{else}\; D''^i[\vec{c}'' \diamond \vec{t}'']$$

where $\vec{c}', \vec{c}'' \subseteq \vec{c} \backslash \vec{b}_0$ and $\vec{t}', \vec{t}'' \subseteq \vec{t}$.

(iii) *The following couples of sets are well-nested:*

$$\begin{gathered} \left(\left\{ C'^i[\vec{a}' \diamond \vec{b}'] \mid C[\vec{a} \diamond \vec{b}] \in \mathcal{C} \right\}, \left\{ D'^i[\vec{c}' \diamond \vec{t}'] \mid D[\vec{c} \diamond \vec{t}] \in \mathcal{D} \right\}\right) \\ \left(\left\{ C''^i[\vec{a}'' \diamond \vec{b}''] \mid C[\vec{a} \diamond \vec{b}] \in \mathcal{C} \right\}, \left\{ D''^i[\vec{c}'' \diamond \vec{t}''] \mid D[\vec{c} \diamond \vec{t}] \in \mathcal{D} \right\}\right) \end{gathered}$$

Proposition 17. *If $(\mathcal{C}, \mathcal{D})$ is such that for all $C_i[\vec{a}_i \diamond \vec{b}_i] \in \mathcal{C}$:*

$$\begin{aligned} &\forall C_j[\vec{a}_j \diamond \vec{b}_j] \in \mathcal{C}, \quad \vec{b}_i \cap \vec{a}_j = \emptyset \\ \textit{and} \quad &\forall D_j[\vec{c}_j \diamond \vec{t}_j] \in \mathcal{D}, \quad \vec{b}_i \cap \vec{c}_j = \emptyset \end{aligned}$$

Then $(\mathcal{C}, \mathcal{D})$ verifies the properties (i),(ii) and (iii) above.

Proof. Trivial by taking $C'^i_j \equiv C''^i_j \equiv C_j$. ∎

a) Main Lemma: We introduce now some tools used in the proof of the main lemma of this subsection, before stating and proving this lemma.

Definition 37. *We let $\textsf{pos}(t)$ be the set of positions of p, and $\textsf{head}$ be the partial function defined on terms such that for all $f \in \mathcal{F}$, for all terms $\vec{t}$, $\textsf{head}(f(\vec{t})) \equiv f$.*

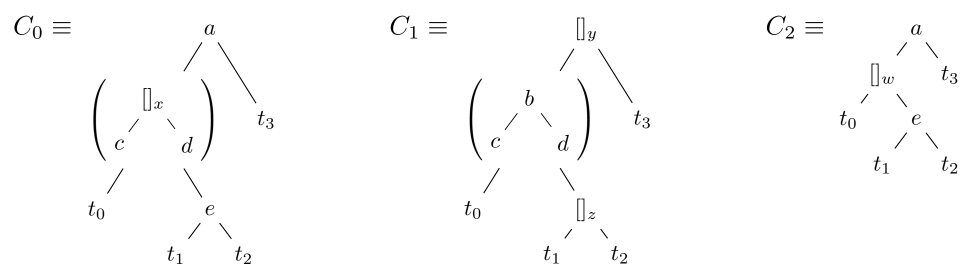
Definition 38. *For all conditional contexts C_0, C_1, we let $C_0 \sqcup C_1$ be the conditional context, if it exists, defined as follows: $\textsf{pos}(C_1 \sqcup C_2) = \textsf{pos}(C_0) \cap \textsf{pos}(C_1)$ and for all position p in $\textsf{pos}(C_0 \sqcup C_1)$:*

$$(C_0 \sqcup C_1)_{|p} \equiv \begin{cases} a & \textsf{if}\; \textsf{head}((C_0)_{|p}) \equiv \textsf{head}((C_1)_{|p}) \equiv a \qquad (a \in \mathcal{F} \cup \mathcal{N}) \\ []_x & \textsf{if}\; (C_0)_{|p} \equiv []_x \wedge \big(\textsf{head}((C_1)_{|p}) \equiv []_x \vee \textsf{head}((C_1)_{|p}) \equiv a\big) \qquad (a \in \mathcal{F} \cup \mathcal{N}) \\ []_x & \textsf{if}\; (C_1)_{|p} \equiv []_x \wedge \big(\textsf{head}((C_0)_{|p}) \equiv []_x \vee \textsf{head}((C_0)_{|p}) \equiv a\big) \qquad (a \in \mathcal{F} \cup \mathcal{N}) \end{cases}$$

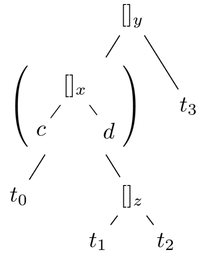
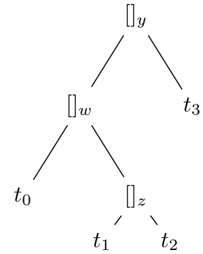
If such a conditional context does not exist then we let $C_0 \sqcup C_1$ be the special element $\textsf{undefined}$. We also let:

$$\textsf{undefined} \sqcup C_0 \equiv C_0 \sqcup \textsf{undefined} \equiv \textsf{undefined}$$

Example 8. For all conditionals a, b, c, d, e, f and terms $t_0, \ldots, t_3$, if we let:



Then we have:

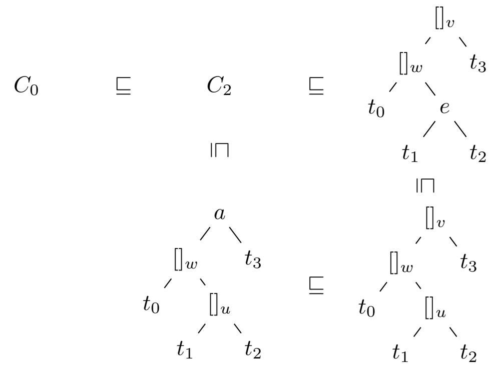
$C_0 \sqcup C_1 \equiv$  $C_1 \sqcup C_2 \equiv$  $C_0 \sqcup C_2 \equiv C_2$

Definition 39. *We let $\sqsubseteq$ be the relation on conditional contexts defined as follows: for all conditional contexts C_0, C_1, we let $C_0 \sqsubseteq C_1$ hold if $\textsf{pos}(C_1) \subseteq \textsf{pos}(C_0)$ and for all position p in $\textsf{pos}(C_1)$:*

$$\text{if } \textsf{head}((C_1)_{|p}) \equiv \begin{cases} a & \text{then } \textsf{head}((C_0)_{|p}) \equiv a \qquad (a \in \mathcal{F} \cup \mathcal{N}) \\ []_x & \text{then } \textsf{head}((C_0)_{|p}) \equiv a \vee \textsf{head}((C_0)_{|p}) \equiv []_x \qquad (a \in \mathcal{F} \cup \mathcal{N}) \end{cases}$$

Moreover we let $C_0 \sqsubseteq \textsf{undefined}$ for all conditional context C_0 (and $\textsf{undefined} \sqsubseteq \textsf{undefined}$).

Example 9. Using the conditional contexts defined in Example 8, we have, for example, the following relations:



Let $\mathcal{S}_{cc}$ be the set of conditional contexts extended with $\textsf{undefined}$.

Proposition 18. *$(\mathcal{S}_{cc}, \sqcup, \sqsubseteq)$ is a semi-lattice. That is we have the following properties:*

(i) $\sqcup$ is associative, commutative, idempotent.
(ii) $\sqsubseteq$ is an order (i.e. reflexive, transitive and antisymmetric).
(iii) For all $C_0, C_1 \in \mathcal{S}_{cc}$, we have $C_0 \sqsubseteq (C_0 \sqcup C_1)$ and $C_1 \sqsubseteq (C_0 \sqcup C_1)$. Moreover $(C_0 \sqcup C_1)$ is the least upper-bound of C_0 and C_1.

Proof. These properties are straightforward to show, we are only going to give the proof that $(C_0 \sqcup C_1)$ is the least upper-bound of C_0 and C_1. Assume that there is C such that:

$$C_0 \sqsubseteq C \sqsubseteq C_0 \sqcup C_1 \qquad\qquad C_1 \sqsubseteq C \sqsubseteq C_0 \sqcup C_1$$

If $C_0 \sqcup C_1 \equiv \textsf{undefined}$ then one can check that $C \equiv \textsf{undefined}$. Otherwise we know that $\textsf{pos}(C_0 \sqcup C_1) = \textsf{pos}(C_0) \cap \textsf{pos}(C_1)$, and that:

$$\textsf{pos}(C_0) \supseteq \textsf{pos}(C) \supseteq \textsf{pos}(C_0 \sqcup C_1) \qquad\qquad \textsf{pos}(C_1) \supseteq \textsf{pos}(C) \supseteq \textsf{pos}(C_0 \sqcup C_1)$$

Hence $\textsf{pos}(C) = \textsf{pos}(C_0 \sqcup C_1)$. Using the fact that $C \sqsubseteq C_0 \sqcup C_1$ we know that for all position $p \in \textsf{pos}(C)$, if $\textsf{head}((C_0 \sqcup C_1)_{|p}) = a$ (with $a \in \mathcal{F} \cup \mathcal{N}$) then $\textsf{head}(C_{|p}) = a$. If $\textsf{head}((C_0 \sqcup C_1)_{|p}) = []_x$ then either $\textsf{head}(C_{|p}) = []_x$ or $\textsf{head}(C_{|p}) = a$ (with $a \in \mathcal{F} \cup \mathcal{N}$). In the former case there is nothing to show, in the the latter case observe that $\textsf{head}((C_0 \sqcup C_1)_{|p}) = []_x$ implies that either $\textsf{head}((C_0)_{|p}) = []_x$ or $\textsf{head}((C_1)_{|p}) = []_x$. W.l.o.g assume $\textsf{head}((C_0)_{|p}) = []_x$. Then using the fact that $C_0 \sqsubseteq C$, we know that $\textsf{head}((C_0)_{|p}) = []_x$ implies that $\textsf{head}((C_0)_{|p}) = []_x$, absurd.

Therefore for all $p \in \textsf{pos}(C)$, $\textsf{head}(C_{|p}) = \textsf{head}((C_0 \sqcup C_1)_{|p})$. Moreover $\textsf{pos}(C) = \textsf{pos}(C_0 \sqcup C_1)$, hence $C \equiv C_0 \sqcup C_1$. ■

Proposition 19. *For all $C_0, C_1 \in \mathcal{S}_{cc}$, if $C_0 \sqsubseteq C_1$ and if:*

$$\forall p, p' \in \textsf{pos}(C_1), (C_1)_{|p} \equiv (C_1)_{|p'} \equiv []_x \;\Rightarrow\; (C_0)_{|p} \equiv (C_0)_{|p'}$$

then $\textsf{cond-st}(C_1 \downarrow_R) \cap \mathcal{T}(\mathcal{F}_s, \mathcal{N}) \subseteq \overline{\textsf{cond-st}}(C_0)$.

Proof. Assume that $C_0 \sqsubseteq C_1$, with $C_0, C_1 \neq \mathsf{undefined}$ (the case $C_0 \neq \mathsf{undefined}$ or $C_1 \neq \mathsf{undefined}$ is easy to handle, with the convention that $\mathsf{cond\text{-}st}(\mathsf{undefined}) = \emptyset$), and that:

$$\forall p, p' \in \mathsf{pos}(C_1), (C_1)_{|p} \equiv (C_1)_{|p'} \equiv []_x \;\Rightarrow\; (C_0)_{|p} \equiv (C_0)_{|p'} \tag{7}$$

First we show that we can extend this property as follows:

$$\forall p, p' \in \mathsf{pos}(C_1), (C_1)_{|p} \equiv (C_1)_{|p'} \;\Rightarrow\; (C_0)_{|p} \equiv (C_0)_{|p'} \tag{8}$$

Let $q = p \cdot q_0$ and $q = p' \cdot q_0$ be positions in $\mathsf{pos}(C_1)$. Since $(C_0)_{|p} \equiv (C_0)_{|p'}$, we know that $\mathsf{head}((C_1)_{|q}) \equiv \mathsf{head}((C_1)_{|q'})$.

- If $\mathsf{head}((C_1)_{|q}) \equiv a$ (with $a \in \mathcal{F} \cup \mathcal{N}$) then, from the fact that $C_0 \sqsubseteq C_1$ we get that $\mathsf{head}((C_0)_{|q}) \equiv a$, and that $\mathsf{head}((C_0)_{|q'}) \equiv a$.
- If $\mathsf{head}((C_1)_{|q}) \equiv []_x$ then using (7) we get that $(C_0)_{|p} \equiv (C_0)_{|p'}$.

Let $\to_{R'}$ be $\to_R$ without the non left-linear rules:

$$\mathsf{if}\ x\ \mathsf{then}\ y\ \mathsf{else}\ y \to y \qquad \mathsf{dec}(\{x\}^r_{\mathsf{pk}(y)}, \mathsf{sk}(y)) \to x \qquad \mathsf{if}\ w\ \mathsf{then}\ (\mathsf{if}\ w\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ z \to \mathsf{if}\ w\ \mathsf{then}\ x\ \mathsf{else}\ z$$

$$\mathsf{if}\ w\ \mathsf{then}\ x\ \mathsf{else}\ (\mathsf{if}\ w\ \mathsf{then}\ y\ \mathsf{else}\ z) \to \mathsf{if}\ w\ \mathsf{then}\ x\ \mathsf{else}\ z$$

We then mimic all reduction $\to_R$ on C_1 by a reduction on C_0, while maintaining $\sqsubseteq$ and the invariant of (7). Mimicking rules in $\to_R$ is easy as they are left-linear. To mimic rules in $(\to_R \setminus \to_{R'})$, we use (8). Formally, we show by induction on the length of the reduction sequence that for all C_1' such that $C_1 \to_R^* C_1'$, there exists C_0' such that $C_0' \sqsubseteq C_1'$, (7) holds for C_0', C_1' and $C_0 \to_R^* C_0'$.

Therefore let C_1' be in R-normal form such that $C_1 \to_R^* C_1'$. Let C_0' be such that $C_0' \sqsubseteq C_1'$,(7) holds for C_0', C_1' and $C_0 \to_R^* C_0'$. C_1' is of the form $D[\vec{b}, \vec{b}_{[]} \diamond \vec{u}]$ where $\vec{b}, \vec{u}$ are if-free and in R-normal form, $\vec{b}$ does not contain any hole variable and $\vec{b}_{[]}$ is a vector of hole variables. Therefore $\mathsf{cond\text{-}st}(C_1 \downarrow_R) \cap \mathcal{T}(\mathcal{F}_s, \mathcal{N}) = \mathsf{cond\text{-}st}(C_1') \cap \mathcal{T}(\mathcal{F}_s, \mathcal{N}) = \vec{b}$. We conclude by observing that $\vec{b} \subseteq \overline{\mathsf{cond\text{-}st}}(C_0')$, and that $\overline{\mathsf{cond\text{-}st}}(C_0') \subseteq \overline{\mathsf{cond\text{-}st}}(C_0)$ by Proposition 14. ∎

Lemma 17. *For all $P \vdash^{\mathsf{npf}} t \sim t'$, for all h, l, the following couple of sets is well-nested:*

$$\left(\left\{\beta \downarrow_R \mid \beta \leq_{\mathsf{c}}^{\mathsf{h},l} (t, P)\right\}, \left\{\gamma \downarrow_R \mid \gamma \leq_{\mathsf{l}}^{\mathsf{h},l} (t, P)\right\}\right)$$

Proof. We do this proof in the case $\mathsf{h} = \epsilon$. The other cases are identical.

We consider an arbitrary ordering $(\beta_i)_{1 \leq i \leq i_{max}}$ of $\{\beta \mid \beta \leq_{\mathsf{c}}^{\mathsf{h},l} (t, P)\}$ and $(\gamma_m)_{1 \leq m \leq m_{max}}$ of $\{\gamma \mid \gamma \leq_{\mathsf{l}}^{\mathsf{h},l} (t, P)\}$.

Using Lemma 16, we know that all $i \neq i_0$, there exists a conditional context $\tilde{\beta}_i$ such that:

$$\beta_i \equiv \tilde{\beta}_i\left[\beta_{i_o}\right] \quad \wedge \quad \mathsf{leave\text{-}st}(\beta_{i_0} \downarrow_R) \cap \overline{\mathsf{cond\text{-}st}}\left(\tilde{\beta}_{i,l}\right) = \emptyset$$

From now on we use $\beta_i^{(i_0)}$ to denote this conditional context, and $[]_{i_0}$ the hole variable used in the conditional contexts $\{\beta_i^{(i_0)} \mid i\}$. We similarly define $\gamma_m^{(i_0)}$ and we have:

$$\gamma_m \equiv \tilde{\gamma}_m\left[\beta_{i_o}\right] \quad \wedge \quad \mathsf{leave\text{-}st}(\beta_{i_0} \downarrow_R) \cap \overline{\mathsf{cond\text{-}st}}\left(\tilde{\gamma}_m\right) = \emptyset$$

We extend this notation by having j range between $-1 \leq j < n_{max}$ (resp. $-1 \leq j < m_{max}$), and having $\beta_i^{(-1)} \equiv \beta_i$ (resp. $\gamma_m^{(-1)} \equiv \gamma_m$).

Consider the following set $\mathcal{S}$:

$$\left\{\left((\sqcup_{j \leq n} \beta_i^{(l_j)})_i, (\sqcup_{j \leq n} \gamma_m^{(l_j)})_m\right) \mid (l_j)_j \text{ distinct indices} \wedge l_0 \equiv -1\right\}$$

Using Proposition 18.(iii) we know that for all $i \neq l_{j_0}$:

$$\beta_i^{(l_{j_0})} \sqsubseteq \sqcup_{j \leq n} \beta_i^{(l_j)} \qquad \wedge \qquad \beta_i^{(l_{j_0})} \sqsubseteq \sqcup_{j \leq n} \gamma_m^{(l_j)}$$

Using Proposition 19 we know that for all j, o and for all $i \neq l_{j_0}$:

$$\overline{\mathsf{cond\text{-}st}}\left(\beta_i^{(l_{j_0})}\right) \supseteq \mathsf{cond\text{-}st}\left(\sqcup_{j \leq n} \beta_i^{(l_j)} \downarrow_R\right) \qquad \wedge \qquad \overline{\mathsf{cond\text{-}st}}\left(\gamma_m^{(l_{j_0})}\right) \supseteq \mathsf{cond\text{-}st}\left(\sqcup_{j \leq n} \gamma_m^{(l_j)} \downarrow_R\right)$$

Which implies that:

$$\mathsf{leave\text{-}st}(\beta_{i_0} \downarrow_R) \cap \mathsf{cond\text{-}st}\left(\sqcup_{j \leq n} \beta_i^{(l_j)} \downarrow_R\right) = \emptyset \qquad \wedge \qquad \mathsf{leave\text{-}st}(\beta_{i_0} \downarrow_R) \cap \mathsf{cond\text{-}st}\left(\sqcup_{j \leq n} \gamma_m^{(l_j)} \downarrow_R\right) = \emptyset \tag{9}$$

Moreover it is quite simple to show that for all $(l_j)_{j \leq n+1}$, for all $i \neq l_{n+1}$:

$$\sqcup_{j \leq n+1} \beta_i^{(l_j)} \equiv \left(\sqcup_{j \leq n} \beta_i^{(l_j)}\right) \{[]_{l_{n+1}} / \sqcup_{j \leq n} \beta_{n+1}^{(l_j)}\}$$

Therefore:

$$\begin{aligned}\sqcup_{j\le n}\beta_i^{(l_j)} &=_R \qquad \left(\sqcup_{j\le n+1}\beta_i^{(l_j)}\right)\{\sqcup_{j\le n}\beta_{n+1}^{(l_j)}/[]_{l_{n+1}}\}\\ &=_R \textsf{if } \left(\sqcup_{j\le n}\beta_{n+1}^{(l_j)}\right) \textsf{ then } \left(\sqcup_{j\le n+1}\beta_i^{(l_j)}\right)\{\textsf{true}/[]_{l_{n+1}}\}\\ &\qquad\qquad\qquad\qquad \textsf{else } \left(\sqcup_{j\le n+1}\beta_i^{(l_j)}\right)\{\textsf{false}/[]_{l_{n+1}}\}\end{aligned}\tag{10}$$

Consider the following set $\mathcal{S}'$:

$$\left\{\left(\left(\sqcup_{j\le n}\beta_i^{(l_j)}\{e_j/[]_{l_j}\}\downarrow_R\right)_i,\left(\sqcup_{j\le n}\gamma_m^{(l_j)}\{e_j/[]_{l_j}\}\downarrow_R\right)_m\right) \mid (l_j)_j \text{ distinct indices} \wedge (e_j)_j \in \{\textsf{true},\textsf{false}\}^n\right\}$$

We show by decreasing induction on n, starting from $n = i_{max}+1$, that all the elements of $\mathcal{S}'$ are well-nested.

b) Base case: If $n = n_{max}+1$ then from (9) we get that for all sequence $(e_j)_j$ in $\{\textsf{true},\textsf{false}\}^n$, for all $j \neq i$:

$$\textsf{leave-st}(\beta_j\downarrow_R) \cap \textsf{cond-st}\left(\left(\sqcup_{j\le n}\beta_i^{(j)}\right)\{e_j/[]_j\}\downarrow_R\right) = \emptyset$$

Moreover we have:

$$\textsf{leave-st}\left(\left(\sqcup_{j\le n}\beta_i^{(j)}\right)\{e_j/[]_j\}\downarrow_R\right) \subseteq \{\beta_i^o \mid o\}$$

Hence we get that the following set is well-nested (case (a)):

$$\left(\left(\sqcup_{j\le n}\beta_i^{(j)}\{e_j/[]_j\}\downarrow_R\right)_i,\left(\sqcup_{j\le n}\gamma_m^{(j)}\{e_j/[]_j\}\downarrow_R\right)_m\right)$$

c) Inductive Case: If $n \le n_{max}$ then from (10) we get that for all sequence $(l_j)_{j\le n+1}$, for all sequence $(e_{l_j})_j$ in $\{\textsf{true},\textsf{false}\}^n$, for all $j \neq i$:

$$\begin{aligned}&\left(\sqcup_{j\le n}\beta_i^{(l_j)}\right)\{e_{l_j}/[]_{l_j} \mid j\le n\} =_R\\ &\quad \textsf{if } \left(\left(\sqcup_{j\le n}\beta_{l_{n+1}}^{(l_j)}\right)\{e_{l_j}/[]_{l_j} \mid j\le n\}\right) \textsf{ then } \left(\sqcup_{j\le n+1}\beta_i^{(l_j)}\right)\{e_{l_j}/[]_{l_j} \mid j\le n\}\{\textsf{true}/[]_{l_{n+1}}\}\\ &\quad\qquad \textsf{else } \left(\sqcup_{j\le n+1}\beta_i^{(l_j)}\right)\{e_{l_j}/[]_{l_j} \mid j\le n\}\{\textsf{false}/[]_{l_{n+1}}\}\end{aligned}$$

Let $e_{l_{n+1}} \equiv \textsf{true}$ (resp. $e_{l_{n+1}} \equiv \textsf{false}$). We get from (9) that for all o and $i \neq l_{n+1}$:

$$\textsf{leave-st}(\beta_{l_{n+1}}\downarrow_R) \cap \notin \textsf{cond-st}\left(\left(\sqcup_{j\le n+1}\beta_i^{(l_j)}\right)\{e_{l_j}/[]_{l_j}\}\downarrow_R\right) = \emptyset$$

We can do a similar reasoning on γ_i to show that for all o:

$$\textsf{leave-st}(\beta_{l_{n+1}}\downarrow_R) \cap \textsf{cond-st}\left(\left(\sqcup_{j\le n+1}\gamma_i^{(l_j)}\right)\{e_{l_j}/[]_{l_j}\}\downarrow_R\right) = \emptyset$$

Moreover by induction hypothesis we know that:

$$\left(\left(\left(\sqcup_{j\le n+1}\beta_i^{(l_j)}\right)\{e_{l_j}/[]_{l_j}\}\downarrow_R\right)_i,\left(\left(\sqcup_{j\le n+1}\gamma_i^{(l_j)}\right)\{e_{l_j}/[]_{l_j}\}\downarrow_R\right)_i\right)$$

is well-nested for $e_{l_{n+1}} \equiv \textsf{true}$ and $e_{l_{n+1}} \equiv \textsf{false}$. We deduce from this that the following set is well nested (case b):

$$\left(\left(\left(\sqcup_{j\le n}\beta_i^{(l_j)}\right)\{e_{l_j}/[]_{l_j}\}\downarrow_R\right)_i,\left(\left(\sqcup_{j\le n}\gamma_i^{(l_j)}\right)\{e_{l_j}/[]_{l_j}\}\downarrow_R\right)_i\right)$$

d) Conclusion: Recall that $\beta_i^{(l_0)} \equiv \beta_i^{(-1)} \equiv \beta_i$. We conclude the proof of this lemma by observing that

$$\left(\left\{C_i^{\mathsf{h}}\left[\vec{b}_i^{\mathsf{h}} \diamond \{\beta_i^{\mathsf{h},o} \mid o\}\right] \mid i\right\},\left\{C_m^{\mathsf{h}}\left[\vec{b}_m^{\mathsf{h}} \diamond \{\gamma_m^{\mathsf{h},o} \mid o\}\right] \mid m\right\}\right)$$

is the couple of sets:

$$\left(\left(\left(\sqcup_{j\le 0}\beta_i^{(l_j)}\right)\{e_{l_j}/[]_{l_j}\}\downarrow_R\right)_i,\left(\left(\sqcup_{j\le 0}\gamma_i^{(l_j)}\right)\{e_{l_j}/[]_{l_j}\}\downarrow_R\right)_i\right)$$

which is in $\mathcal{S}'$, and therefore well-nested. ∎

C. Spurious Conditionals

Definition 40. *An if-free conditional b is said to be* spurious *in a term t if $b \downarrow_R \notin \mathsf{cond\text{-}st}(t \downarrow_R)$.*

Definition 41. *A set of positions is said to be spurious in a term t if it is non-empty and $t[\mathit{true}/x \mid x \in S] =_R t[\mathit{false}/x \mid x \in S] =_R t$. A spurious set is* minimal *(resp.* maximal*) if it has not strict spurious subset (resp. overset), and a spurious set is* rooted *if there exists $p \in S$ such that $\forall p' \in S, p \leq p'$ (i.e. is a common ancestor of all positions in S).*

Example 10. Let $a \equiv \mathsf{eq}(A, 0)$ and $b \equiv \mathsf{eq}(B, 0)$ be two conditionals. Consider the following term t:

$$\begin{array}{l} \mathsf{if}\ b\ \mathsf{then}\ \ \mathsf{if}\ a\ \mathsf{then}\ \mathsf{if}\ b\ \mathsf{then}\ T\ \mathsf{else}\ U \\ \qquad\qquad\quad \mathsf{else}\ V \\ \qquad \mathsf{else}\ \ \mathsf{if}\ a\ \mathsf{then}\ T \\ \qquad\qquad\quad \mathsf{else}\ \mathsf{if}\ a\ \mathsf{then}\ V\ \mathsf{else}\ V \end{array}$$

Then the conditional b is spurious in t, since b is not a subterm of $t \downarrow_R \equiv \mathsf{if}\ a\ \mathsf{then}\ T\ \mathsf{else}\ V$. Moreover the conditional a is a subterm of $t \downarrow_R$, hence is spurious. Nonetheless, the set of position $S = \{220\}$ is spurious. Indeed we have:

$$\begin{array}{l} \mathsf{if}\ b\ \mathsf{then}\ \ \mathsf{if}\ a\ \mathsf{then}\ \mathsf{if}\ b\ \mathsf{then}\ T\ \mathsf{else}\ U \\ \qquad\qquad\quad \mathsf{else}\ V \\ \qquad \mathsf{else}\ \ \mathsf{if}\ a\ \mathsf{then}\ T \\ \qquad\qquad\quad \mathsf{else}\ \mathsf{if}\ \boxed{a}_{220}\ \mathsf{then}\ V\ \mathsf{else}\ V \end{array} =_R \begin{array}{l} \mathsf{if}\ b\ \mathsf{then}\ \ \mathsf{if}\ a\ \mathsf{then}\ \mathsf{if}\ b\ \mathsf{then}\ T\ \mathsf{else}\ U \\ \qquad\qquad\quad \mathsf{else}\ V \\ \qquad \mathsf{else}\ \ \mathsf{if}\ a\ \mathsf{then}\ T \\ \qquad\qquad\quad \mathsf{else}\ \mathsf{if}\ \boxed{\mathsf{true}}_{220}\ \mathsf{then}\ V\ \mathsf{else}\ V \end{array}$$

$$=_R \begin{array}{l} \mathsf{if}\ b\ \mathsf{then}\ \ \mathsf{if}\ a\ \mathsf{then}\ \mathsf{if}\ b\ \mathsf{then}\ T\ \mathsf{else}\ U \\ \qquad\qquad\quad \mathsf{else}\ V \\ \qquad \mathsf{else}\ \ \mathsf{if}\ a\ \mathsf{then}\ T \\ \qquad\qquad\quad \mathsf{else}\ \mathsf{if}\ \boxed{\mathsf{false}}_{220}\ \mathsf{then}\ V\ \mathsf{else}\ V \end{array}$$

a) Spurious Conditionals to Spurious Sets: Knowing that a conditional a is spurious in a term t does not necessarily mean that we know a spurious set of positions S such that for all $p \in S$, $t_{|p} \equiv a$. If b is in R-normal form this is easy, but terms in proof form are not in R-normal form. The following proposition shows that such a set of positions exists, under some conditions.

Proposition 20. *Let $\vec{a}, \vec{b}, \vec{c}$ be if-free conditionals in R-normal form. Let t be the term:*

$$t \equiv B\left[\vec{c} \diamond \left(\vec{w}, \mathit{if}\ C[\vec{b} \diamond \vec{a}]\ \mathit{then}\ u\ \mathit{else}\ v\right)\right]$$

Let $a \in \vec{a}$ be a spurious conditional in t such that:

- $a \notin \vec{b} \cup \{\mathit{true}, \mathit{false}\} \cup \mathsf{cond\text{-}st}(u \downarrow_R) \cup \mathsf{cond\text{-}st}(v \downarrow_R)$.
- $a \notin \rho$ *where ρ is the set of conditionals appearing on the path from the root to* ($\mathit{if}\ C[\vec{b} \diamond \vec{a}]\ \mathit{then}\ u\ \mathit{else}\ v$).

Then we have:

$$B\left[\vec{c} \diamond \left(\vec{w}, \mathit{if}\ C[\vec{b} \diamond \vec{a}]\ \mathit{then}\ u\ \mathit{else}\ v\right)\right] =_R B\left[\vec{c} \diamond \left(\vec{w}, \mathit{if}\ C[\vec{b} \diamond \vec{a}', \mathit{true}]\ \mathit{then}\ u\ \mathit{else}\ v\right)\right]$$

where $\vec{a}' = \vec{a} \backslash \{a\}$.

Proof. We recall that:

$$t \equiv B\left[\vec{c} \diamond \left(\vec{w}, \mathsf{if}\ C[\vec{b} \diamond \vec{a}]\ \mathsf{then}\ u\ \mathsf{else}\ v\right)\right]$$

We start with the simple observation that:

$$\begin{array}{rl} \mathsf{if}\ C[\vec{b} \diamond \vec{a}]\ \mathsf{then}\ u\ \mathsf{else}\ v\ =_R & \mathsf{if}\ a\ \mathsf{then}\ \mathsf{if}\ C[\vec{b} \diamond \vec{a}', \mathsf{true}]\ \mathsf{then}\ u\ \mathsf{else}\ v \\ & \qquad\quad \mathsf{else}\ \mathsf{if}\ C[\vec{b} \diamond \vec{a}', \mathsf{false}]\ \mathsf{then}\ u\ \mathsf{else}\ v \end{array}$$

Let $C_u[\vec{b}_u \diamond \vec{t}_u]$ and $C_v[\vec{b}_v \diamond \vec{t}_v]$ be the R-normal forms of u and v. Let C_l, C_r be such that :

$$\mathsf{if}\ C[\vec{b} \diamond \vec{a}', \mathsf{true}]\ \mathsf{then}\ u\ \mathsf{else}\ v =_R C_l[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a}' \diamond \vec{t}_u, \vec{t}_v]$$
$$\mathsf{if}\ C[\vec{b} \diamond \vec{a}', \mathsf{false}]\ \mathsf{then}\ u\ \mathsf{else}\ v =_R C_r[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a}' \diamond \vec{t}_u, \vec{t}_v]$$

Since $a \notin \text{cond-st}(u\downarrow_R), \text{cond-st}(v\downarrow_R)$ we know that $a \notin \vec{b}_u, \vec{b}_v$. Moreover since $\vec{a}' = \vec{a}\backslash\{a\}$ and $a \notin \vec{b}$ we know that $a \notin \vec{b}_u, \vec{b}_v, \vec{b}, \vec{a}'$. Therefore:

$$a \notin \text{cond-st}(C_l[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a}' \diamond \vec{t}_u, \vec{t}_v]) \quad \text{and} \quad a \notin \text{cond-st}(C_r[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a}' \diamond \vec{t}_u, \vec{t}_v]) \tag{11}$$

In a second time we get rid in C_l and C_r of all the conditionals appearing in ρ. We let $\vec{a}^{\mathsf{l}}$ and $\vec{a}^{\mathsf{r}}$ be such that:

$$\vec{a}^{\mathsf{l}} \subseteq \vec{b}_u, \vec{b}_v, \vec{b}, \vec{a}'\backslash\rho \quad \wedge \quad \vec{a}^{\mathsf{r}} \subseteq \vec{b}_u, \vec{b}_v, \vec{b}, \vec{a}'\backslash\rho \tag{12}$$

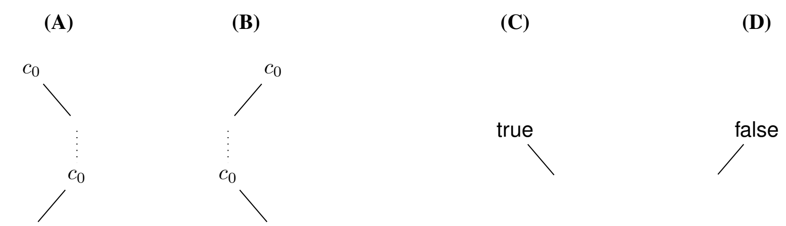
and C'_l, C'_r such that:

$$B\left[\vec{c} \diamond \left(\vec{w}, C_l[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a}' \diamond \vec{t}_u, \vec{t}_v]\right)\right] =_R B\left[\vec{c} \diamond \left(\vec{w}, C'_l[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v]\right)\right] \tag{13}$$

$$B\left[\vec{c} \diamond \left(\vec{w}, C_r[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a}' \diamond \vec{t}_u, \vec{t}_v]\right)\right] =_R B\left[\vec{c} \diamond \left(\vec{w}, C'_r[\vec{a}^{\mathsf{r}} \diamond \vec{t}_u, \vec{t}_v]\right)\right] \tag{14}$$

Therefore we deduce from (11) and (12) that $a \notin \vec{a}^{\mathsf{l}}$ and $a \notin \vec{a}^{\mathsf{r}}$.

b) Case 1: If there exists $c_0 \in \vec{c}$ such that the path ρ from the root of t to $\text{if } C[\vec{b} \diamond \vec{a}] \text{ then } u \text{ else } v$ contains one of the following shapes, where solid edges represent one element of the path ρ, and dotted edges represent a summary of a part of the path ρ.



In these four cases the result is easy to show. Since the proof are very similar we only describe case **(A)**: in that case we know that there exists a decomposition of B,$\vec{c}$ and $\vec{w}$ into, respectively, $B_1, \dots, B_5$, $\vec{c}_1, \dots, \vec{c}_5$ and $\vec{w}_1, \dots, \vec{w}_5$ such that:

$$B\left[\vec{c} \diamond \left(\vec{w}, \text{if } C[\vec{b} \diamond \vec{a}] \text{ then } u \text{ else } v\right)\right] \equiv$$
$$B_1\left[\vec{c}_1 \diamond \left(\vec{w}_1, \begin{array}{l} \text{if } c_0 \text{ then } B_2\left[\vec{c}_2 \diamond \vec{w}_2\right] \\ \text{else } B_3\left[\vec{c}_3 \diamond \left(\vec{w}_3, \begin{array}{l} \text{if } c_0 \text{ then } \boxed{B_4\left[\vec{c}_4 \diamond \left(\vec{w}_4, \text{if } C[\vec{b} \diamond \vec{a}] \text{ then } u \text{ else } v\right)\right]} \\ \text{else } B_5\left[\vec{c}_5 \diamond \vec{w}_5\right] \end{array}\right)\right] \end{array}\right)\right]$$

We can then rewrite the term $B_4\left[\vec{c}_4 \diamond \left(\vec{w}_4, \text{if } C[\vec{b} \diamond \vec{a}] \text{ then } u \text{ else } v\right)\right]$ using:

$$\text{if } b \text{ then } u \text{ else } (\text{if } b \text{ then } \mathbf{v} \text{ else } w) \to_R^* \text{if } b \text{ then } u \text{ else } (\text{if } b \text{ then } \mathbf{v}' \text{ else } w) \qquad (\text{ for all term } \mathbf{v}')$$

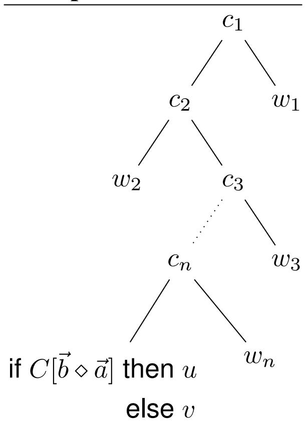
which yields the following term (we framed in red the part where the rewriting occurs):

$$B\left[\vec{c} \diamond \left(\vec{w}, \text{if } C[\vec{b} \diamond \vec{a}] \text{ then } u \text{ else } v\right)\right] =_R$$
$$B_1\left[\vec{c}_1 \diamond \left(\vec{w}_1, \begin{array}{l} \text{if } c_0 \text{ then } B_2\left[\vec{c}_2 \diamond \vec{w}_2\right] \\ \text{else } B_3\left[\vec{c}_3 \diamond \left(\vec{w}_3, \begin{array}{l} \text{if } c_0 \text{ then } \boxed{B_4\left[\vec{c}_4 \diamond \left(\vec{w}_4, \text{if } C[\vec{b} \diamond \vec{a}', \text{true}] \text{ then } u \text{ else } v\right)\right]} \\ \text{else } B_5\left[\vec{c}_5 \diamond \vec{w}_5\right] \end{array}\right)\right] \end{array}\right)\right]$$

c) Case 2:: Let s be such that $t \equiv s[\text{if } C[\vec{b} \diamond \vec{a}] \text{ then } u \text{ else } v]$. If none of the shapes of **Case 1** occurs, then we know that there exists B' such that $s =_R B'\left[\vec{c} \diamond (\vec{w}, [])\right]$ and the path ρ' from the root to $[]$ is a subset of ρ and does not contain duplicates, true and false. The existence of such a B' is proved by induction on the number of duplicate conditionals, true and false occurring on ρ': indeed since the shape (A) and (B) (resp. (C) and (D)) are forbidden in ρ, we know that if we have a duplicate (resp. true or false) then we can always rewrite B such that the hole containing s does not disappear.

Let $\rho' = c_1, \dots, c_n$. In a second time we are going to take B' as small as possible, i.e. only a branch $c_1, \dots, c_n$.

Example of if-context B':



Wet let $\vec{w} = w_1, \dots, w_n$, and we have:

$$s =_R B' \left[c_1, \dots, c_n \diamond w_1, \dots, w_n, []\right]$$

We let $\prec_u$ be a total ordering on if-free conditional in R-normal form such that the $n+1$ maximum elements are $c_1 \prec_u \cdots \prec c_n \prec_u a$. For all $1 \leq i \leq n$, we let $W_i[\vec{d_i} \diamond \vec{e_i}]$ be the $R_{\prec_u}$-normal form of w_i. We have:

$$s =_R B' \left[c_1, \dots, c_n \diamond \left(W_i[\vec{d_i} \diamond \vec{e_i}]\right)_{i \leq n}, []\right]$$

For all i, we let $W'_i[\vec{d}'_i \diamond \vec{e}'_i]$ be terms in R-normal form such that $\vec{d}'_i \cap \{c_j \mid j \leq i\} = \emptyset$ and:

$$s =_R B' \left[c_1, \dots, c_n \diamond \left(W'_i[\vec{d}'_i \diamond \vec{e}'_i]\right)_{i \leq n}, []\right]$$

Using (13) and (14) we get:

$$t =_R B' \left[c_1, \dots, c_n \diamond \left(W'_i[\vec{d}'_i \diamond \vec{e}'_i]\right)_{i \leq n}, \begin{array}{r} \text{if } a \text{ then } C'_l[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v] \\ \text{else } C'_r[\vec{a}^{\mathsf{r}} \diamond \vec{t}_u, \vec{t}_v] \end{array}\right]$$

It is then quite easy to show by induction on the length of the reduction sequence that there exists a sequence $1 \leq i_1 < \cdots < i_k \leq n$ and an if-context B'' such that:

$$\begin{aligned} &\left(B' \left[c_1, \dots, c_n \diamond \left(W'_i[\vec{d}'_i \diamond \vec{e}'_i]\right)_{i \leq n}, \begin{array}{r} \text{if } a \text{ then } C'_l[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v] \\ \text{else } C'_r[\vec{a}^{\mathsf{r}} \diamond \vec{t}_u, \vec{t}_v] \end{array}\right]\right) \downarrow_{R_{\prec_u}} \\ =_R \;& B'' \left[c_{i_1}, \dots, c_{i_k} \diamond \left(W'_{i_j}[\vec{d}'_{i_j} \diamond \vec{e}'_{i_j}]\right)_{j \leq k}, \left(\begin{array}{r} \text{if } a \text{ then } C'_l[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v] \\ \text{else } C'_r[\vec{a}^{\mathsf{r}} \diamond \vec{t}_u, \vec{t}_v] \end{array}\right) \downarrow_{R_{\prec_u}}\right] \end{aligned}$$

We deduce from this that a is spurious in:

$$\text{if } a \text{ then } C'_l[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v] \text{ else } C'_r[\vec{a}^{\mathsf{r}} \diamond \vec{t}_u, \vec{t}_v]$$

Since a will stay the top-most conditional in the R-normal form of this term (because of the order $\prec_u$ we chose), and since $a \neq \mathsf{true}$ $a \neq \mathsf{false}$ and $a \notin \vec{a}^{\mathsf{l}}, \vec{a}^{\mathsf{r}}$, there is only one rule that can be applied: if a then x else $x \to x$. Consequently:

$$C'_l[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v] \;=_R\; C'_r[\vec{a}^{\mathsf{r}} \diamond \vec{t}_u, \vec{t}_v]$$

Hence:

$$\begin{aligned} t &=_R B' \left[c_1, \dots, c_n \diamond \left(W'_i[\vec{d}'_i \diamond \vec{e}'_i]\right)_{i \leq n}, C'_l[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v]\right] \\ &=_R s\left[C'_l[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v]\right] \\ &=_R B\left[\vec{c} \diamond \left(\vec{w}, C'_l[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v]\right)\right] \end{aligned}$$

Hence using (13) we get:

$$t =_R B\left[\vec{c} \diamond \left(\vec{w}, C_l[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a}' \diamond \vec{t}_u, \vec{t}_v]\right)\right] =_R B\left[\vec{c} \diamond \left(\vec{w}, \text{if } C[\vec{b} \diamond \vec{a}', \mathsf{true}] \text{ then } u \text{ else } v\right)\right]$$ ∎

d) Properties of R:

Proposition 21. *For all simple term:*

$$B\left[\left(C_i[\vec{a}_i, a \diamond \vec{b}_i, a]\right)_i \diamond \left(D_j[\vec{c}_j, a \diamond \vec{t}_j]\right)_j\right]$$

such that $a, (\vec{a}_i, \vec{b}_i)_i, (\vec{c}_j, \vec{t}_j)_j$ are if-free and in R-normal form and $a \notin \vec{a}_i \cup \vec{b}_i \cup \vec{c}_j$, if:

$$t \in \text{leave-st}\left(\left(B\left[\left(C_i[\vec{a}_i, a \diamond \vec{b}_i, a]\right)_i \diamond \left(D_j[\vec{c}_j, a \diamond \vec{t}_j]\right)_j\right]\right) \downarrow_R\right)$$

then:

$$\begin{aligned} & t \in \text{leave-st}\left(\left(B\left[\left(C_i[\vec{a}_i, \mathit{true} \diamond \vec{b}_i, \mathit{true}]\right)_i \diamond \left(D_j[\vec{c}_j, \mathit{true} \diamond \vec{t}_j]\right)_j\right]\right) \downarrow_R\right) \\ \textit{or} \quad & t \in \text{leave-st}\left(\left(B\left[\left(C_i[\vec{a}_i, \mathit{false} \diamond \vec{b}_i, \mathit{false}]\right)_i \diamond \left(D_j[\vec{c}_j, \mathit{false} \diamond \vec{t}_j]\right)_j\right]\right) \downarrow_R\right) \end{aligned}$$

Proof. We know that:

$$\begin{aligned} & B\left[\left(C_i[\vec{a}_i, a \diamond \vec{b}_i, a]\right)_i \diamond \left(D_j[\vec{c}_j, a \diamond \vec{t}_j]\right)_j\right] \\ =_R \quad & \text{if } a \text{ then } \boxed{B\left[\left(C_i[\vec{a}_i, \mathsf{true} \diamond \vec{b}_i, \mathsf{true}]\right)_i \diamond \left(D_j[\vec{c}_j, \mathsf{true} \diamond \vec{t}_j]\right)_j\right]} \; B_{\mathsf{true}} \\ & \qquad \text{else } \boxed{B\left[\left(C_i[\vec{a}_i, \mathsf{false} \diamond \vec{b}_i, \mathsf{false}]\right)_i \diamond \left(D_j[\vec{c}_j, \mathsf{false} \diamond \vec{t}_j]\right)_j\right]} \; B_{\mathsf{false}} \end{aligned}$$

Let $\succ_u$ be a total order on if-free conditionals in R-normal form such that a is minimal. It is quite simple to show that:

$$\begin{aligned} & \left(\left(B\left[\left(C_i[\vec{a}_i, a \diamond \vec{b}_i, a]\right)_i \diamond \left(D_j[\vec{c}_j, a \diamond \vec{t}_j]\right)_j\right]\right) \downarrow_{R_{\succ_u}}\right) \\ \equiv & \begin{cases} \left(B\left[\left(C_i[\vec{a}_i, \mathsf{true} \diamond \vec{b}_i, \mathsf{true}]\right)_i \diamond \left(D_j[\vec{c}_j, \mathsf{true} \diamond \vec{t}_j]\right)_j\right]\right) \downarrow_{R_{\succ_u}} & \text{if } B_{\mathsf{true}} =_R B_{\mathsf{false}} \\ \text{if } a \text{ then } \left(\left(B\left[\left(C_i[\vec{a}_i, \mathsf{true} \diamond \vec{b}_i, \mathsf{true}]\right)_i \diamond \left(D_j[\vec{c}_j, \mathsf{true} \diamond \vec{t}_j]\right)_j\right]\right) \downarrow_{R_{\succ_u}}\right) & \text{otherwise} \\ \quad \text{else } \left(\left(B\left[\left(C_i[\vec{a}_i, \mathsf{false} \diamond \vec{b}_i, \mathsf{false}]\right)_i \diamond \left(D_j[\vec{c}_j, \mathsf{false} \diamond \vec{t}_j]\right)_j\right]\right) \downarrow_{R_{\succ_u}}\right) & \end{cases} \end{aligned}$$

The wanted result follows easily from Proposition 7 ∎

Proposition 22. *For all simple terms:*

$$C[\vec{a} \diamond \vec{b}] \qquad B^l\left[\left(C_i^l[\vec{a}_i^l \diamond \vec{b}_i^l]\right)_i \diamond \left(D_j^l[\vec{c}_j^l \diamond \vec{t}_j^l]\right)_j\right] \qquad B^r\left[\left(C_i^r[\vec{a}_i^r \diamond \vec{b}_i^r]\right)_i \diamond \left(D_j^r[\vec{c}_j^r \diamond \vec{t}_j^r]\right)_j\right]$$

such that:

- *For all $x \in \{l, r\}$, for all i, $(\vec{a}_i^x, \vec{b}_i^x, \vec{c}_i^x, \vec{t}_i^x)_i$ are if-free and in R-normal form.*
- *$\vec{a}, \vec{b}$ are if-free, in R-normal form and $(\vec{a} \cup \vec{b}) \cap \{\mathsf{true}, \mathsf{false}\} = \emptyset$.*
- *$\vec{b} \cap (\bigcup_{x \in \{l,r\}, i} \vec{a}_i^n, \vec{b}_i^n, \vec{c}_i^n) = \emptyset$.*
- *$\vec{a} \cap \vec{b} = \emptyset$.*

we have that for all $x \in \{l, r\}$:

$$\begin{aligned} & t \in \text{leave-st}\left(\left(B^x\left[\left(C_i^x[\vec{a}_i^x \diamond \vec{b}_i^x]\right)_i \diamond \left(D_j^x[\vec{c}_j^x \diamond \vec{t}_j^x]\right)_j\right]\right) \downarrow_R\right) \\ \implies \quad & t \in \text{leave-st}\left(\left(\begin{array}{r} \textit{if } C[\vec{a} \diamond \vec{b}] \textit{ then } B^l\left[\left(C_i^l[\vec{a}_i^l \diamond \vec{b}_i^l]\right)_i \diamond \left(D_j^l[\vec{c}_j^l \diamond \vec{t}_j^l]\right)_j\right] \\ \textit{else } B^r\left[\left(C_i^r[\vec{a}_i^r \diamond \vec{b}_i^r]\right)_i \diamond \left(D_j^r[\vec{c}_j^r \diamond \vec{t}_j^r]\right)_j\right] \end{array}\right) \downarrow_R\right) \end{aligned}$$

Proof. We prove this by induction on $|\vec{a}|$.

e) Base Case: The case $x = l$ and $x = r$ are exactly the same, therefore we assume that $x = l$. We have $C[\vec{a} \diamond \vec{b}] \equiv b$, where b is an if-free conditional. Let $\succ_u$ be any total order on if-free conditionals in R-normal form such that b is minimal. We then let $D^l[\vec{a}^l \diamond \vec{t}^l]$ and $D^r[\vec{a}^r \diamond \vec{t}^r]$ be the $R_{\succ_u}$-normal form of:

$$B^l\left[\left(C_i^l[\vec{a}_i^l \diamond \vec{b}_i^l]\right)_i \diamond \left(D_j^l[\vec{c}_j^l \diamond \vec{t}_j^l]\right)_j\right] \text{ and } B^r\left[\left(C_i^r[\vec{a}_i^r \diamond \vec{b}_i^r]\right)_i \diamond \left(D_j^r[\vec{c}_j^r \diamond \vec{t}_j^r]\right)_j\right]$$

Since:

$$t \in \text{leave-st}\left(\left(B^l\left[\left(C_i^l[\vec{a}_i^l \diamond \vec{b}_i^l]\right)_i \diamond \left(D_j^l[\vec{c}_j^l \diamond \vec{t}_j^l]\right)_j\right]\right) \downarrow_R\right)$$

we know by Proposition 7 that:

$$t \in \text{leave-st}\left(\left(D^l[\vec{a}^l \diamond \vec{t}^l]\right) \downarrow_{R_{\succ_u}}\right) \tag{15}$$

Using the fact that $(\vec{a}_i^l, \vec{b}_i^l, \vec{c}_i^l, \vec{t}_i^l)_i$ are if-free and in R-normal form, it is simple to show by induction on the length of the reduction that $\vec{a}^l \subseteq (\vec{a}_i^l, \vec{b}_i^l, \vec{c}_i^l)_i$. Together with the fact that $b \notin (\bigcup_{x \in \{l,r\}, i} \vec{a}_i^n, \vec{b}_i^n, \vec{c}_i^n)$, this shows that $b \notin \vec{a}^l$. Similarly $\vec{a}^r \subseteq (\vec{a}_i^r, \vec{b}_i^r, \vec{c}_i^r)_i$ and $b \notin \vec{a}^r$.

We know that:

$$\begin{array}{r} \text{if } b \text{ then } B^l\left[\left(C_i^l[\vec{a}_i^l \diamond \vec{b}_i^l]\right)_i \diamond \left(D_j^l[\vec{c}_j^l \diamond \vec{t}_j^l]\right)_j\right] \\ \text{else } B^r\left[\left(C_i^r[\vec{a}_i^r \diamond \vec{b}_i^r]\right)_i \diamond \left(D_j^r[\vec{c}_j^r \diamond \vec{t}_j^r]\right)_j\right] \end{array} \quad =_R \quad \underbrace{\begin{array}{r} \text{if } b \text{ then } D^l[\vec{a}^l \diamond \vec{t}^l] \\ \text{else } D^r[\vec{a}^r \diamond \vec{t}^r] \end{array}}_{s}$$

Since b is and if-free conditional in R-normal form minimal for $\succ_u$, since $D^l[\vec{a}^l \diamond \vec{t}^l]$ and $D^r[\vec{a}^r \diamond \vec{t}^r]$ are in $R_{\succ_u}$-normal form and since $b \notin \vec{a}^l \cup \vec{a}^r$, there is only one rule that may be applicable to s: $\text{if } b \text{ then } x \text{ else } x \rightarrow x$.

If the rule is not applicable then s is in $R_{\succ_u}$-normal form, (15) implies that $t \in \text{leave-st}(s \downarrow_{R_{\succ_u}})$, which by Proposition 7 shows that:

$$t \in \text{leave-st}\left(\left(\begin{array}{r}\text{if } C[\vec{a} \diamond \vec{b}] \text{ then } B^l\left[\left(C_i^l[\vec{a}_i^l \diamond \vec{b}_i^l]\right)_i \diamond \left(D_j^l[\vec{c}_j^l \diamond \vec{t}_j^l]\right)_j\right] \\ \text{else } B^r\left[\left(C_i^r[\vec{a}_i^r \diamond \vec{b}_i^r]\right)_i \diamond \left(D_j^r[\vec{c}_j^r \diamond \vec{t}_j^r]\right)_j\right]\end{array}\right)\downarrow_R\right)$$

If the rule is applicable then $s \downarrow_{R_{\succ_u}} \equiv D^l[\vec{a}^l \diamond \vec{t}^l]$. (15) implies that $t \in \text{leave-st}(s \downarrow_{R_{\succ_u}})$, which by Proposition 7 shows the wanted result.

f) Inductive Case: Assume that the result holds for m, and consider $\vec{a}$ of length $m+1$. Again w.l.o.g. we can take $x = l$. Let $a \in \vec{a}$, and $\vec{a}_0 = \vec{a} \backslash a$. We know that:

- There exist $C'[\vec{a}' \diamond \vec{b}']$ and $C''[\vec{a}'' \diamond \vec{b}'']$ such that:
$$C[\vec{a} \diamond \vec{b}] =_R \text{if } a \text{ then } C'[\vec{a}' \diamond \vec{b}'] \text{ else } C''[\vec{a}'' \diamond \vec{b}'']$$
with $\vec{a}' \cup \vec{a}'' \subseteq \vec{a}_0$ and $\vec{b}' \cup \vec{b}'' \subseteq \vec{b}$.
- For all $x \in \{l, r\}$, there exist $C_i'^x[\vec{a}_i'^x \diamond \vec{b}_i'^x]$ and $C_i''^x[\vec{a}_i''^x \diamond \vec{b}_i''^x]$ such that:
$$C_i^x[\vec{a}_i^x \diamond \vec{b}_i^x] =_R \text{if } a \text{ then } C_i'^x[\vec{a}_i'^x \diamond \vec{b}_i'^x] \text{ else } C_i''^x[\vec{a}_i''^x \diamond \vec{b}_i''^x]$$
with $\vec{a}_i'^x \cup \vec{a}_i''^x \subseteq \vec{a}_i^x \backslash \{a\}$ and $\vec{b}_i'^x \cup \vec{b}_i''^x \subseteq \vec{b}_i^x \cup \{\text{true}, \text{false}\} \backslash \{a\}$.
- For all $x \in \{l, r\}$, there exist $D'^x[\vec{c}_j'^x \diamond \vec{t}_j'^x]$ and $D''^x[\vec{c}_j''^x \diamond \vec{t}_j''^x]$ such that:
$$D_j^x[\vec{c}_j^x \diamond \vec{t}_j^x] =_R \text{if } a \text{ then } D_j'^x[\vec{c}_j'^x \diamond \vec{t}_j'^x] \text{ else } D_j''^x[\vec{c}_j''^x \diamond \vec{t}_j''^x]$$
with $\vec{c}_j'^x \cup \vec{c}_j''^x \subseteq \vec{c}_j^x \backslash \{a\}$ and $\vec{t}_j'^x \cup \vec{t}_j''^x \subseteq \vec{t}_j^x \cup \{\text{true}, \text{false}\} \backslash \{a\}$.

Using Proposition 21 we know that:

$$t \in \text{leave-st}\left(\left(B^l\left[\left(C_i'^l[\vec{a}_i'^l \diamond \vec{b}_i'^l]\right)_i \diamond \left(D_j'^l[\vec{c}_j'^l \diamond \vec{t}_j'^l]\right)_j\right]\right)\downarrow_R\right) \tag{16}$$
$$\text{or} \quad t \in \text{leave-st}\left(\left(B^l\left[\left(C_i''^l[\vec{a}_i''^l \diamond \vec{b}_i''^l]\right)_i \diamond \left(D_j''^l[\vec{c}_j''^l \diamond \vec{t}_j''^l]\right)_j\right]\right)\downarrow_R\right) \tag{17}$$

Assume that we are in Case (16) (the other case is exactly the same). We can then rewrite the initial term as follows:

$$\left.\begin{array}{r}\text{if } C[\vec{a} \diamond \vec{b}] \text{ then } B^l\left[\left(C_i^l[\vec{a}_i^l \diamond \vec{b}_i^l]\right)_i \diamond \left(D_j^l[\vec{c}_j^l \diamond \vec{t}_j^l]\right)_j\right] \\ \text{else } B^r\left[\left(C_i^r[\vec{a}_i^r \diamond \vec{b}_i^r]\right)_i \diamond \left(D_j^r[\vec{c}_j^r \diamond \vec{t}_j^r]\right)_j\right]\end{array}\right\} s$$

$$=_R \left.\begin{array}{r}\text{if } a \text{ then } \boxed{\begin{array}{r}\text{if } C'[\vec{a}' \diamond \vec{b}'] \text{ then } B^l\left[\left(C_i'^l[\vec{a}_i'^l \diamond \vec{b}_i'^l]\right)_i \diamond \left(D_j'^l[\vec{c}_j'^l \diamond \vec{t}_j'^l]\right)_j\right] \\ \text{else } B^r\left[\left(C_i'^r[\vec{a}_i'^r \diamond \vec{b}_i'^r]\right)_i \diamond \left(D_j'^r[\vec{c}_j'^r \diamond \vec{t}_j'^r]\right)_j\right]\end{array}}\ s_l \\ \text{else if } C''[\vec{a}'' \diamond \vec{b}''] \text{ then } B^l\left[\left(C_i''^l[\vec{a}_i''^l \diamond \vec{b}_i''^l]\right)_i \diamond \left(D_j''^l[\vec{c}_j''^l \diamond \vec{t}_j''^l]\right)_j\right] \\ \text{else } B^r\left[\left(C_i''^r[\vec{a}_i''^r \diamond \vec{b}_i''^r]\right)_i \diamond \left(D_j''^r[\vec{c}_j''^r \diamond \vec{t}_j''^r]\right)_j\right]\end{array}\right\} s'$$

We start by checking that the induction hypothesis can be applied to the red framed term s_l. The first two conditions are trivial, let us check the last two ones:

- Since $\vec{a}' \subseteq \vec{a}$ and $\vec{b}' \subseteq \vec{b}$, it is easy to check that $\vec{a}' \cap \vec{b}' = \emptyset$.
- Since:
$$\vec{a}_i'^x \subseteq \vec{a}_i^x \qquad \vec{b}_i'^x \subseteq \vec{b}_i^x \cup \{\text{true}, \text{false}\} \qquad \vec{c}_j'^x \subseteq \vec{c}_j^x$$
we know that:
$$\left(\bigcup\nolimits_{i, x \in \{l,r\}} \vec{a}_i'^x, \vec{b}_i'^x, \vec{c}_i'^x\right) \subseteq \left(\bigcup\nolimits_{i, x \in \{l,r\}} \vec{a}_i^x, \vec{b}_i^x, \vec{c}_i^x\right) \cup \{\text{true}, \text{false}\}$$
From the fact that $\vec{b} \cap (\bigcup_{x \in \{l,r\}, i} \vec{a}_i^x, \vec{b}_i^x, \vec{c}_i^x) = \emptyset$ and $\vec{b} \cap \{\text{true}, \text{false}\} = \emptyset$ we deduce that:
$$\vec{b} \cap \left(\bigcup\nolimits_{i, x \in \{l,r\}} \vec{a}_i'^x, \vec{b}_i'^x, \vec{c}_i'^x\right) = \emptyset$$

Finally since $\vec{b}' \subseteq \vec{b}$ we get:

$$\vec{b}' \cap \left(\bigcup_{i,x\in\{l,r\}} \vec{a}_i'^x, \vec{b}_i'^x, \vec{c}_i'^x\right) = \emptyset$$

Hence by induction hypothesis:

$$t \in \mathsf{leave\text{-}st}\left(\left(\begin{array}{r} \mathsf{if}\ C'[\vec{a}' \diamond \vec{b}']\ \mathsf{then}\ B^l\left[\left(C_i'^l[\vec{a}_i'^l \diamond \vec{b}_i'^l]\right)_i \diamond \left(D_j'^l[\vec{c}_j'^l \diamond \vec{t}_j'^l]\right)_j\right] \\ \mathsf{else}\ B^r\left[\left(C_i''^r[\vec{a}_i''^r \diamond \vec{b}_i''^r]\right)_i \diamond \left(D_j''^r[\vec{c}_j''^r \diamond \vec{t}_j''^r]\right)_j\right] \end{array}\right) \downarrow_R\right) \tag{18}$$

Moreover as $\vec{a}' \cup \vec{a}'' \subseteq \vec{a}_0 = \vec{a} \backslash \{a\}$ and $\vec{a} \cap \vec{b} = \emptyset$, we know that:

$$a \notin \vec{a}' \cup \vec{a}'' \cup \vec{b}' \cup \vec{b}'' \cup \left(\bigcup_{i,x\in\{l,r\}} \vec{a}_i^n, \vec{b}_i^n, \vec{c}_i^n\right)$$

Since:

$$\vec{a}_i'^x \cup \vec{a}_i''^x \subseteq \vec{a}_i^x \qquad \vec{b}_i'^x \cup \vec{b}_i''^x \subseteq \vec{b}_i^x \cup \{\mathsf{true}, \mathsf{false}\} \qquad \vec{c}_j'^x \cup \vec{c}_j''^x \subseteq \vec{c}_j^x$$

and using the fact that $a \notin \{\mathsf{true}, \mathsf{false}\}$, we get from (18) that:

$$a \notin \vec{a}' \cup \vec{a}'' \cup \vec{b}' \cup \vec{b}'' \cup \left(\bigcup_{i,x\in\{l,r\}} \vec{a}_i'^n, \vec{b}_i'^n, \vec{c}_i'^n\right) \cup \left(\bigcup_{i,x\in\{l,r\}} \vec{a}_i''^n, \vec{b}_i''^n, \vec{c}_i''^n\right)$$

Hence we can apply again the induction hypothesis (with $m = 1$) to s', which shows that $t \in \mathsf{leave\text{-}st}(s' \downarrow_R) \equiv \mathsf{leave\text{-}st}(s \downarrow_R$). ■

g) Sufficient Conditions for Non Spuriousness of Leaves: We now give sufficient conditions to show that a leave term is not spurious.

Proposition 23. *For all simple term:*

$$s \equiv A\left[\vec{d} \diamond \left(B_l\left[(\beta_{i,l})_i \diamond (\gamma_{j,l})_j\right]\right)_l\right]$$

such that:

(i) $\vec{d}$ *are if-free and in R-normal form, and for all* i, j, l, $\mathsf{cond\text{-}st}(\beta_{i,l} \downarrow_R) \cap \overrightarrow{\mathsf{leave\text{-}st}}(\beta_{i,l} \downarrow_R) = \emptyset$.

(ii) $\left(\vec{d} \cup \bigcup_i \mathsf{leave\text{-}st}(\beta_{i,l} \downarrow_R)\right) \cap \{\mathit{true}, \mathit{false}\} = \emptyset$.

(iii) *For every positions* $p < p'$ *in* $A[\_ \diamond (B_l)_l]$ *such that* $s_{|p} \equiv \zeta$ *and* $s_{|p'} \equiv \zeta'$, *we have* $\mathsf{leave\text{-}st}(\zeta \downarrow_R) \cap \mathsf{leave\text{-}st}(\zeta' \downarrow_R) = \emptyset$.

(iv) *For all* l, *for all* i, j, $\mathsf{leave\text{-}st}(\beta_{i,l} \downarrow_R) \cap \mathsf{leave\text{-}st}(\beta_{j,l} \downarrow_R) \neq \emptyset$ *implies that* $\beta_{i,l} \equiv \beta_{j,l}$.

(v) *For all* l, *the following couple of sets is well-nested:*

$$\left(\{\beta_{i,l} \downarrow_R \mid i\}, \{\gamma_{j,l} \downarrow_R \mid j\}_j\right)$$

for all l, j, *there exists* $t \in \vec{t}_{j,l}$ *such that* $t \in \mathsf{leave\text{-}st}(s \downarrow_R)$.

Proof. For all l, i, j, we let $C_{i,l}[]$, $D_{j,l}[]$ be if-contexts and $\vec{a}_{i,l}$, $\vec{b}_{i,l}, \vec{c}_{j,l}$, $\vec{t}_{j,l}$ be if-free terms in R-normal form such that:

$$\vec{a}_{i,l} \equiv \mathsf{cond\text{-}st}(\beta_{i,l} \downarrow_R) \qquad \vec{b}_{i,l} \equiv \mathsf{leave\text{-}st}(\beta_{i,l} \downarrow_R) \qquad \vec{c}_{i,l} \equiv \mathsf{cond\text{-}st}(\gamma_{j,l} \downarrow_R) \qquad \vec{t}_{i,l} \equiv \mathsf{leave\text{-}st}(\gamma_{j,l} \downarrow_R)$$

$$\beta_{i,l} \downarrow_R \equiv\ C_{i,l}[\vec{a}_{i,l} \diamond \vec{b}_{i,l}] \qquad \gamma_{j,l} \downarrow_R \equiv\ D_{j,l}[\vec{c}_{j,l} \diamond \vec{t}_{j,l}]$$

We start by showing that this is the case if $\vec{d} = \emptyset$ and $A \equiv []$ in the first part of the proof, and then will deal with the general case in the second part.

h) Part 1: Since $\vec{d} = \emptyset$ we know that:

$$s \equiv B\left[\left(C_i[\vec{a}_i \diamond \vec{b}_i]\right)_i \diamond \left(D_j[\vec{c}_j \diamond \vec{t}_j]\right)_j\right]$$

satisfying conditions (i) to (v).

We let $\mathsf{nested\text{-}if}(B)$ be the maximum number of nested $\mathsf{if\ then\ else}$, and $\vec{a}_0$ be the conditionals of the basic conditional at the root of B. We prove the proposition by induction on ($\mathsf{nested\text{-}if}(B)$,$|\vec{a}_0|$), ordered with the lexicographic ordering.

i) Part 1: Base Case: The base case is simple: it suffices to notice that since $\vec{c}, \vec{t}$ are if-free and in R-normal form:

$$\mathsf{leave\text{-}st}(s \downarrow_R) = \mathsf{leave\text{-}st}(D[\vec{c} \diamond \vec{t}] \downarrow_R) \subseteq \vec{t}$$

This is a simple proof by induction on the length of the reduction sequence.

j) Part 1: First Inductive Case: Assume that the property holds for (n,ω) and lets show that it holds for $(n+1,0)$. Consider:

$$\begin{aligned} s \equiv \textsf{if } b_0 \textsf{ then } & B^l\left[\left(C_i^l[\vec{a}_i^l \diamond \vec{b}_i^l]\right)_i \diamond \left(D_j^l[\vec{c}_j^l \diamond \vec{t}_j^l]\right)_j\right] \\ \textsf{else } & B^r\left[\left(C_i^r[\vec{a}_i^r \diamond \vec{b}_i^r]\right)_i \diamond \left(D_j^r[\vec{c}_j^r \diamond \vec{t}_j^r]\right)_j\right] \end{aligned}$$

where B^l and B^r are such that $\textsf{nested-if}(B^l) \leq n$ and $\textsf{nested-if}(B^r) \leq n$. Using the well-nested condition, we know that for all $i \neq 0, x \in \{l,r\}$, there exists two if-context $C_i'^x, C_i''^x$ such that:

$$C_i^x[\vec{a}_i^x \diamond \vec{b}_i^x] =_R \textsf{if } b_0 \textsf{ then } C_i'^x[\vec{a}_i'^x \diamond \vec{b}_i'^x] \textsf{ else } C_i''^x[\vec{a}_i''^x \diamond \vec{b}_i''^x]$$

where $\vec{a}_i'^x, \vec{a}_i''^x \subseteq \vec{a}_i^x \backslash b_0$ and $\vec{b}_i'^x, \vec{b}_i''^x \subseteq \vec{b}_i^x$. Similarly for all $j, x \in \{l,r\}$, we know that there exists two if-context $D_j'^x, D_j''^x$ such that:

$$D_j^x[\vec{c}_j^x \diamond \vec{t}_j^x] =_R \textsf{if } b_0 \textsf{ then } D_j'^x[\vec{c}_j'^x \diamond \vec{t}_j'^x] \textsf{ else } D_j''^x[\vec{c}_j''^x \diamond \vec{t}_j''^x]$$

where $\vec{c}_j'^x, \vec{c}_j''^x \subseteq \vec{c}_j^x \backslash b_0$ and $\vec{t}_j'^x, \vec{t}_j''^x \subseteq \vec{t}_j^x$. We can rewrite the term s as follows:

$$\begin{aligned} s \equiv \textsf{if } b_0 \textsf{ then } & \boxed{B^l\left[\left(C_i''^l[\vec{a}_i''^l \diamond \vec{b}_i''^l]\right)_i \diamond \left(D_j''^l[\vec{c}_j''^l \diamond \vec{t}_j''^l]\right)_j\right]}\; s_l \\ \textsf{else } & \boxed{B^r\left[\left(C_i''^r[\vec{a}_i''^r \diamond \vec{b}_i''^r]\right)_i \diamond \left(D_j''^r[\vec{c}_j''^r \diamond \vec{t}_j''^r]\right)_j\right]}\; s_r \end{aligned}$$

Using the induction hypothesis on the framed term s_l (resp. s_r), we know that for all j, there exists $t \in \vec{t}_j''^l \subseteq \vec{t}_j^l$ (resp. $t \in \vec{t}_j''^r \subseteq \vec{t}_j^r$) such that:

$$\begin{aligned} & t \in \textsf{leave-st}\left(B^l\left[\left(C_i''^l[\vec{a}_i''^l \diamond \vec{b}_i''^l]\right)_i \diamond \left(D_j''^l[\vec{c}_j''^l \diamond \vec{t}_j''^l]\right)_j\right]\right)\downarrow_R \\ \Big(\text{resp.}\quad & t \in \textsf{leave-st}\left(B^r\left[\left(C_i''^r[\vec{a}_i''^r \diamond \vec{b}_i''^r]\right)_i \diamond \left(D_j''^r[\vec{c}_j''^r \diamond \vec{t}_j''^r]\right)_j\right]\right)\downarrow_R\Big) \end{aligned}$$

We now want to apply Proposition 22 to show that $t \in \textsf{leave-st}(s\downarrow_R)$. The only difficulty lies in showing that:

$$b_0 \cap \left(\bigcup_i \vec{a}_i''^l, \vec{a}_i''^r, \vec{b}_i''^l, \vec{b}_i''^r, \vec{c}_i''^l, \vec{c}_i''^r\right) = \emptyset$$

We know that $b_0 \cap \left(\bigcup_i \vec{a}_i''^l, \vec{a}_i''^r, \vec{c}_i''^l, \vec{c}_i''^r\right) = \emptyset$ (since $\vec{a}_i''^l \subseteq \vec{a}_i^l \backslash \{b_0\}, \dots$), so it only remains to show that $b_0 \notin \bigcup_i \vec{b}_i''^l, \vec{b}_i''^r$. This follows from the hypothesis (iii), since b_0 is at the root of B and therefore for all i, $b_0 \notin \vec{b}_i^l \supseteq \vec{b}_i''^l$ (resp. $b_0 \notin \vec{b}_i^r \supseteq \vec{b}_i''^r$).

k) Part 1: Second Inductive Case: Now assume that the property holds for $(n+1,k)$ and lets show that it holds for $(n+1,k+1)$. Consider:

$$\begin{aligned} s \equiv \textsf{if } C_0[\vec{a}_0 \diamond \vec{b}_0] \textsf{ then } & B^l\left[\left(C_i[\vec{a}_i \diamond \vec{b}_i]\right)_{i\in I^l} \diamond \left(D_j[\vec{c}_j \diamond \vec{t}_j]\right)_{j\in J^l}\right] \\ \textsf{else } & B^r\left[\left(C_i[\vec{a}_i \diamond \vec{b}_i]\right)_{i\in I^r} \diamond \left(D_j[\vec{c}_j \diamond \vec{t}_j]\right)_{j\in J^r}\right] \end{aligned}$$

where B^l and B^r are such that of $\textsf{nested-if}(B^l) \leq n$, $\textsf{nested-if}(B^r) \leq n$, and $|\vec{a}_0| = k+1$.

We are looking for m such that for all j, $\vec{a}_m \cap \vec{b}_j = \emptyset$, $\vec{b}_m \subseteq \vec{a}_0$ and $\vec{a}_m, \vec{b}_m \subseteq \vec{a}_0, \vec{b}_0$.

- If there exists k_0 such that $\vec{a}_0 \cap \vec{b}_{k_0} \neq \emptyset$ then we know that $\vec{a}_{k_0}, \vec{b}_{k_0} \subseteq \vec{a}_0$ and $\vec{a}_{k_0}, \vec{b}_{k_0} \subset \vec{a}_0, \vec{b}_0$. We repeat this process and build a sequence $(k_l)_l$ such that for all l, $\vec{a}_{k_{l+1}}, \vec{b}_{k_{l+1}} \subseteq \vec{a}_{k_l}$ and $\vec{a}_{k_{l+1}}, \vec{b}_{k_{l+1}} \subset \vec{a}_{k_l}, \vec{b}_{k_l}$.
  This sequence is necessarily finite. Let l_{max} it length and let $m = k_{l_{max}-1}$. We know that for all j, $\vec{a}_m \cap \vec{b}_j = \emptyset$ (otherwise we could extend the sequence). Moreover we know that $\vec{b}_m \subseteq \vec{a}_0$ and $\vec{a}_m, \vec{b}_m \subset \vec{a}_0, \vec{b}_0$.
- If for all k_0, $\vec{a}_0 \cap \vec{b}_{k_0} = \emptyset$ then we take $m = 0$.

Using the well-nested hypothesis, we know that for all $j \in I^l \cup I^r$, there exist two if-context C_j', C_j'' such that:

$$C_j[\vec{a}_j \diamond \vec{b}_j] =_R \textsf{if } C_m[\vec{a}_m \diamond \vec{b}_m] \textsf{ then } C_j'[\vec{a}_j' \diamond \vec{b}_j'] \textsf{ else } C_j''[\vec{a}_j'' \diamond \vec{b}_j'']$$

where $\vec{a}_j', \vec{a}_j'' \subseteq \vec{a}_j \backslash \vec{b}_m$ and $\vec{b}_j', \vec{b}_j'' \subseteq \vec{b}_j$. Similarly there exist two if-context D_j', D_j'' such that:

$$D_j[\vec{c}_j \diamond \vec{t}_j] =_R \textsf{if } C_m[\vec{a}_m \diamond \vec{b}_m] \textsf{ then } D_j'[\vec{c}_j' \diamond \vec{t}_j'] \textsf{ else } C_j''[\vec{c}_j'' \diamond \vec{t}_j'']$$

where $\vec{c}_j', \vec{c}_j'' \subseteq \vec{a}_j \backslash \vec{b}_m$ and $\vec{t}_j', \vec{t}_j'' \subseteq \vec{t}_j$.

We let B^l_{true} (resp. B^r_{true}) be the if-context obtained from B^l (resp. B^r) by replacing every conditional hole $[]_i$ that is mapped to $C_m[\vec{a}_m \diamond \vec{b}_m]$ in s by its then branch. Similarly we define B^l_{false} (resp. B^r_{false}) by replacing every conditional hole $[]_i$ that is mapped to $C_m[\vec{a}_m \diamond \vec{b}_m]$ in s by its else branch. By consequence:

$$
\begin{aligned}
s \equiv \mathsf{if}\ C_m[\vec{a}_m \diamond \vec{b}_m]\ \mathsf{then}\ & \boxed{\begin{aligned} &\mathsf{if}\ C'_0[\vec{a}'_0 \diamond \vec{b}'_0]\ \mathsf{then}\ B^l_{\mathsf{true}}\left[\left(C'_i[\vec{a}'_i \diamond \vec{b}'_i]\right)_{i \in I^l_{\mathsf{true}}} \diamond \left(D'_j[\vec{c}'_j \diamond \vec{t}'_j]\right)_{j \in J^l_{\mathsf{true}}}\right] \\ &\qquad\qquad \mathsf{else}\ B^r_{\mathsf{true}}\left[\left(C'_i[\vec{a}'_i \diamond \vec{b}'_i]\right)_{i \in I^r_{\mathsf{true}}} \diamond \left(D'_j[\vec{c}'_j \diamond \vec{t}'_j]\right)_{j \in J^r_{\mathsf{true}}}\right] \end{aligned}} \ s_{\mathsf{true}} \\
\mathsf{else}\ & \boxed{\begin{aligned} &\mathsf{if}\ C''_0[\vec{a}''_0 \diamond \vec{b}''_0]\ \mathsf{then}\ B^l_{\mathsf{false}}\left[\left(C''_i[\vec{a}''_i \diamond \vec{b}''_i]\right)_{i \in I^l_{\mathsf{false}}} \diamond \left(D''_j[\vec{c}''_j \diamond \vec{t}''_j]\right)_{j \in J^l_{\mathsf{false}}}\right] \\ &\qquad\qquad \mathsf{else}\ B^r_{\mathsf{false}}\left[\left(C''_i[\vec{a}''_i \diamond \vec{b}''_i]\right)_{i \in I^r_{\mathsf{false}}} \diamond \left(D''_j[\vec{c}''_j \diamond \vec{t}''_j]\right)_{j \in J^r_{\mathsf{false}}}\right] \end{aligned}} \ s_{\mathsf{false}}
\end{aligned}
$$

We then have the following property: $J^l = J^l_{\mathsf{true}} \cup J^l_{\mathsf{false}}$, and $J^r = J^r_{\mathsf{true}} \cup J^r_{\mathsf{false}}$.

We want to show that for all $j \in J^l \cup J^r$, $\exists t \in \vec{t}_j.\, t \in \mathsf{leave\text{-}st}(s \downarrow_R)$. Let $j \in J^l$ (the proof for $j \in J^r$ is similar), then either $j \in J^l_{\mathsf{true}}$ or $j \in J^l_{\mathsf{false}}$. In the former case we apply the induction hypothesis to s_{true}, and in the latter to s_{false}. Lets check that the premises hold for s_{true} (the same proof works for s_{false}):

- (i) and (ii) trivially hold.
- (iii) is simple, as we only removed some nodes from the if-context and (iii) is stable by embedding.
- Checking that (iv) holds is straightforward. Assume that there exists $i, j \in I^l_{\mathsf{true}} \cup I^r_{\mathsf{false}} \cup \{0\}$ such that $\vec{b}'_i \cap \vec{b}'_j \neq \emptyset$. Since $\vec{b}'_i \subseteq \vec{b}_i$ and $\vec{b}'_j \subseteq \vec{b}_j$ we know that $\vec{b}_i \cap \vec{b}_j \neq \emptyset$. Therefore $C_i[\vec{a}_i \diamond \vec{b}_i] \equiv C_j[\vec{a}_j \diamond \vec{b}_j]$. Hence w.l.o.g. we can assume that:
$$C'_i[\vec{a}'_i \diamond \vec{b}'_i] \equiv C'_j[\vec{a}'_j \diamond \vec{b}'_j] \qquad \text{and} \qquad C''_i[\vec{a}''_i \diamond \vec{b}''_i] \equiv C''_j[\vec{a}''_j \diamond \vec{b}''_j]$$
- Using the inductive property of well-nested couples (item (iv)) we know that the following couple of sets is well-nested:
$$\left(\left\{C'_i[\vec{a}'_i \diamond \vec{b}'_i] \mid i \in I^l \cup I^r \cup \{0\}\right\}, \left\{D'_j[\vec{c}'_j \diamond \vec{t}'_j] \mid j \in J^l \cup J^r\right\}_j\right)$$
Since if $(\mathcal{C}, \mathcal{D})$ is well-nested and $\mathcal{C}' \subseteq \mathcal{C} \wedge \mathcal{D}' \subseteq \mathcal{D}$ then $(\mathcal{C}', \mathcal{D}')$ is well-nested, we know that the following couple of sets is well-nested:
$$\left(\left\{C'_i[\vec{a}'_i \diamond \vec{b}'_i] \mid i \in I^l_{\mathsf{true}} \cup I^r_{\mathsf{true}} \cup \{0\}\right\}, \left\{D'_j[\vec{c}'_j \diamond \vec{t}'_j] \mid j \in J^l_{\mathsf{true}} \cup J^r_{\mathsf{true}}\right\}_j\right)$$

Since $\vec{a}'_0 \subset \vec{a}_0$ (resp. $\vec{a}''_0 \subset \vec{a}_0$), we can apply the induction hypothesis to s_{true} (resp s_{false}), which shows that for all $j \in J^l_{\mathsf{true}}$ (resp. $j \in J^r_{\mathsf{true}}$), there exists $t \in \vec{t}'_j$ such that $t \in \mathsf{leave\text{-}st}(s_{\mathsf{true}} \downarrow_R)$ (resp. $t \in \mathsf{leave\text{-}st}(s_{\mathsf{false}} \downarrow_R)$).

Let $S = I^l \cup I^r \cup \{0\} \cup J^l \cup J^r$. Let S_m be the subset of $I^l \cup I^r \cup \{0\}$ such that for all $i \in S_m$, $C_i[\vec{a}_i \diamond \vec{b}_i] \equiv C_m[\vec{a}_m \diamond \vec{b}_m]$ and $S' = S \backslash S_m$. We now want to apply Proposition 22 to show that $t \in \mathsf{leave\text{-}st}(s \downarrow_R)$. The only difficulty lies in showing that:

$$b_m \cap \left(\textstyle\bigcup_{i \in S'} \vec{a}'_i, \vec{a}''_i, \vec{b}'_i, \vec{b}''_i, \vec{c}'_i, \vec{c}''_i\right) = \emptyset$$

We know that $b_m \cap \left(\bigcup_{i \in S'} \vec{a}'_i, \vec{a}''_i, \vec{c}'_i, \vec{c}''_i\right) = \emptyset$ (since $\vec{a}'_i \subseteq \vec{a}_i \backslash \vec{b}_m, \dots$), so it only remains to show that:

$$\vec{b}_m \cap \bigcup_{i \in S'} \vec{b}'_i, \vec{b}''_i = \emptyset \tag{19}$$

Using hypothesis (iv) we know that for all $i \in S, \vec{b}_i \cap \vec{b}_m \neq \emptyset$ implies $i \in S_m$. Therefore since $\vec{b}'_i \subseteq \vec{b}_i$ (resp. $\vec{b}''_i \subseteq \vec{b}_i$), if $\vec{b}_m \cap \vec{b}'_i \neq \emptyset$ (resp. $\vec{b}_m \cap \vec{b}''_i \neq \emptyset$) then $i \in S_m$. Since $S' = S \backslash S_m$, we know that (19) holds.

l) Part 2: The proof of the general case is exactly the same than the one we did for the first inductive case of Part 1. ■

APPENDIX VI
IF-FREE CONDITIONALS

Given an if-free term s in R-normal form, s can be rewritten using R into a more complex term:

$$u \equiv C\left[\left(D_i\left[\vec{a}_i \diamond \vec{b}_i\right]\right)_i \diamond \vec{t}\right]$$

that is not if-free. Basically, the following proposition shows that as long as the term u does not contain true and false conditionals, the term s will always appear in the right-most and left-most branches of C. This is actually an invariant preserved by the term rewriting system R without the rules:

$$\mathsf{if\ true\ then}\ v\ \mathsf{else}\ w \rightarrow w \qquad\qquad \mathsf{if\ false\ then}\ v\ \mathsf{else}\ w \rightarrow w$$

Proposition 24. *For all if-free term s in R-normal form, if $s =_R C\left[\left(D_i\left[\vec{a}_i \diamond \vec{b}_i\right]\right)_i \diamond \vec{t}\right]$ where:*

- *$\vec{t} \cup \bigcup_i (\vec{a}_i \cup \vec{b}_i)$ are if-free and in R-normal form.*
- *Let i be such that $D_i\left[\vec{a}_i \diamond \vec{b}_i\right]$ is a term appearing on the left-most (resp. right-most) branch of C. Then $\mathsf{false} \notin \vec{a}_i \cup \vec{b}_i$ (resp. $\mathsf{true} \notin \vec{a}_i \cup \vec{b}_i$).*

Then the left-most (resp. right-most) element of $\vec{t}$ is s.

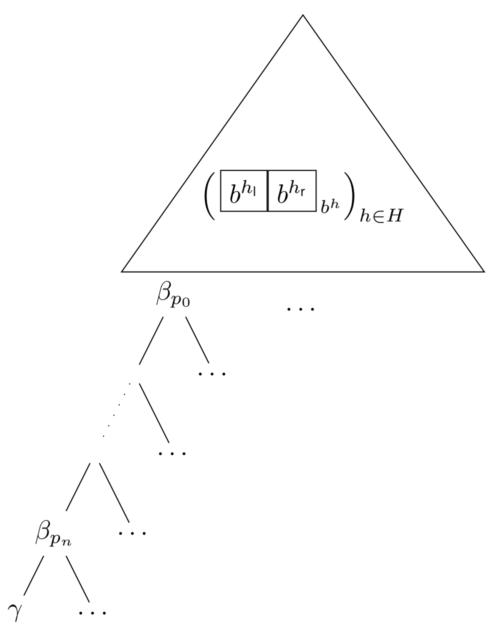
Proof. If suffices to show that the existence of a decomposition satisfying these two properties is preserved by $\rightarrow_R$, which is simple. We conclude by observing that since s is if-free, the only decomposition of $s\downarrow_R$ into $C\left[\left(D_i\left[\vec{a}_i \diamond \vec{b}_i\right]\right)_i \diamond \vec{t}\right]$ is such that $C \equiv []$. Hence $\vec{t}$ is a single element u, and $u \equiv s\downarrow_R \equiv s$. ■

We are now ready to prove Proposition 6, which we recall below.

Proposition. *Let b an if-free conditional in R-normal, with $b \not\equiv \mathsf{false}$ (resp. $b \not\equiv \mathsf{true}$). Then there exists no derivation of $b \sim \mathsf{false}$ (resp. $b \sim \mathsf{true}$) in $\mathcal{A}_{\succ}$.*

Proof. We prove only that there is no derivation of $b \sim \mathsf{false}$ in $\mathcal{A}_{\succ}$ (the proof that there is no derivation of $b \sim \mathsf{true}$ in $\mathcal{A}_{\succ}$ is exactly the same). We prove this by contradiction. Let b an if-free conditional in R-normal form, and let P be such that $P \vdash^{\mathsf{npf}} b \sim \mathsf{false}$. We choose b such that P is of minimal size.

First the minimality of the derivation implies that for all $\mathsf{h} \in \mathsf{index}(P)$, for all b_0 such that $b_0 \leq^{\mathsf{h}}_{\mathsf{cs}} (b, P)$ or $b_0 \leq^{\mathsf{h}}_{\mathsf{cs}} (\mathsf{false}, P)$, $b_0 \neq \mathsf{false}$. Let $H = \mathsf{cs\text{-}pos}(P)$. We now focus on the left-most branch of the proof:



Let $l \in \mathsf{label}(P)$. First we show that for all $\beta \leq^{\epsilon,l}_{\mathsf{c}} (b, P)$, $\beta \neq_R \mathsf{false}$. Assume that this is not the case, let $\beta =_R \mathsf{false}$ and β' be such that $(\beta, \beta') \leq^{\epsilon,l}_{\mathsf{c\sim c}} (b \sim \mathsf{false}, P)$. If $\beta =_R \beta' =_R \mathsf{false}$ then there is an easy proof cut elimination which yields a smaller proof P' of $b \sim \mathsf{false}$.
Hence assume $\beta' \neq_R \mathsf{false}$. If $\beta =_R \mathsf{false}$ then $\mathsf{leave\text{-}st}(\beta\downarrow_R) = \{\mathsf{false}\} = \mathsf{leave\text{-}st}(\mathsf{false}\downarrow_R)$. As β is a normalized basic conditional, using Proposition 16 we have $\beta \equiv \mathsf{false}$.
There exists a derivation of $\beta \sim \beta'$ in $\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2}$. Since $\beta \equiv \mathsf{false}$, no rules in $\mathsf{FA_s}$ are applied. Therefore the derivation is only an application of $\mathsf{CCA2}$, which is not possible.
Similarly we do not have $\beta \neq_R \mathsf{false}$ and $\beta' =_R \mathsf{false}$.
Using Proposition 16 we know that $\beta \neq_R \mathsf{false}$ implies that for all $u \in \mathsf{leave\text{-}st}(\beta\downarrow_R)$, $u \not\equiv \mathsf{false}$. Moreover by assumptions, for all $a \in \mathsf{cond\text{-}st}(\beta\downarrow_R)$, $a \not\equiv \mathsf{false}$.
We let $(\gamma, \gamma') \leq^{\epsilon,l}_{\mathsf{l}} (b \sim \mathsf{false}, P)$ be the left-most elements (as shown in the Figure). For all $a \in \mathsf{cond\text{-}st}(\gamma\downarrow_R)$, $a \not\equiv \mathsf{false}$. Hence if we let $u_0 \in \mathsf{leave\text{-}st}(\gamma\downarrow_R)$ be the left-most leave element of $\gamma\downarrow_R$, then by Proposition 24 we know that $u_0 \equiv b$.
Similarly, by applying the exact same reasoning to the other side, we know that the left-most leaf element u'_0 of $\gamma'\downarrow_R$ is false, and by Proposition 16 we get that $\gamma' \equiv \mathsf{false}$. Since there exists a derivation of $\gamma \sim \gamma'$ in $\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2}$, no rule in $\mathsf{FA_s}$ is applied. Therefore the derivation is only an application of $\mathsf{CCA2}$. Contradiction. ■

We can then ensure that any proof P of $t \sim t'$ is not containing a $\mathsf{CS}_\Box$ or $\overline{\mathsf{BFA}}$ application on true or false: if we have a $\mathsf{CS}_\Box$ or $\overline{\mathsf{BFA}}$ application on $(\mathsf{true}, \mathsf{true})$ or $(\mathsf{false}, \mathsf{false})$ then there is a proof cut elimination without it yielding a smaller proof, and the previous proposition ensures that there are no $\mathsf{CS}_\Box$ or $\overline{\mathsf{BFA}}$ application on $(\mathsf{true}, b), (b, \mathsf{true}), (\mathsf{false}, b)$ or (b, false) (with $b \neq_R \mathsf{false}, \mathsf{true}$).

Proposition 25. *For all $P \vdash^{\mathsf{npf}} t \sim t'$, there exists P' such that $P' \vdash^{\mathsf{npf}} t \sim t'$ and for all $l \in \mathit{label}(P'), h \in \mathit{index}(P'), \mathsf{x} \in \{\mathit{l}, \mathit{r}\}$ we have:*

$$\forall \beta \in \left((\leq_{\mathsf{c}}^{h_{\mathsf{x}}, l} \cup \leq_{\mathsf{cs}}^{h_{\mathsf{x}}})(t, P') \right) \cup \left((\leq_{\mathsf{c}}^{h_{\mathsf{x}}, l} \cup \leq_{\mathsf{cs}}^{h_{\mathsf{x}}})(t', P') \right), \quad \{\mathit{false}, \mathit{true}\} \cap \mathsf{leave\text{-}st}(\beta \downarrow_R) = \emptyset$$

Proof. We can construct a proof P' from P through simple proof cut eliminations such that:

$$\{(\mathsf{true}, \mathsf{true}), (\mathsf{false}, \mathsf{false})\} \cap (\leq_{\mathsf{c} \sim \mathsf{c}}^{h_{\mathsf{x}}, l} (t \sim t', P) \cup \leq_{\mathsf{cs} \sim \mathsf{cs}}^{h_{\mathsf{x}}} (t \sim t', P)) = \emptyset$$

Then using Proposition 6 we know that for all:

$$(\beta, \beta') \in (\leq_{\mathsf{c} \sim \mathsf{c}}^{h_{\mathsf{x}}, l} (t \sim t', P) \cup \leq_{\mathsf{cs} \sim \mathsf{cs}}^{h_{\mathsf{x}}} (t \sim t', P))$$

the conditionals β and β' are such that $\beta \neq_R \mathsf{false}$ and $\beta' \neq_R \mathsf{false}$ (same with true). Finally if $\beta \neq_R \mathsf{false}$ then one can easily check that for all $u \in \mathsf{leave\text{-}st}(\beta \downarrow_R)$, $u \not\equiv \mathsf{false}$ (idem with true). ■

We recall that showed in Lemma 2 that if $\vdash_{\mathcal{A}_{\mathsf{FA_s}}} b, b \sim b', b''$ then $b' \equiv b''$. We are now ready to give the proof of Lemma 9, which generalize this to the case where $\vdash^{\mathsf{npf}} b, b \sim b', b''$, but only when b, b', b'' are if-free.

Lemma (9)**.** *For all a, a', b, c such that their R-normal forms are if-free and such that $a =_R a'$, if $P \vdash^{\mathsf{npf}} a, a' \sim b, c$ then $b =_R c$.*

Proof. Let $t \equiv \langle a\,,\,a \rangle$ and $t' \equiv \langle b\,,\,c \rangle$, we know that there exists P' such that $P' \vdash^{\mathsf{npf}} t \sim t'$ since $P \vdash^{\mathsf{npf}} a, a' \sim b, c$. Moreover using Proposition 25 we know that for all $h \in \mathsf{index}(P)$, for all l, x:

$$\forall \beta \in \left((\leq_{\mathsf{c}}^{h_{\mathsf{x}}, l} \cup \leq_{\mathsf{cs}}^{h_{\mathsf{x}}, l})(t, P') \right) \cup \left((\leq_{\mathsf{c}}^{h_{\mathsf{x}}, l} \cup \leq_{\mathsf{cs}}^{h_{\mathsf{x}}, l})(t', P') \right), \quad \{\mathsf{false}, \mathsf{true}\} \cap \mathsf{leave\text{-}st}(\beta \downarrow_R) = \emptyset$$

Let $(\gamma, \gamma') \leq_{\mathsf{l}}^{\epsilon, l} (t \sim t', P)$ be the left-most elements of t and t'. By Proposition 24 we know that $\langle a\,,\,a \rangle \downarrow_R \in \mathsf{leave\text{-}st}(\gamma \downarrow_R)$ and $\langle b\,,\,c \rangle \downarrow_R \in \mathsf{leave\text{-}st}(\gamma' \downarrow_R)$. More precisely we know that $\langle b\,,\,c \rangle$ is the left-most element of $\gamma' \downarrow_R$.

Since $\gamma \sim \gamma'$ is provable in $\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2}$, we know that there exists $\gamma_1, \gamma_2, \gamma'_1, \gamma'_2$ such that they are $\mathcal{S}_l^P$-normalized basic terms and $\gamma =_R \langle \gamma_1\,,\,\gamma_2 \rangle$, $\gamma' =_R \langle \gamma'_1\,,\,\gamma'_2 \rangle$, and the formula $\gamma_1, \gamma_2 \sim \gamma'_1, \gamma'_2$ is provable in $\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2}$.

Moreover $a \in \mathsf{leave\text{-}st}(\gamma_1 \downarrow_R)$ and $a \in \mathsf{leave\text{-}st}(\gamma_2 \downarrow_R)$, hence $\mathsf{leave\text{-}st}(\gamma_1 \downarrow_R) \cap \mathsf{leave\text{-}st}(\gamma_2 \downarrow_R) \neq \emptyset$. Using Proposition 16 we deduce that $\gamma_1 \equiv \gamma_2$.

Therefore there exists a proof of $\gamma_1, \gamma_1 \sim \gamma'_1, \gamma'_2$ in $\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA2}$, and by Lemma 2 we get that $\gamma'_1 \equiv \gamma'_2$.

We conclude by observing that since $\langle b\,,\,c \rangle$ is the let-most element of $\gamma' \downarrow_R$, b (resp. c) is the left-most element of γ'_1 (resp. γ'_2). Therefore $b \equiv c$. ■

Definition 42. *For all term t, we let $<_{\mathit{bc}}^{\mathcal{S}} t$ be the set of $\mathcal{S}$-normalized basic conditional appearing in t, defined inductively by:*

- *If t is a $\mathcal{S}$-normalized simple term $C[\vec{b} \diamond \vec{u}]$, then:*

$$<_{\mathit{bc}}^{\mathcal{S}} t \;=\; \vec{b} \;\cup\; \left(<_{\mathit{bc}}^{\mathcal{S}} \vec{b}\right) \;\cup\; \left(<_{\mathit{bc}}^{\mathcal{S}} \vec{u}\right)$$

- *If t is a $\mathcal{S}$-normalized basic term $B[\vec{w}, (\alpha_i)_i, (\mathit{dec}_j)_j]$, then:*

$$<_{\mathit{bc}}^{\mathcal{S}} t \;=\; \bigcup_i <_{\mathit{bc}}^{\mathcal{S}} \alpha_i \;\cup\; \bigcup_j <_{\mathit{bc}}^{\mathcal{S}} \mathit{dec}_j$$

- *For all $\mathcal{S}$-encryption oracle call $t \equiv \{u\}_{\mathit{pk}}^r$, then:*

$$<_{\mathit{bc}}^{\mathcal{S}} t \;=\; <_{\mathit{bc}}^{\mathcal{S}} u$$

- *For all $\mathcal{S}$-decryption oracle call $C[\vec{b} \diamond \vec{u}]$, let s, sk such that terms in $\vec{u}$ are of the form $\mathbf{0}(\mathit{dec}(s[(\alpha_i), (\mathit{dec}_j)_j], \mathit{sk}))$ or $\mathit{dec}(s[(\alpha_i), (\mathit{dec}_j)_j], \mathit{sk})$, and u is if-free. Then:*

$$<_{\mathit{bc}}^{\mathcal{S}} t \;=\; \vec{b} \;\cup\; \left(<_{\mathit{bc}}^{\mathcal{S}} \vec{b}\right) \;\cup\; \bigcup_i <_{\mathit{bc}}^{\mathcal{S}} \alpha_i \;\cup\; \bigcup_j <_{\mathit{bc}}^{\mathcal{S}} \mathit{dec}_j$$

Proposition 26. *For all term β such that β is a $\mathcal{S}$-normalized basic term, $\mathcal{S}$-normalized simple term, $\mathcal{S}$-decryption oracle call or $\mathcal{S}$-encryption oracle call we have:*

$$\overline{\mathsf{cond\text{-}st}}(\beta) = \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \beta} \overline{\mathsf{leave\text{-}st}}(u)$$

Proof. We prove this by induction on the order $<^{\mathcal{S}}_{\mathsf{ind}}$.

a) Base Case: If β is minimal for $<^{\mathcal{S}}_{\mathsf{ind}}$, then we have the following cases:

- $\mathcal{S}$-decryption oracle call: β is of the form $C[\vec{b} \diamond \vec{u}]$, and there exists s, sk such that terms in $\vec{u}$ are of the form $\mathbf{0}(\mathsf{dec}(s, \mathsf{sk}))$ or $\mathsf{dec}(s, \mathsf{sk})$, and u is if-free. Moreover by minimality of β the vector of terms $\vec{b}$ must be empty, since for all $b \in \vec{b}$, b is a $\mathcal{S}$-normalized basic term.
Hence $\overline{\mathsf{cond\text{-}st}}(\beta) = \emptyset$. Finally since β is minimal there are no u such that $u <^{\mathcal{S}}_{\mathsf{bc}} \beta$.
- $\mathcal{S}$-encryption oracle call case cannot happen, as β would not be minimal.
- $\mathcal{S}$-normalized basic term: β contains no $\mathsf{if\ then\ else}$ symbol, hence $\overline{\mathsf{cond\text{-}st}}(\beta) = \emptyset$. Moreover since β is minimal there are no u such that $u <^{\mathcal{S}}_{\mathsf{bc}} \beta$.
- $\mathcal{S}$-normalized simple term case cannot happen, as β would not be minimal.

b) Inductive Case: Let β be such that for all $\beta' \neq \beta$, if $\beta' <^{\mathcal{S}}_{\mathsf{ind}} \beta$ then the property holds for β'.

- $\mathcal{S}$-normalized basic term: β is of the form $B[\vec{w}, (\alpha_i)_i, (\mathsf{dec}_j)_j]$. The result is then immediate by induction hypothesis and using the definition of $\overline{\mathsf{cond\text{-}st}}(\cdot)$ and $<^{\mathcal{S}}_{\mathsf{bc}}$:

$$\begin{aligned}
\overline{\mathsf{cond\text{-}st}}(\beta) &= \bigcup_i \overline{\mathsf{cond\text{-}st}}(\alpha_i) \cup \bigcup_j \overline{\mathsf{cond\text{-}st}}(\mathsf{dec}_i) && \text{(By definition of } \overline{\mathsf{cond\text{-}st}}(\cdot)\text{)}\\
&= \bigcup_i \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \alpha_i} \overline{\mathsf{leave\text{-}st}}(u) \cup \bigcup_j \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \mathsf{dec}_j} \overline{\mathsf{leave\text{-}st}}(u) && \text{(By induction hypothesis)}\\
&= \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \beta} \overline{\mathsf{leave\text{-}st}}(u) && \text{(By definition of } <^{\mathcal{S}}_{\mathsf{bc}}\text{)}
\end{aligned}$$

- $\mathcal{S}$-decryption oracle call: t is of the form $C[\vec{b} \diamond \vec{u}]$, where there exists s, sk such that terms in $\vec{u}$ are of the form $\mathbf{0}(\mathsf{dec}(s[(\alpha_i), (\mathsf{dec}_j)_j], \mathsf{sk}))$ or $\mathsf{dec}(s[(\alpha_i), (\mathsf{dec}_j)_j], \mathsf{sk})$, and u is if-free. Then:

$$\begin{aligned}
\overline{\mathsf{cond\text{-}st}}(\beta) &= \bigcup_i \overline{\mathsf{cond\text{-}st}}(\alpha_i) \cup \bigcup_j \overline{\mathsf{cond\text{-}st}}(\mathsf{dec}_i) \cup \overline{\mathsf{cond\text{-}st}}(\vec{g}) \cup \overline{\mathsf{leave\text{-}st}}(\vec{g})\\
& \text{(By definition of } \overline{\mathsf{cond\text{-}st}}(\cdot)\text{)}\\
&= \bigcup_i \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \alpha_i} \overline{\mathsf{leave\text{-}st}}(u) \cup \bigcup_j \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \mathsf{dec}_j} \overline{\mathsf{leave\text{-}st}}(u) \cup \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \vec{g}} \overline{\mathsf{leave\text{-}st}}(u) \cup \overline{\mathsf{leave\text{-}st}}(\vec{g})\\
& \text{(By induction hypothesis: remark that guards in } \vec{g} \text{ are } \mathcal{S}\text{-normalized basic terms s.t. } \vec{g} \leq^{\mathcal{S}}_{\mathsf{bt}} \beta\text{)}\\
&= \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \beta} \overline{\mathsf{leave\text{-}st}}(u) && \text{(By definition of } <^{\mathcal{S}}_{\mathsf{bc}}\text{)}
\end{aligned}$$

- $\mathcal{S}$-encryption oracle call: t is of the form $\{s\}^r_{\mathsf{pk}}$, then:

$$\begin{aligned}
\overline{\mathsf{cond\text{-}st}}(\beta) &= \overline{\mathsf{cond\text{-}st}}(s) && \text{(By definition of } \overline{\mathsf{cond\text{-}st}}(\cdot)\text{)}\\
&= \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} s} \overline{\mathsf{leave\text{-}st}}(u) && \text{(By induction hypothesis)}\\
&= \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \beta} \overline{\mathsf{leave\text{-}st}}(u) && \text{(By definition of } <^{\mathcal{S}}_{\mathsf{bc}}\text{)}
\end{aligned}$$

- $\mathcal{S}$-normalized simple term: t is of the form $C[\vec{b} \diamond \vec{v}]$. Then:

$$\begin{aligned}
\overline{\mathsf{cond\text{-}st}}(\beta) &= \overline{\mathsf{cond\text{-}st}}(\vec{b}) \cup \overline{\mathsf{cond\text{-}st}}(\vec{v}) \cup \overline{\mathsf{leave\text{-}st}}(\vec{b}) && \text{(By definition of } \overline{\mathsf{cond\text{-}st}}(\cdot)\text{)}\\
&= \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \vec{b}} \overline{\mathsf{leave\text{-}st}}(u) \cup \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \vec{v}} \overline{\mathsf{leave\text{-}st}}(u) \cup \overline{\mathsf{leave\text{-}st}}(\vec{b}) && \text{(By induction hypothesis)}\\
&= \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \beta} \overline{\mathsf{leave\text{-}st}}(u) && \text{(By definition of } <^{\mathcal{S}}_{\mathsf{bc}}\text{)}
\end{aligned}$$

$\blacksquare$

Proposition 27. *Let $P \vdash^{\mathsf{npf}} t \sim t'$. Then for all h, l for all $\beta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t,P)$, $\overline{\mathsf{cond\text{-}st}}(\beta) \cap \overline{\mathsf{leave\text{-}st}}(\beta) = \emptyset$.*

Proof. Let h, l and $\beta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t,P)$ be such that $\overline{\mathsf{cond\text{-}st}}(\beta) \cap \overline{\mathsf{leave\text{-}st}}(\beta) \neq \emptyset$. By Proposition 26 this means that there exists a $\mathcal{S}_l$-normalized basic term $u <_{\mathsf{bc}}^{\mathcal{S}_l} \beta$ such that $\overline{\mathsf{leave\text{-}st}}(u) \cap \overline{\mathsf{leave\text{-}st}}(\beta) \neq \emptyset$.

Using Proposition 16 we know that $u \equiv \beta$. But $u <_{\mathsf{bc}}^{\mathcal{S}_l} \beta$ implies that u is a strict subterm of β. Absurd. $\blacksquare$

Definition 43. *Let $P \vdash^{\mathsf{npf}} t \sim t'$, we know that t is of the form:*

$$t \equiv C\left[\left(\boxed{b^{h_l}}\,\boxed{b^{h_r}}_{\,b^h}\right)_{h \in H} \diamond \left(D_l\left[(\beta)_{\beta \leq_{\mathsf{c}}^{\epsilon,l}(t,P)} \diamond (\gamma)_{\gamma \leq_{\mathsf{l}}^{\epsilon,l}(t,P)}\right]\right)_{l \in L}\right]$$

For all l, we let:

- *${}^{\delta}\mathsf{cs\text{-}path}^{\epsilon,l}(t,P)$ be the directed path of conditional occurring from the root of t to $D_l[]$ in P.*
- *${}^{\delta}\mathsf{cs\text{-}path}_{\sim}{}^{\epsilon,l}(t \sim t',P)$ be the directed path of pairs of conditionals occurring from the root of (t,t') to $D_l[]$ in P.*

We extend this to all $h \in \mathsf{index}(P), \mathsf{x} \in \{\mathsf{l},\mathsf{r}\}$ by having:

$$\begin{aligned} & {}^{\delta}\mathsf{cs\text{-}path}^{h_{\mathsf{x}},l}(t,P) = {}^{\delta}\mathsf{cs\text{-}path}^{\epsilon,l}(b, \mathsf{extract}_{\mathsf{x}}(h,P)) \\ \text{and}\quad & {}^{\delta}\mathsf{cs\text{-}path}_{\sim}{}^{h_{\mathsf{x}},l}(t \sim t',P) = {}^{\delta}\mathsf{cs\text{-}path}_{\sim}{}^{\epsilon,l}(b \sim b', \mathsf{extract}_{\mathsf{x}}(h,P)) \end{aligned}$$

where $\mathsf{extract}_{\mathsf{x}}(h,P)$ is a proof of $b \sim b'$.

Lemma 18. *Let $P \vdash^{\mathsf{npf}} t \sim t'$. There exists P' such that $P' \vdash^{\mathsf{npf}} t \sim t'$ and for all $h \in \mathsf{index}(P')$ with $h \neq \epsilon$, for all $\mathsf{x} \in \{\mathsf{l},\mathsf{r}\}$, if we let $\mathsf{h} = h_{\mathsf{x}}$ and $P^{\mathsf{h}} = \mathsf{extract}_{\mathsf{x}}(h,P')$ be the proof of $b^{\mathsf{h}} \sim b'^{\mathsf{h}}$ then for all $l \in \mathsf{label}(P^{\mathsf{h}})$:*

(a) *The proof P^{h} does not use the $\{\overline{\mathsf{BFA}}(b,b')\}$ rules.*
(b) *$\mathsf{cs\text{-}path}^{\mathsf{h},l}(t,P)$ (resp. $\mathsf{cs\text{-}path}^{\mathsf{h},l}(t',P)$) does not contain two occurrences of the same conditional.*
(c) *For all $\gamma \leq_{\mathsf{l}}^{\mathsf{h},l} (t,P')$, $(b^{\mathsf{h}} \downarrow_R) \in \mathsf{leave\text{-}st}(\gamma \downarrow_R)$ and for all $\gamma' \leq_{\mathsf{l}}^{\mathsf{h},l} (t',P')$, $(b'^{\mathsf{h}} \downarrow_R) \in \mathsf{leave\text{-}st}(\gamma' \downarrow_R)$.*
(d) *For all $\beta \leq_{\mathsf{c}}^{\epsilon,l} (t,P')$, $\mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \mathsf{cs\text{-}path}^{\epsilon,l}(t,P) = \emptyset$ (same for t').*
(e) *For all $\gamma \leq_{\mathsf{l}}^{\epsilon,l} (t,P')$, $\mathsf{leave\text{-}st}(t \downarrow_R) \cap \mathsf{leave\text{-}st}(\gamma \downarrow_R) \neq \emptyset$ (same for t').*

Proof. Using Proposition 25, we know that we have P such that $P \vdash^{\mathsf{npf}} t \sim t'$ and for all $l \in \mathsf{label}(P), h \in \mathsf{index}(P), \mathsf{x} \in \{\mathsf{l},\mathsf{r}\}$ we have:

$$\forall \beta \in \left((\leq_{\mathsf{c}}^{h_{\mathsf{x}},l} \cup \leq_{\mathsf{cs}}^{h_{\mathsf{x}},l})(t,P)\right) \cup \left((\leq_{\mathsf{c}}^{h_{\mathsf{x}},l} \cup \leq_{\mathsf{cs}}^{h_{\mathsf{x}},l})(t',P)\right), \quad \{\mathsf{false},\mathsf{true}\} \cap \mathsf{leave\text{-}st}(\beta \downarrow_R) = \emptyset \tag{20}$$

First we start by rewriting the proof P so that all CS application are of the form:

$$\frac{b,(u_i)_i \sim b',(u'_i)_i \quad b,(v_i)_i \sim b',(v'_i)_i}{(\mathsf{if}\ b\ \mathsf{then}\ u_i\ \mathsf{else}\ v_i)_i \sim (\mathsf{if}\ b'\ \mathsf{then}\ u'_i\ \mathsf{else}\ v'_i)_i}\ \mathsf{CS} \tag{21}$$

We prove by induction on n, starting with the inner-most CS conditionals, that there exists P such that $P \vdash^{\mathsf{npf}} t \sim t'$, (20) is true for P and the following properties hold for all $h, h' \in \mathsf{index}(P)$:

(i) If $\mathsf{if\text{-}depth}_P(h) \geq n$ then the $\mathsf{extract}_{\mathsf{l}}(h,P)$ and $\mathsf{extract}_{\mathsf{r}}(h,P)$ do not use the $\{\overline{\mathsf{BFA}}(b,b')\}$ rules.
(ii) If $\mathsf{if\text{-}depth}_P(h) \geq n$ then for all x, l, $\mathsf{cs\text{-}path}^{h_{\mathsf{x}},l}(t,P)$ and $\mathsf{cs\text{-}path}^{h_{\mathsf{x}},l}(t',P)$ do not contain two occurrences of the same conditional.
(iii) If $\mathsf{if\text{-}depth}_P(h) \geq n$ then for all x, if $\mathsf{extract}_{\mathsf{x}}(h,P)$ is the proof of $b \sim b'$ then for all l, for all $\gamma \leq_{\mathsf{l}}^{h_{\mathsf{x}},l} (t,P)$, $(b \downarrow_R) \in \mathsf{leave\text{-}st}(\gamma \downarrow_R)$ and for all $\gamma' \leq_{\mathsf{l}}^{h_{\mathsf{x}},l} (t',P)$, $(b' \downarrow_R) \in \mathsf{leave\text{-}st}(\gamma' \downarrow_R)$.
(iv) If $\mathsf{if\text{-}depth}_P(h) < n$ then for all $h, h' \in \mathsf{index}(P)$ such that $h \leq h'$, if we let h'' be such that $h' = h \cdot h''$ and x be such that $h'' \in \mathsf{index}(\mathsf{extract}_{\mathsf{x}}(h,P))$, then for all x', for all $l \in \mathsf{label}(\mathsf{extract}_{\mathsf{x}'}(h',P))$, we have

$$^{\delta}\mathsf{cs\text{-}path}^{h_{\mathsf{x}},l}(t,P) \supseteq {}^{\delta}\mathsf{cs\text{-}path}^{h'_{\mathsf{x}'},l}(t,P)$$

Let $n_{\max}$ be the maximal $\mathsf{if\text{-}depth}$ in the proof of $t \sim t'$:

$$n_{\max} = \max_{h \in \mathsf{index}(P)} \mathsf{if\text{-}depth}_P(h)$$

c) Base Case:: We are going to show that the invariants hold at $n_{\max}+1$. Invariants (i), (ii) and (iii) are obvious, since there exists no h such that $\mathsf{if\text{-}depth}_P(h) \geq n_{\max}+1$; and invariant (iv) is a consequence of the rewriting done in (21).

d) Inductive Case:: Assume that the property holds for $n+1$ and let us show that it holds for n.

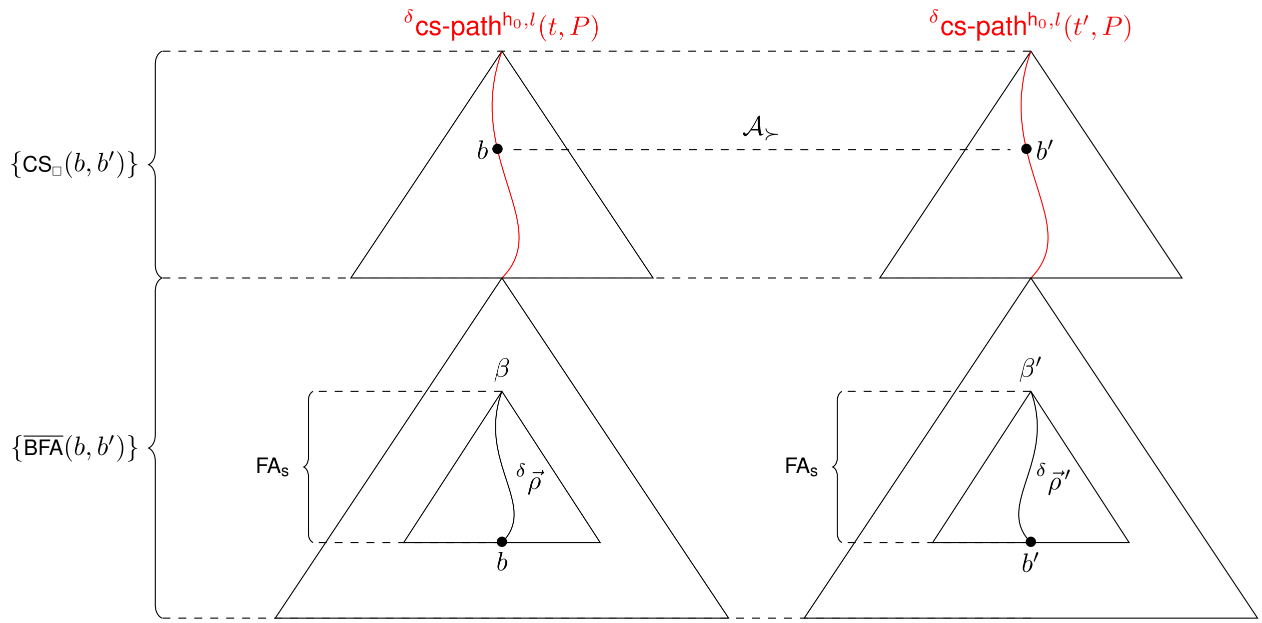


Fig. 12. Corresponding occurrences of b and b' in the proof of Lemma 18

e) Step 1: Let $l \in \mathsf{label}(P)$ and $h_0 \in \mathsf{h\text{-}branch}(l)$ such that $\mathsf{if\text{-}depth}_P(h_0) = n$. Let $x_0 \in \{\mathsf{l}, \mathsf{r}\}$ and $\mathsf{h}_0 = h_{0x_0}$. We start by showing that for all l, for all $\beta \leq_{\mathsf{c}}^{\mathsf{h}_0,l} (t,P)$, if there exists $b \in \mathsf{cs\text{-}path}^{\mathsf{h}_0,l}(t,P)$ such that $b \in \mathsf{leave\text{-}st}(\beta\downarrow_R)$ then there exists $(b,b') \in \mathsf{cs\text{-}path}_{\sim}^{\mathsf{h}_0,l}(t,P)$ and β' such that $(\beta,\beta') \leq_{\mathsf{c}\sim\mathsf{c}}^{\mathsf{h}_0,l} (t \sim t', P)$ and:

- $b' \in \mathsf{leave\text{-}st}(\beta'\downarrow_R)$.
- The directed path ${}^{\delta}\vec{\rho}$ (resp. ${}^{\delta}\vec{\rho}'$) of the conditionals occurring from the root of $\beta\downarrow_R$ (resp. $\beta'\downarrow_R$) to the leave b (resp. b') is such that ${}^{\delta}\vec{\rho} \subseteq {}^{\delta}\mathsf{cs\text{-}path}^{\mathsf{h}_0,l}(t,P)$ (resp. ${}^{\delta}\vec{\rho}' \subseteq {}^{\delta}\mathsf{cs\text{-}path}^{\mathsf{h}_0,l}(t,P)$).

This is described in Fig. 12.

Let $\beta \leq_{\mathsf{c}}^{\mathsf{h}_0,l} (t,P)$ and $b \in \mathsf{cs\text{-}path}^{\mathsf{h}_0,l}(t,P)$ such that $b \in \mathsf{leave\text{-}st}(\beta\downarrow_R)$. We know that there exists b' and β' such that $(b,b') \in \mathsf{cs\text{-}path}_{\sim}^{\mathsf{h}_0,l}(t,P)$ and $(\beta,\beta') \leq_{\mathsf{c}\sim\mathsf{c}}^{\mathsf{h}_0,l} (t \sim t', P)$.

Let $h \in \mathsf{cs\text{-}pos}(\mathsf{extract}_{x_0}(h_0,P))$ and x be the direction taken in l at h be such that $\mathsf{extract}(h,P)$ is the rule $\mathsf{CS}_{\square}(b,b')$. We know that $\mathsf{extract}_{\mathsf{x}}(h,P)$ is a proof of $a \sim a'$, where $a =_R b$ and $a' =_R b'$. As $\mathsf{if\text{-}depth}(h) = n+1$ we know by induction hypothesis (i) that $\mathsf{extract}_{\mathsf{x}}(h,P)$ does not uses $\{\overline{\mathsf{BFA}}(b,b')\}$. Hence the set $\leq_{\mathsf{l}}^{\epsilon,l} (a, \mathsf{extract}_{\mathsf{x}}(h,P))$ is the singleton $\{\gamma_l\}$ and the set $\leq_{\mathsf{l}}^{\epsilon,l} (a', \mathsf{extract}_{\mathsf{x}}(h,P))$ is the singleton $\{\gamma_l'\}$. Let $H = \mathsf{index}(\mathsf{extract}_{\mathsf{x}}(h,P))$, we have:

$$a \equiv C\left[(b^g)_{g\in H} \diamond (\gamma_{l_a})_{l_a}\right] \qquad a' \equiv C\left[(b'^g)_{g\in H} \diamond \left(\gamma'_{l_a}\right)_{l_a}\right]$$

By induction hypothesis (iii) we know that $b \in \mathsf{leave\text{-}st}(\gamma_l\downarrow_R)$ and $b' \in \mathsf{leave\text{-}st}(\gamma_l'\downarrow_R)$. γ_l and β are $\mathcal{S}_l$-normalized basic terms, hence using Proposition 16 we know that $\beta \equiv \gamma_l$. We can extract from the branch l of P a proof of $\gamma_l, \beta \sim \gamma_l', \beta'$ in $\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \overline{\mathsf{CCA2}}$. Therefore, using Lemma 2, we get that $\beta' \equiv \gamma_l'$. Since $b' \in \mathsf{leave\text{-}st}(\gamma_l'\downarrow_R)$, we deduce that $b' \in \mathsf{leave\text{-}st}(\beta'\downarrow_R)$. This concludes the proof of the first bullet point.

By induction hypothesis (iv) we know that

$${}^{\delta}\mathsf{cs\text{-}path}^{(h_0)_{\mathsf{x}_0},l}(t,P) \supseteq {}^{\delta}\mathsf{cs\text{-}path}^{h_{\mathsf{x}},l}(t,P) \quad \wedge \quad {}^{\delta}\mathsf{cs\text{-}path}^{(h_0)_{\mathsf{x}_0},l}(t',P) \supseteq {}^{\delta}\mathsf{cs\text{-}path}^{h_{\mathsf{x}},l}(t',P)$$

By definition of $\vec{\rho}$, $\mathsf{cond\text{-}st}(\gamma_l\downarrow_R) \supseteq \vec{\rho}$. More precisely, using the facts that $a \equiv C\left[(b^g)_{g\in H} \diamond (\gamma_l)_l\right]$ and since $\mathsf{cond\text{-}st}(a\downarrow_R) = \{b\}$, and invariant (ii), we can show that ${}^{\delta}\vec{\rho} \subseteq {}^{\delta}\mathsf{cs\text{-}path}^{h_{\mathsf{x}},l}(t,P)$. By consequence, ${}^{\delta}\vec{\rho} \subseteq {}^{\delta}\mathsf{cs\text{-}path}^{(h_0)_{\mathsf{x}_0},l}(t,P)$. Similarly we show that ${}^{\delta}\vec{\rho}' \subseteq {}^{\delta}\mathsf{cs\text{-}path}^{(h_0)_{\mathsf{x}_0},l}(t',P)$.

f) Step 2: By doing some proof cut elimination, we can guarantee that for all l, for all $\beta \leq_{\mathsf{c}}^{\mathsf{h}_0,l} (t,P)$:

$$\mathsf{leave\text{-}st}(\beta\downarrow_R) \cap \mathsf{cs\text{-}path}^{\mathsf{h}_0,l}(t,P) = \emptyset$$

Assume this is not the case: using **Step 1** we have:

$${}^{\delta}\vec{\rho} \subseteq {}^{\delta}\mathsf{cs\text{-}path}^{(h_0)_{\mathsf{x}_0},l}(t,P) \quad \wedge \quad {}^{\delta}\vec{\rho}' \subseteq {}^{\delta}\mathsf{cs\text{-}path}^{(h_0)_{\mathsf{x}_0},l}(t',P)$$

Therefore we can rewrite β and β' into, respectively, b and b' (this is possible because we have an inclusion between the *directed paths*, not just the paths). We can then rewrite b and b' into true if we are on the then branch of b and b' (i.e. $\mathsf{x} = \mathsf{l}$), and false if we are on the else branch (i.e. $\mathsf{x} = \mathsf{r}$). Finally we get rid of true and false using R, and check that the resulting proof verifies (20) and the induction invariants.

g) Step 2 b.: Then we show that we can assume that (ii) holds through some proof rewriting, while maintaining invariant (iv).

Let $(a, a'), (b, b') \leq^{\mathsf{h}_0}_{\mathsf{cs}\sim\mathsf{cs}} (t, P)$ such that $a \equiv b$ and they are on the same branch l. Since they are on the same branch, we can extract a proof $Q \vdash^{\mathsf{npf}} a, a \sim a', b'$. Moreover $a \downarrow_R, a' \downarrow_R, b' \downarrow_R$ are if-free, therefore by Lemma 9 we have $a' \equiv b'$. We then do our standard proof cut elimination to get rid of the duplicate. We need to make sure that this ensure that invariant (iv) at rank n holds for Q: this follows from the fact that invariant (iv) holds for P at rank $n+1$ and that the cut takes place at depth n.

h) Step 3: We then show that (iii) holds. Let $b^{\mathsf{h}_0}, b'^{\mathsf{h}_0}$ be such that $\mathsf{extract}_{\mathsf{x}_0}(h, P) \vdash^{\mathsf{npf}} b^{\mathsf{h}_0} \sim b'^{\mathsf{h}_0}$. We know that:

$$b^{\mathsf{h}_0} \equiv C\left[\left(\left(\boxed{b^{h_\mathsf{l}}}\boxed{b^{h_\mathsf{r}}}_{b^h})_x\right)_{h \in H^{\mathsf{h}_0}} \diamond \left(D^{\mathsf{h}_0}_l\left[(\beta)_{\beta \leq^{\mathsf{h}_0,l}_{\mathsf{c}}(t,P)} \diamond (\gamma)_{\gamma \leq^{\mathsf{h}_0,l}_{\mathsf{l}}(t,P)}\right]\right)_{l \in L^{\mathsf{h}_0}}\right]$$

where $H^{\mathsf{h}_0} = \mathsf{cs\text{-}pos}(\mathsf{extract}_{\mathsf{x}_0}(h_0, P))$ and $L^{\mathsf{h}_0} = \mathsf{label}(\mathsf{extract}_{\mathsf{x}_0}(h_0, P))$.

To prove that for all l, for all $\gamma \leq^{\mathsf{h}_0,l}_{\mathsf{l}} (t, P)$, we have $b^{\mathsf{h}_0} \downarrow_R \in \mathsf{leave\text{-}st}(\gamma \downarrow_R)$, we only need to show that the hypotheses of Proposition 23 hold for b^{h_0} (then we do the same thing with b'^{h_0} to show that for all $\gamma' \leq^{\mathsf{h}_0,l}_{\mathsf{l}} (t', P)$ we have $b'^{\mathsf{h}_0} \downarrow_R \in \mathsf{leave\text{-}st}(\gamma' \downarrow_R)$):

- (23.i): the only difficulty lies in proving that for all $\beta \leq^{\mathsf{h}_0,l}_{\mathsf{c}} (t, P)$, $\mathsf{cond\text{-}st}(\beta \downarrow_R) \cap \mathsf{leave\text{-}st}(\beta \downarrow_R) = \emptyset$, which is shown in Proposition 27.
- (23.ii): this is a consequence of the fact that (20) holds for P.
- (23.iii): for pairs in $(\mathsf{cs\text{-}path}^{\mathsf{h}_0,l}(t, P))^2$ this was shown in **Step 2 b**. For couples of positions in $D^{\mathsf{h}_0}_l \times D^{\mathsf{h}_0}_l$ we have a proof cut elimination: let $p < p'$ be the positions in b^{h_0} of $\beta_0, \beta_1 \leq^{\mathsf{h}_0,l}_{\mathsf{c}} (t, P)$ on the same branch such that $\mathsf{leave\text{-}st}(\beta_0) \cap \mathsf{leave\text{-}st}(\beta_1) \neq \emptyset$. By Proposition 16 we know that $\beta_0 \equiv \beta_1$. Let β'_0, β'_1 be the conditionals at positions, respectively, p and p' in b'^{h_0}. We know that $(\beta_0, \beta'_0), (\beta_1, \beta'_1) \leq^{\mathsf{h}_0,l}_{\mathsf{c}} (t \sim t', P)$. We can extract from P a proof of:

  $$\beta_0, \beta_0 \sim \beta'_0, \beta'_1$$

  in $\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \overline{\mathsf{CCA2}}$, hence using Lemma 2 we get that $\beta'_0 \equiv \beta'_1$. Therefore we can do the following proof cut elimination: if p' is on the then branch of p then we can rewrite β_1 and β'_1 into true in, respectively, b^{h_0} and b'^{h_0}. We then rewrite the two terms using R to remove the true conditionals. This yields a new proof Q in proof normal form, such that (20) and the induction invariants hold. We do a similar cut elimination with false if p' is in the else of p.
  Finally the result proven at **Step 2** shows that we do not have cross cases $\mathsf{cs\text{-}path}^{\mathsf{h}_0,l}(t, P) \times D^{\mathsf{h}_0}_l$.
- (23.iv): this is a consequence of Corollary 1.(i).
- (23.v): this is a consequence of Lemma 17.

i) Step 4: We conclude by showing that we can get rid of the $\{\overline{\mathsf{BFA}}(b, b')\}$ applications.

Using Corollary 1.(ii) and the proof Q constructed at **Step 3**, we know that for all $\gamma, \gamma' \leq^{\mathsf{h}_0,l}_{\mathsf{l}} (t, Q)$, $\gamma \equiv \gamma'$ (and the same holds for (t', Q)). Therefore there is a proof cut elimination that allows us to remove all $\{\overline{\mathsf{BFA}}(b, b')\}$ applications, by rewriting:

$$D_l\left[\_ \diamond (\gamma)_{\gamma \leq^{\mathsf{h}_0,l}_{\mathsf{l}}(t,Q)}\right] \quad \text{and} \quad D_l\left[\_ \diamond (\gamma')_{\gamma \leq^{\mathsf{h}_0,l}_{\mathsf{l}}(t',Q)}\right]$$

into, respectively, γ_0 and γ'_0 (where $\gamma_0 \leq^{\mathsf{h}_0,l}_{\mathsf{l}} (t, Q)$ and $\gamma'_0 \leq^{\mathsf{h}_0,l}_{\mathsf{l}} (t', Q)$).

j) Conclusion: To conclude, we can first observe that the properties (a),(b) and (c) are implied by, respectively, (i), (ii) and (iii) for $n = 0$. The proof that (d) (resp. (e)) holds is exactly the same than the one we did at **Step 2** (resp. **Step 3**). ■

APPENDIX VII
BOUNDING THE BASIC TERMS

A. α-Bounded Conditionals

We are ready to do the final proof cut eliminations, which will yield derivation of bounded size w.r.t. $|t\downarrow_R| + |t'\downarrow_R|$. To bound the size of cut-free derivations, we are going to bound the size of all normalized basic terms and case-study conditionals appearing in such derivations. To do this, we first introduce the notion of (t, P)-α-bounded terms, where $P \vdash^{\mathsf{npf}} t \sim t'$, and then prove that (t, P)-α-bounded terms are of bounded size w.r.t. $|t\downarrow_R| + |t'\downarrow_R|$. Basically, a term β in $\leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P)$ or $\mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P)$ is (t, P)-α-bounded if we are in one of the following case:

- β is a normalized basic term, and β has a leaf term appearing in $\mathsf{st}(t\downarrow_R)$. Since β is uniquely characterized by its leaf terms, this bound β.
- Let β' be the term matching β on the *right*. If β' shares a leaf term with $\mathsf{st}(t'\downarrow_R)$, then, by the previous observation, β' is bounded. Since β and β' differ only by the content of their encryptions, this also bound β.
- If β is a case-study conditional (i.e. in $\mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P)$), and if there exists a (t, P)-α-bounded normalized basic term ε such that β appears in ε's leaf terms. Indeed, since ε is bounded, it has finitely many leaf terms, which are of bounded size. Hence β is also of bounded size.
- If β is a normalized basic terms used in the sub-proof of $b \sim b'$, where b and b' are (t, P)-α-bounded case-study conditionals, and if b appears in β's leaf terms. Again, since β is uniquely characterized by any of its leaf terms, and since b is bounded, we know that β is bounded.
- Finally, if β is a decryption guard of some decryption oracle call d, where d appears in a (t, P)-α-bounded normalized basic term ζ. Since ζ is bounded, and since β is a sub-term of ζ, the term β is also bounded.

We formally define what is a (t, P)-α-bounded terms.

Definition 44. *For all $P \vdash^{\mathsf{npf}} t \sim t'$, the set of (t, P)-α-bounded terms is the smallest subset of:*

$$\left\{\beta \mid \exists \mathsf{h}, l.\, \beta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P)\right\} \cup \left\{b \mid \exists \mathsf{h}.\, b \in \mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P)\right\}$$

such that for all h, l, for all β $(\leq_{\mathsf{bt}}^{\mathsf{h},l} \cup \mathsf{cs\text{-}path}^{\mathsf{h},l})\ (t, P)$, β is (t, P)-α-bounded if:

- ***Base case:*** *$\mathsf{h} = \epsilon$ and $\mathsf{leave\text{-}st}(\beta\downarrow_R) \cap \mathsf{st}(t\downarrow_R) \neq \emptyset$.*
- ***Base case:*** *$\mathsf{h} = \epsilon$ and there exists β' such that:*

$$(\beta, \beta')\ (\leq_{\mathsf{l}\sim\mathsf{l}}^{\epsilon,l} \cup \leq_{\mathsf{c}\sim\mathsf{c}}^{\epsilon,l} \cup \mathsf{cs\text{-}path}^{\epsilon,l})\ (t \sim t', P)$$

  and $\mathsf{leave\text{-}st}(\beta'\downarrow_R) \cap \mathsf{st}(t'\downarrow_R) \neq \emptyset$.
- ***Inductive case, same label:*** *$\beta \in \mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P)$ and there exists $\varepsilon \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P)$ such that ε is (t, P)-α-bounded and $\beta \in \mathsf{leave\text{-}st}(\varepsilon\downarrow_R)$.*
- ***Inductive case, different labels:*** *$\beta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P)$, there exists h' such that $\mathsf{h} \in \mathsf{cs\text{-}pos}(\mathsf{h}')$ and $b \in \mathsf{cs\text{-}path}^{\mathsf{h}',l}(t, P)$ such that b is (t, P)-α-bounded and $b \in \mathsf{leave\text{-}st}(\beta\downarrow_R)$.*
- ***Inductive case, guard:*** *$\beta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P)$, there exists $\varepsilon \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P)$ such that:*
  - *$\varepsilon \equiv B[\vec{w}, (\alpha_i)_i, (\mathsf{dec}_j)_j]$ is (t, P)-α-bounded.*
  - *β is a guard of a $\mathcal{S}_l^P$-decryption oracle call $d \in (\mathsf{dec}_j)_j$.*

We continue our proof cut eliminations, starting from the derivations constructed in Lemma 18. We let $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$ be the restriction of $\vdash^{\mathsf{npf}}$ to derivations satisfying the properties guaranteed by Lemma 18 which use only (t, P)-α-bounded terms. Moreover, we require that no basic conditionals appears twice on the same branch.

Definition 45. *For all proof P, term t, t', we write $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$ if:*

(I) *$P \vdash^{\mathsf{npf}} t \sim t'$ and the properties (a) to (e) of Lemma 18 hold.*

(II) *The following sets are sets of, respectively, (t, P)-α-bounded and (t', P)-α-bounded terms:*

$$\left\{\beta \mid \exists \mathsf{h}, l.\, \beta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P')\right\} \cup \left\{b \mid \exists \mathsf{h}.\, b \leq_{\mathsf{cs}}^{\mathsf{h}} (t, P')\right\}$$
$$\left\{\beta' \mid \exists \mathsf{h}, l.\, \beta' \leq_{\mathsf{bt}}^{\mathsf{h},l} (t', P')\right\} \cup \left\{b' \mid \exists \mathsf{h}.\, b' \leq_{\mathsf{cs}}^{\mathsf{h}} (t', P')\right\}$$

(III) *For every $l \in \mathsf{label}(\epsilon)$, for every path $\vec{\rho}$ of $\mathcal{S}_l^P$-normalized basic conditional from the root of t to some leave, $\vec{\rho}$ does not contain any duplicates. The same property must hold for t'.*

We can now give the proof of Lemma 10, which we recall below.

Lemma (10). *$\vdash_\alpha^{\mathsf{npf}}$ is complete with respect to $\vdash^{\mathsf{npf}}$.*

Proof. Let P be such that $P \vdash^{\mathsf{npf}} t \sim t'$, where P is obtained using Lemma 18. Therefore P satisfies the item (I) of Definition 45. Now, we are going to build from P a proof P' of $t \sim t'$ that satisfies the item (II) and (III) of Definition 45.

We are going to show that, if there exists β in:

$$\left\{\beta \mid \exists \mathsf{h}, l.\, \beta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P')\right\} \cup \left\{b \mid \exists \mathsf{h}.\, b \leq_{\mathsf{cs}}^{\mathsf{h}} (t, P')\right\}$$

such that β is not (t,P)-α-bounded, then there is a cut elimination removing β (we describe the cut elimination used later in the proof). Moreover, the resulting proof will have a smaller number of basic terms which are not (t,P)-α-bounded, hence we will conclude by induction. First, we want to pick a term β maximal for a carefully chosen relation.

a) Order $<_g$: Let $<_g$ be the transitive closure of the relation $\ll_g$ on:

$$\bigcup_{\mathsf{h} \in \mathsf{index}(P)} \left\{(\beta, \mathsf{h}) \mid \exists l. \beta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P)\right\} \cup \bigcup_{\mathsf{h} \in \mathsf{index}(P)} \left\{(b, \mathsf{h}) \mid \exists l. b \in \mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P)\right\}$$

defined by:

$$(\zeta, \mathsf{h}) \ll_g (\zeta', \mathsf{h}') \text{ iff } \begin{cases} \mathsf{h} = \mathsf{h}' \wedge \zeta, \zeta' \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P) \wedge \ \zeta \text{ is a guard of some decryption oracle call } d \in \mathsf{st}(\zeta') \\ \mathsf{h} = \mathsf{h}' \wedge \zeta \in \mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P) \wedge \zeta' \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P) \wedge \zeta \in \mathsf{leave\text{-}st}(\zeta' \downarrow_R) \\ \mathsf{h} > \mathsf{h}' \wedge \zeta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P) \wedge \zeta' \in \mathsf{cs\text{-}path}^{\mathsf{h}',l}(t, P) \wedge \zeta' \in \mathsf{leave\text{-}st}(\zeta \downarrow_R) \end{cases}$$

First we show that $<_g$ is a strict order. As it is transitive, we just need to show that it is an antisymmetric relation. For all h, the restriction $<_g^{\mathsf{h}}$ of $<_g$ to:

$$\left\{(\beta, \mathsf{h}) \mid \exists l. \beta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P)\right\} \cup \left\{(b, \mathsf{h}) \mid \exists l. b \in \mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P)\right\}$$

is a strict order, as it is included in the embedding relation. To show that $<_g$ is a strict order on its full domain, we simply use the facts that for all h, $<_g^{\mathsf{h}}$ is a strict order and that when we go from the domain of $<_g^{\mathsf{h}}$ to the domain of $<_g^{\mathsf{h}'}$, we have $\mathsf{h}' > \mathsf{h}$.

W.l.o.g. we assume that (β, h) is maximal for $<_g$ among the set of terms that are not (t,P)-α-bounded. Consider an arbitrary l such that $\mathsf{h} \in \mathsf{h\text{-}branch}(l)$. Since β is not (t,P)-α-bounded, we know that if β is a guard of some decryption oracle call $d \in \mathsf{st}(\zeta)$ with $\zeta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t,P)$, then ζ is not (t,P)-α-bounded. By maximality of β, it follows that if $\beta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t,P)$ then β is not a decryption guard of any $\zeta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t,P)$.

b) Case $\mathsf{h} = \epsilon$: First we are going to describe what to do for $\mathsf{h} = \epsilon$. From Lemma 18.(e), we know that for every $l \in \mathsf{label}(P)$, for all $\gamma \leq_{\mathsf{l}}^{\epsilon,l} (t,P)$, the basic term γ is (t,P)-α-bounded. Therefore $\beta \not\leq_{\mathsf{l}}^{\epsilon,l} (t,P)$. Moreover, from Lemma 18.(d) we get that $\beta \leq_{\mathsf{c}}^{\epsilon,l} (t,P)$ and $\beta \in \mathsf{cs\text{-}path}^{\epsilon,l}(t,P)$ are mutually exclusive. Putting everything together, we have three cases:

(i) either $\beta \ (\not\leq_{\mathsf{l}}^{\epsilon,l} \cup \leq_{\mathsf{c}}^{\epsilon,l}) \ (t,P)$ and $\beta \notin \mathsf{cs\text{-}path}^{\epsilon,l}(t,P)$.

(ii) or $\beta \ (\not\leq_{\mathsf{l}}^{\epsilon,l} \cup \not\leq_{\mathsf{c}}^{\epsilon,l}) \ (t,P)$ and $\beta \in \mathsf{cs\text{-}path}^{\epsilon,l}(t,P)$.

(iii) $\beta \ (\not\leq_{\mathsf{l}}^{\epsilon,l} \cup \not\leq_{\mathsf{c}}^{\epsilon,l}) \ (t,P)$ and $\beta \notin \mathsf{cs\text{-}path}^{\epsilon,l}(t,P)$.

We first focus on case i. We explain how to deal with ii and iii later.

- **i, Part 1** Assume that we are in case i). Let β' be such that $(\beta, \beta') \ (\leq_{\mathsf{c}\sim\mathsf{c}}^{\epsilon,l}) \ (t \sim t', P)$. Since β is not (t,P)-α-bounded we know that for all $u \in \mathsf{leave\text{-}st}(\beta \downarrow_R)$, for all $u' \in \mathsf{leave\text{-}st}(\beta' \downarrow_R)$, u and u' are spurious in, respectively, t and t'. We let:

$$t \equiv C\left[\vec{b}_{cs} \diamond D_l\left[(\beta_i)_{i \in J} \diamond (\gamma_m)_{m \in M}\right], \Delta\right]$$
$$t' \equiv C\left[\vec{b}'_{cs} \diamond D_l\left[(\beta'_i)_{i \in J} \diamond (\gamma'_m)_{m \in M}\right], \Delta'\right]$$

  where, for every $i \in J$, $(\beta_i, \beta'_i) \leq_{\mathsf{c}\sim\mathsf{c}}^{\epsilon,l} (t \sim t', P)$, and for every $m \in M$, $(\gamma_m, \gamma'_m) \leq_{\mathsf{l}\sim\mathsf{l}}^{\epsilon,l} (t \sim t', P)$. Moreover, we assume that for every $i \in J$, the hole $[]_i$ (which is mapped to β_i) appears exactly once in D_l. We define the set of indices $I = \{i \in J \mid \beta \equiv \beta_i\}$. Using Corollary 1.(i), we know that:

$$I = \{i \in J \mid \mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \mathsf{leave\text{-}st}(\beta_i \downarrow_R) \neq \emptyset\}$$

  We know that we have a proof of $(\beta_i)_{i \in I} \sim (\beta'_i)_{i \in I}$ in the fragment $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \overline{\mathsf{CCA2}})$. Therefore:

$$\forall b, b' \in \{\beta'_i \mid i \in I\}, b \equiv b' \equiv \beta' \tag{22}$$

  Indeed, if $|I| = 1$ then this is obvious, and if $|I| > 1$ we use Lemma 2 (since all the terms on the left are the same). We let $I' = \{i \in J \mid \beta' \equiv \beta'_i\}$. Using the same proof than for I, we know that $I' = \{i \in J \mid \mathsf{leave\text{-}st}(\beta' \downarrow_R) \cap \mathsf{leave\text{-}st}(\beta'_i \downarrow_R) \neq \emptyset\}$. We deduce from this that:

$$\forall b, b' \in \{\beta_i \mid i \in I'\}, b \equiv b' \equiv \beta \tag{23}$$

From (22) we get that $I \subseteq I'$ and conversely from (23) we get that $I' \subseteq I$. Therefore we have the equality $I = I'$.

- **i, Part 2** For every $i \notin I$, using Lemma 15 on β we know that there exists $\tilde{\beta}_i[]$ such that:

$$\tilde{\beta}_i[\beta] \equiv \beta_i \qquad \text{and} \qquad \mathsf{leave\text{-}st}(\beta\downarrow_R) \cap \mathsf{cond\text{-}st}(\tilde{\beta}_i[]\downarrow_R) = \emptyset$$

Similarly, for every $m \in M$, there exists $\tilde{\gamma}_m[]$ such that:

$$\tilde{\gamma}_m[\beta] \equiv \gamma_m \qquad \text{and} \qquad \mathsf{leave\text{-}st}(\beta\downarrow_R) \cap \mathsf{cond\text{-}st}(\tilde{\gamma}_m[]\downarrow_R) = \emptyset$$

Then we have:

$$\begin{aligned} t &\equiv C\left[\vec{b}_{cs} \diamond \left(D_l\left[(\beta_i)_{i\in J} \diamond (\gamma_m)_{m\in M}\right], \Delta\right)\right] \\ &\equiv C\left[\vec{b}_{cs} \diamond \left(D_l\left[\left((\beta)_{i\in I}, (\tilde{\beta}_i[\beta])_{i\notin I}\right) \diamond (\tilde{\gamma}_m[\beta])_{m\in M}\right], \Delta\right)\right] \end{aligned}$$

Let $C_\beta[\vec{b}_\beta \diamond \vec{u}_\beta] \equiv \beta\downarrow_R$. We have:

$$\begin{aligned} & D_l\left[\left((\beta)_{i\in I}, (\tilde{\beta}_i[\beta])_{i\notin I}\right) \diamond (\tilde{\gamma}_m[\beta])_{m\in M}\right] \\ =_R\ & \mathsf{if}\ C_\beta[\vec{b}_\beta \diamond \vec{u}_\beta]\ \mathsf{then}\ D_l\left[\left((\mathsf{true})_{i\in I}, (\tilde{\beta}_i[\mathsf{true}])_{i\notin I}\right) \diamond (\tilde{\gamma}_m[\mathsf{true}])_{m\in M}\right] \\ & \qquad\qquad\qquad\quad \mathsf{else}\ D_l\left[\left((\mathsf{false})_{i\in I}, (\tilde{\beta}_i[\mathsf{false}])_{i\notin I}\right) \diamond (\tilde{\gamma}_m[\mathsf{false}])_{m\in M}\right] \end{aligned}$$

Since $\vec{u}_\beta = \mathsf{leave\text{-}st}(\beta\downarrow_R)$, for every $u \in \vec{u}_\beta$, $i \in J$ and $m \in M$, we know that $u \notin \mathsf{cond\text{-}st}(\tilde{\beta}_i[]\downarrow_R)$ and $u \notin \mathsf{cond\text{-}st}(\tilde{\gamma}_m[]\downarrow_R)$. Let $\vec{\rho}$ be the conditionals appearing on the path from the root of t to $D_l[\_]$. Using Lemma 18.(d), we know that $\vec{u}_\beta \cap \vec{\rho} = \emptyset$. Let $(u_o)_{o\in O}$ be such that $\vec{u} \equiv (u_o)_{o\in O}$. By applying Proposition 20 to all u we know that:

$$\begin{aligned} & C\left[\vec{b}_{cs} \diamond \begin{pmatrix} \mathsf{if}\ C_\beta\left[\vec{b}_\beta \diamond \vec{u}_\beta\right]\ \mathsf{then}\ D_l\left[\left((\mathsf{true})_{i\in I}, (\tilde{\beta}_i[\mathsf{true}])_{i\notin I}\right) \diamond (\tilde{\gamma}_i[\mathsf{true}])_m\right] \\ \mathsf{else}\ D_l\left[\left((\mathsf{false})_{i\in I}, (\tilde{\beta}_i[\mathsf{false}])_{i\notin I}\right) \diamond (\tilde{\gamma}_i[\mathsf{false}])_m\right], \Delta \end{pmatrix}\right] \\ =_R\ & C\left[\vec{b}_{cs} \diamond \begin{pmatrix} \mathsf{if}\ C_\beta\left[\vec{b}_\beta \diamond (\mathsf{true})_o\right]\ \mathsf{then}\ D_l\left[\left((\mathsf{true})_{i\in I}, (\tilde{\beta}_i[\mathsf{true}])_{i\notin I}\right) \diamond (\tilde{\gamma}_i[\mathsf{true}])_m\right] \\ \mathsf{else}\ D_l\left[\left((\mathsf{false})_{i\in I}, (\tilde{\beta}_i[\mathsf{false}])_{i\notin I}\right) \diamond (\tilde{\gamma}_i[\mathsf{false}])_m\right], \Delta \end{pmatrix}\right] \\ =_R\ & C\left[\vec{b}_{cs} \diamond \left(D_l\left[\left((\mathsf{true})_{i\in I}, (\tilde{\beta}_i[\mathsf{true}])_{i\notin I}\right) \diamond (\tilde{\gamma}_i[\mathsf{true}])_m\right], \Delta\right)\right] \end{aligned} \tag{24}$$

- **i, Part 2.b** We do exactly the same thing on the other side: for all $i \notin I$ we know that there exists $\tilde{\beta}'_i[]$ such that:

$$\tilde{\beta}'_i[\beta'] \equiv \beta'_i \qquad \text{and} \qquad \mathsf{leave\text{-}st}(\beta'\downarrow_R) \cap \mathsf{cond\text{-}st}(\tilde{\beta}'_i[]\downarrow_R) = \emptyset$$

And, for every $m \in M$, there exists $\tilde{\gamma}'_m[]$ such that:

$$\tilde{\gamma}'_m[\beta'] \equiv \gamma'_m \qquad \text{and} \qquad \mathsf{leave\text{-}st}(\beta'\downarrow_R) \cap \mathsf{cond\text{-}st}(\tilde{\gamma}'_m[]\downarrow_R) = \emptyset$$

Then by the same reasoning we have:

$$\begin{aligned} t' &\equiv C\left[\vec{b}'_{cs} \diamond \left(D_l\left[(\beta'_i)_i \diamond (\gamma'_m)_{m\in M}\right], \Delta'\right)\right] \\ &\equiv C\left[\vec{b}'_{cs} \diamond \left(D_l\left[\left((\beta')_{i\in I}, (\tilde{\beta}'_i[\beta'])_{i\notin I}\right) \diamond (\tilde{\gamma}'_m[\beta'])_{m\in M}\right], \Delta'\right)\right] \\ &=_R C\left[\vec{b}'_{cs} \diamond \left(D_l\left[\left((\mathsf{true})_{i\in I}, (\tilde{\beta}'_i[\mathsf{true}])_{i\notin I}\right) \diamond (\tilde{\gamma}'_m[\mathsf{true}])_{m\in M}\right], \Delta'\right)\right] \end{aligned} \tag{25}$$

Observe that corresponding sub-terms of (24) and (25) can be matched to corresponding sub-terms of t and t'. It is straightforward to build a proof of the equivalence of (24) and (25) using P, except for the $\mathsf{CCA2}$ applications side-conditions. We argue why the side-conditions carry over from the derivation P later in the proof.

- **ii and iii** The case ii works similarly to the case i, except that we use Lemma 9 instead of Lemma 2. The case iii is exactly like the case i when taking $I = \emptyset$.

c) Case $\mathsf{h} \neq \epsilon$*:* In that case, thanks to Lemma 18.(a), we know that $\beta \not\leq_{\mathsf{c}}^{\mathsf{h},l} (t, P)$. We have three cases:

(a) either $\beta \leq_{\mathsf{l}}^{\mathsf{h},l} (t, P)$: using Lemma 18.(c), there exists h_0, b^h such that $\mathsf{h} \in \mathsf{cs\text{-}pos}(\mathsf{h}_0)$, $b^h \in \mathsf{cs\text{-}path}^{\mathsf{h}_0,l}(t, P)$ and $(b^h\downarrow_R) \in \mathsf{leave\text{-}st}(\beta\downarrow_R)$. Since $\mathsf{h} \in \mathsf{cs\text{-}pos}(\mathsf{h}_0)$ implies that $\mathsf{h}_0 < \mathsf{h}$, we know that $\beta <_g b^h$. We then have two cases. Either b^h is (t, P)-α-bounded, and then using the inductive case for different labels of the definition of (t, P)-α-bounded terms, we know that β is (t, P)-abounded. Absurd. Or b^h is not (t, P)-α-bounded, which contradicts the maximality of β among the set of terms which are not (t, P)-abounded. Absurd.

(b) either $\beta \not\leq_{\mathsf{l}}^{\mathsf{h},l} (t, P)$ and $\beta \in \mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P)$: this case is done exactly like case (ii).

(c) either $\beta \not\leq_{\mathsf{l}}^{\mathsf{h},l} (t, P)$ and $\beta \notin \mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P)$: this case is done exactly like case (iii).

d) Valid Proof Rewriting: We do the rewritings described above for every h such that (β, h) is maximal for $<_g$, and for every l such that $\beta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P)$ or $\beta \in \mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P)$, *simultaneously*. It remains to check that this is a valid cut elimination. The only difficulty lies in checking that all the side-conditions of the $\mathsf{CCA2}$ axiom hold. This is tedious, but here are the key ingredients:

- β is not a guard, and the encryptions that need to be guarded in a decryption are invariant by our proof cut elimination. Therefore decryptions that were well-guarded before are still well-guarded after the cut.
- We did the proof rewriting simultaneously for all h such that (β, h) is maximal for $<_g$. Consider h' such that (β, h') is not maximal for $<_g$: then there exists h such that (β, h) is maximal for $<_g$ and $\mathsf{h} < \mathsf{h}'$. Therefore, the sub-proof at index h' is removed by the proof rewriting. This ensure that, for all branch l where a rewriting occurred, we removed all occurrences of β. Therefore, if an encryption used to contain β then all occurrences of this encryption have been rewritten in the same way. This guarantees that the freshness condition on encryption randomness still holds.
- The length constraints on encryption oracle calls still holds thanks to the branch invariance property of the length predicate $\mathsf{EQL}(\_, \_)$.

e) Conclusion: This concludes the proof of the second bullet point of the definition $\vdash_\alpha^{\mathsf{npf}}$. The third bullet point is much simpler. We want to show that for all $l \in \mathsf{label}(\epsilon)$, for every path $\vec{\rho}$ of $\mathcal{S}_l^P$-normalized basic conditional from the root of t to some leave, $\vec{\rho}$ does not contain any duplicates. We show this by proof cut elimination as follows: let $(\beta, \beta'_0) \leq_{\mathsf{c}\sim\mathsf{c}}^{\epsilon,l} (t, P)$ and $(\beta, \beta'_1) \leq_{\mathsf{c}\sim\mathsf{c}}^{\epsilon,l} (t, P)$, using Lemma 2 we have $\beta'_0 \equiv \beta'_1$. Since they are on the same branch, one may rewrite the lowest occurrence of β and β'_0 into their then branch (we could also use the else branch). This yield a smaller proof, and one can check that all the other properties are invariant of this proof cut elimination. We directly concludes by induction. ■

B. Bounding the Number of Nested Basic Conditionals

We use the previous lemma to bound the number of basic conditionals appearing in a proof $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$. Looking at the definition of (t, P)-α-bounded terms, one may try to show that for every $\beta \in (\leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P) \cup \mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P))$, if β is (t, P)-α-bounded then there exists $u \in \mathsf{leave\text{-}st}(\beta \downarrow_R)$ such that $u \in \mathsf{st}(t \downarrow_R) \cup \mathsf{st}(t' \downarrow_R)$. Since $\mathsf{st}(t \downarrow_R) \cup \mathsf{st}(t' \downarrow_R)$ is finite, and since a basic term is uniquely characterized by any of its leaves, this would allow us to bound the number of basic terms appearing in $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$.

Unfortunately, this is not always the case. Indeed, consider $(\beta, \beta') \leq_{\mathsf{c}}^{\mathsf{h},l} (t \sim t', P)$ such that β' has a leaf term appearing in t', but β shares no leaf term with β' nor t:

$$\mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \mathsf{leave\text{-}st}(\beta' \downarrow_R) = \emptyset \qquad \mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \mathsf{st}(t \downarrow_R) = \emptyset \qquad \mathsf{leave\text{-}st}(\beta' \downarrow_R) \cap \mathsf{st}(t' \downarrow_R) \neq \emptyset$$

β' is α-bounded since it shares a leaf term with t', and using the second case, β is α-bounded too. But β shares no leaf term with t and t'.

Still, we can bound β. Since $(\beta, \beta') \leq_{\mathsf{c}}^{\mathsf{h},l} (t \sim t', P)$, we observe that $\beta \equiv B[\vec{w}, (\alpha_i)_i, (\mathsf{dec}_j)_j]$ and $\beta' \equiv B[\vec{w}, (\alpha'_i)_i, (\mathsf{dec}'_j)_j]$. Using the fact that $\mathsf{leave\text{-}st}(\beta' \downarrow_R) \cap \mathsf{st}(t' \downarrow_R)$ and that β is a $\mathcal{S}_l$-normalized basic term, we know that every leaf $u \in \mathsf{leave\text{-}st}(\beta \downarrow_R)$ is in $\mathsf{st}(t' \downarrow_R)$, *modulo the content of the $\mathcal{S}_l$-encryption oracle calls*. This motivate the introduction of the notion of *leaf frame*.

a) Leaf frame: Let β be a $\mathcal{S}_l$-normalized basic term, and $u, v \in \mathsf{leave\text{-}st}(\beta \downarrow_R)$ be leaf terms of β. Then u and v only differ by their encryptions. That is, if one replace all the zero decryptions $\mathbf{0}(\mathsf{dec}(\_, \mathsf{sk}))$ by $\mathsf{dec}(\_, \mathsf{sk})$, and all the leaves of encryptions $\{m\}_{\mathsf{pk}}^{\mathsf{n}}$ by $\{[]_\alpha\}_{\mathsf{pk}}^{\mathsf{n}}$ (where α is the unique term of $\mathcal{E}_l$ such that $\alpha \equiv \{\_\}_{\mathsf{pk}}^{\mathsf{n}}$) in u and in v then you get the same context. We formalize this below, and use it to generalize Proposition 16.

Definition 46. *Let $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$ and l be a branch label in $\mathsf{label}(P)$. We define the left* leaf frame $\mathsf{l\text{-}frame}_l^P$ *of $\beta \in (\leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P) \cup \mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P))$ inductively as follows:*

$$\mathsf{l\text{-}frame}_l^P(s) \;\equiv\; \begin{cases} \{[]_\alpha\}_{\mathsf{pk}}^{n} & \text{if } \exists \alpha \equiv \{m\}_{\mathsf{pk}}^{n} \in \mathcal{E}_l^P \wedge s \equiv \{\_\}_{\mathsf{pk}}^{n} \\ \mathsf{dec}(\mathsf{l\text{-}frame}_l^P(s), \mathsf{sk}) & \text{if } \mathsf{sk} \in \mathcal{K}_l^P \wedge s \equiv \mathbf{0}(\mathsf{dec}(s, \mathsf{sk})) \\ \mathsf{l\text{-}frame}_l^P(v) & \text{if } s \equiv \mathsf{if}\, b \;\mathsf{then}\; u \;\mathsf{else}\; v \\ f((\mathsf{l\text{-}frame}_l^P(u_i))_i) & \text{otherwise} \end{cases}$$

We also let $\underline{\mathsf{l\text{-}frame}}_l^P(\beta)$ be $\mathsf{l\text{-}frame}_l^P(\beta)$ where we make every hole variable appear at most once, by replacing a hole variable $[]_\alpha$ occurring at position p in β by $[]_{\alpha,p}$.

We define the right leaf frame $\mathsf{r\text{-}frame}_l^P$ *(and its underlined version $\underline{\mathsf{r\text{-}frame}}_l^P$) of $\beta \in (\leq_{\mathsf{bt}}^{\mathsf{h},l} (t', P) \cup \mathsf{cs\text{-}path}^{\mathsf{h},l}(t', P))$, using $\mathcal{E}_l'^P$ instead of $\mathcal{E}_l^P$.*

Remark 10. We have two remarks:

- We state some results only for $\mathsf{l\text{-}frame}$. The corresponding results for $\mathsf{r\text{-}frame}$ are obtained by symmetry.

- The hole variables in $\underline{\text{l-frame}}_l^P(\beta)$ are annotated by both the position p of the hole *and* the encryption α that appears at p in β. By consequence, if two normalized basic terms β and β' are such that $\underline{\text{l-frame}}_l^P(\beta)$ and $\underline{\text{l-frame}}_l^P(\beta')$ share a hole variable $[]_{\alpha,p}$, it means that β and β' contain the *same encryption α at the same position p*. This is crucial, as we want $\underline{\text{l-frame}}_l^P$ to uniquely characterize normalized basic terms.

Example 11. For all $\mathcal{S}_l^P$-decryption oracle call dec guarding $\mathsf{dec}(s[(\alpha_i)_i, (\mathsf{dec}_j)_j], \mathsf{sk})$, if for all i, $\alpha_i \equiv \{\_\}_{\mathsf{pk}_i}^{\mathsf{n}_i}$ then:

$$\text{l-frame}_l^P(\mathsf{dec}) \equiv \mathsf{dec}\Big(s\Big[\big(\{[]_{\alpha_i}\}_{\mathsf{pk}_i}^{\mathsf{n}_i}\big)_i, \big(\text{l-frame}_l^P(\mathsf{dec}_j)\big)_j\Big], \mathsf{sk}\Big)$$

We also give an example of $\underline{\text{l-frame}}_l^P$. Assuming that $\alpha_0 \equiv \{A\}_{\mathsf{pk}}^{\mathsf{n}_0}$ and $\alpha_1 \equiv \{B\}_{\mathsf{pk}}^{\mathsf{n}_1}$ are encryptions in $\mathcal{E}_l^P$:

$$\underline{\text{l-frame}}_l^P\left(\langle \alpha_0 \,,\, \langle \alpha_1 \,,\, \alpha_0 \rangle\rangle\right) \equiv \langle \{[]_{\alpha_0,00}\}_{\mathsf{pk}}^{\mathsf{n}_0} \,,\, \langle \{[]_{\alpha_1,100}\}_{\mathsf{pk}}^{\mathsf{n}_1} \,,\, \{[]_{\alpha_0,110}\}_{\mathsf{pk}}^{\mathsf{n}_0} \rangle\rangle$$

Proposition 28. *Let $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$ and $l \in \mathsf{label}(P)$. Let b be an if-free term in R-normal form. For every $\mathcal{S}_l$-normalized basic terms γ, if $b \in \text{leave-st}(\gamma \downarrow_R)$ then $\text{l-frame}_l^P(b) \equiv \text{l-frame}_l^P(\gamma)$.*

Proof. This is by induction on the size of γ. ∎

Proposition 29. *Let $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$ and $l \in \mathsf{label}(P)$. For every $\mathcal{S}_l$-normalized basic terms β, β', if $\text{l-frame}_l^P(\beta) \equiv \text{l-frame}_l^P(\beta')$ then $\beta \equiv \beta'$.*

Proof. The proof is exactly the same than for Proposition 16. ∎

Proposition 30. *Let $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$ and $l \in \mathsf{label}(P)$. For all h, if $(b, b') \leq_{\mathsf{cs}\sim\mathsf{cs}}^{\mathsf{h},l} (t \sim t', P)$ then there exists h' and $(\gamma, \gamma') \left(\leq_{\mathsf{c}\sim\mathsf{c}}^{\mathsf{h}',l} \cup \leq_{\mathsf{l}\sim\mathsf{l}}^{\mathsf{h}',l}\right) (t \sim t', P)$ such that $b \in \text{leave-st}(\gamma \downarrow_R)$ and $b' \in \text{leave-st}(\gamma' \downarrow_R)$.*

Proof. Let h, x be such that $\mathsf{h} = h_{\mathsf{x}}$. Let $h_0 \in \text{cs-pos}(\text{extract}_{\mathsf{x}}(h, P))$ and x_0 be such that x_0 is the direction taken in l at position h_0, and such that $Q = \text{extract}_{\mathsf{x}_0}(h_0, P)$ is a proof of $b \sim b'$.

Using the fact that the sub-proofs of $\mathsf{CS}_\Box$ conditionals of P do not use the $\overline{\mathsf{BFA}}$ rule, we know that Q lies in the fragment:

$$\mathfrak{F}(\mathsf{CS}_\Box \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \overline{\mathsf{CCA2}})$$

Let $(\gamma, \gamma') \leq_{\mathsf{l}\sim\mathsf{l}}^{\epsilon,l} (b \sim b', Q)$. Using the property (c) of Lemma 18 (which holds thanks to $\vdash_\alpha^{\mathsf{npf}}$), we know that $b \in \text{leave-st}(\gamma \downarrow_R)$ and $b \in \text{leave-st}(\gamma' \downarrow_R)$. ∎

Proposition 31. *Let $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$ and $l \in \mathsf{label}(P)$. For all h, if $(\beta, \beta') \left(\leq_{\mathsf{c}\sim\mathsf{c}}^{\mathsf{h},l} \cup \leq_{\mathsf{l}\sim\mathsf{l}}^{\mathsf{h},l} \cup \text{cs-path}_\sim^{\mathsf{h},l}\right) (t \sim t', P)$ then $\text{l-frame}_l^P(\beta) \equiv \text{r-frame}_l^P(\beta')$.*

Proof. First we deal with the case $(\beta, \beta') \left(\leq_{\mathsf{c}\sim\mathsf{c}}^{\mathsf{h},l} \cup \leq_{\mathsf{l}\sim\mathsf{l}}^{\mathsf{h},l}\right) (t \sim t', P)$. We know that we can extract a proof Q (from P) such that $Q \vdash_\alpha^{\mathsf{npf}} \beta \sim \beta'$ and Q is in the fragment $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \overline{\mathsf{CCA2}})$. The result follows from the definitions of l-frame_l^P and r-frame_l^P.

Now we deal with the case $(\beta, \beta')\ (\text{cs-path}_\sim^{\mathsf{h},l})\ (t \sim t', P)$. Using Proposition 30 we know that there exists h' and $(\gamma, \gamma') \left(\leq_{\mathsf{c}\sim\mathsf{c}}^{\mathsf{h}',l} \cup \leq_{\mathsf{l}\sim\mathsf{l}}^{\mathsf{h}',l}\right) (t \sim t', P)$ such that $\beta \in \text{leave-st}(\gamma \downarrow_R)$ and $\beta' \in \text{leave-st}(\gamma' \downarrow_R)$. Since β is if-free and in R-normal form, we obtain that $\text{l-frame}_l^P(\beta) \equiv \text{l-frame}_l^P(\gamma)$ by applying Proposition 28. Similarly $\text{r-frame}_l^P(\beta') \equiv \text{r-frame}_l^P(\gamma')$. Moreover, from the previous case, we get that $\text{l-frame}_l^P(\gamma) \equiv \text{r-frame}_l^P(\gamma')$. Hence $\text{l-frame}_l^P(\beta) \equiv \text{r-frame}_l^P(\beta')$. ∎

Proposition 32. *Let $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$ and $l \in \mathsf{label}(P)$. For every $\mathcal{S}_l$-normalized basic terms β, β', $\text{l-frame}_l^P(\beta) \equiv \text{l-frame}_l^P(\beta')$ if and only if $\underline{\text{l-frame}}_l^P(\beta) \equiv \underline{\text{l-frame}}_l^P(\beta')$.*

Proof. This is obvious, since the hole variable annotations in $\underline{\text{l-frame}}_l^P$ uniquely characterize both the position of the hole and the encryption appearing at this position. ∎

Proposition 33. *Let $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$ and $l \in \mathsf{label}(P)$. For every $\mathcal{S}_l$-normalized basic terms β, β' and substitutions θ, θ', if $\underline{\text{l-frame}}_l^P(\beta)\theta \equiv \underline{\text{l-frame}}_l^P(\beta')\theta'$ then $\underline{\text{l-frame}}_l^P(\beta) \equiv \underline{\text{l-frame}}_l^P(\beta')$.*

Proof. We prove this by induction on the size of β. The base case is trivial, lets deal with the inductive case. Let β and β' be $\mathcal{S}_l^P$-normalized basic terms, we know that $\beta \equiv B[\vec{w}, (\alpha_i)_i, (\mathsf{dec}_j)_j]$ where:

- for every i, $\alpha_i \equiv \{m_i\}_{\mathsf{pk}_i}^{\mathsf{n}_i} \in \mathcal{E}_l^P$.
- for every j, dec_j is a decryption oracle call for $\mathsf{dec}(s_j, \mathsf{sk}_j)$ in $\mathcal{D}_l^P$.

Similarly, we have a decomposition of β' into $B'[\vec{w}', (\alpha_i')_i, (\mathsf{dec}_j')_j]$. By definition of l-frame_l^P, and using the fact that $\mathsf{fresh}(\mathcal{R}_l^P; \vec{w})$, we have:

$$\text{l-frame}_l^P(\beta) \equiv B[\vec{w}, (\{[]_{\alpha_i}\}_{\mathsf{pk}_i}^{\mathsf{n}_i})_i, \mathsf{dec}(\text{l-frame}_l^P(s_j), \mathsf{sk}_j)]$$

Similarly:

$$\text{l-frame}_l^P(\beta') \equiv B'[\vec{w}', (\{[]_{\alpha'_i}\}_{\mathsf{pk}'_i}^{\mathsf{n}'_i})_i, \mathsf{dec}(\text{l-frame}_l^P(s'_j), \mathsf{sk}'_j)]$$

We have three cases:

- Either $\beta \equiv \{m\}_{\mathsf{pk}}^{\mathsf{n}} \in \mathcal{E}_l^P$. Then $\underline{\text{l-frame}}_l^P(\beta) \equiv \{[]_{\beta,0}\}_{\mathsf{pk}}^{\mathsf{n}}$. By definition of l-frame, and using the fact that $\underline{\text{l-frame}}_l^P(\beta)\theta \equiv \underline{\text{l-frame}}_l^P(\beta')\theta'$, we get that β' is of the form $\{m'\}_{\mathsf{pk}}^{\mathsf{n}}$. We deduce from the freshness side condition of n that $m' \equiv m$.
- Or $\beta \equiv \mathsf{dec}$ where dec is a $\mathcal{S}_l^P$-decryption oracle call guarding $\mathsf{dec}(s, \mathsf{sk})$. Then $\underline{\text{l-frame}}_l^P(\beta) \equiv \mathsf{dec}(\underline{\text{l-frame}}_l^P(s), \mathsf{sk})\mu$, where μ is the substitution that lifts positions of s into positions of $\mathsf{dec}(s, \mathsf{sk})$, i.e. for every $\alpha \in \mathcal{E}_l^P$ and position $p \in \mathsf{pos}(s)$:

  $$\mu([]_{\alpha,p}) \equiv []_{\alpha,0\cdot p}$$

  By definition of l-frame, and using the fact that $\underline{\text{l-frame}}_l^P(\beta)\theta \equiv \underline{\text{l-frame}}_l^P(\beta')\theta'$ and that β' is a $\mathcal{S}_l^P$-normalized basic term, we get that β' is also some dec' where dec' is a $\mathcal{S}_l^P$-decryption oracle call guarding $\mathsf{dec}(s', \mathsf{sk})$.
  Moreover we have $\underline{\text{l-frame}}_l^P(s)\mu\theta \equiv \text{l-frame}_l^P(s')\mu\theta$, and s, s' are $\mathcal{S}_l^P$-normalized basic terms. Hence by induction hypothesis $\underline{\text{l-frame}}_l^P(s) \equiv \underline{\text{l-frame}}_l^P(s')$, which concludes this case.
- Or we are not in one of the two cases above. Then, there exists $f \in \mathcal{F} \setminus \{\text{if\_then\_else\_}, \mathbf{0}\}$ s.t. $\beta \equiv f(u_1, \dots, u_n)$ and $\beta' \equiv f(u'_1, \dots, u'_n)$, where $u_1, \dots, u_n$ and $u'_1, \dots, u'_n$ are $\mathcal{S}_l^P$-normalized basic term. Hence $\underline{\text{l-frame}}_l^P(\beta)$ and $\underline{\text{l-frame}}_l^P(\beta')$ both starts with the function symbol f.
  Moreover, if we let, for very $1 \le i \le n$, μ_i be the lifting substitution such that, for every $\alpha \in \mathcal{E}_l^P$ and position p, $\mu_i([]_{\alpha,p}) \equiv []_{\alpha,i\cdot p}$, then:

  $$\underline{\text{l-frame}}_l^P(\beta) \equiv f(\underline{\text{l-frame}}_l^P(u_1)\mu_1, \dots, \underline{\text{l-frame}}_l^P(u_n)\mu_n) \qquad \underline{\text{l-frame}}_l^P(\beta') \equiv f(\underline{\text{l-frame}}_l^P(u'_1)\mu_1, \dots, \underline{\text{l-frame}}_l^P(u'_n)\mu_n)$$

  We apply θ to the equations above, and use the fact that $\underline{\text{l-frame}}_l^P(\beta)\theta \equiv \underline{\text{l-frame}}_l^P(\beta')\theta$:

  $$\begin{aligned} f(\underline{\text{l-frame}}_l^P(u_1)\mu_1\theta, \dots, \underline{\text{l-frame}}_l^P(u_n)\mu_n\theta) &\equiv \underline{\text{l-frame}}_l^P(\beta)\theta \\ &\equiv \underline{\text{l-frame}}_l^P(\beta')\theta \\ &\equiv f(\underline{\text{l-frame}}_l^P(u'_1)\mu_1\theta, \dots, \underline{\text{l-frame}}_l^P(u'_n)\mu_n\theta) \end{aligned}$$

  Hence, for every $1 \le i \le n$, $\underline{\text{l-frame}}_l^P(u_i)\mu_i\theta \equiv \underline{\text{l-frame}}_l^P(u'_i)\mu_i\theta$. By induction hypothesis, we deduce that $\underline{\text{l-frame}}_l^P(u_i) \equiv \underline{\text{l-frame}}_l^P(u'_i)$. Therefore $\underline{\text{l-frame}}_l^P(\beta) \equiv \underline{\text{l-frame}}_l^P(\beta')$. ∎

Definition 47. *We let $<_{\mathsf{st}}$ be the strict, well-founded, subterm ordering.*

b) Nested Sequences of Basic Conditionals: We want to bound the number of nested basic conditional appearing in $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$. Using the contrapositive of Proposition 29, we know that when $\beta <_{\mathsf{st}} \beta'$ we have $\text{l-frame}_l^P(\beta) \not\equiv \text{l-frame}_l^P(\beta')$. Moreover, using Proposition 32 and Proposition 33, we know that $\text{l-frame}_l^P(\beta) \not\equiv \text{l-frame}_l^P(\beta')$ implies that $\underline{\text{l-frame}}_l^P(\beta)\theta \not\equiv \underline{\text{l-frame}}_l^P(\beta')\theta'$ (for every substitutions θ, θ').

Therefore, for any sequence of nested $\mathcal{S}_l^P$-normalized basic conditionals:

$$\beta_1 <_{\mathsf{st}} \cdots <_{\mathsf{st}} \beta_n$$

and for any substitutions $\theta_1, \dots, \theta_n$, we know that $(\underline{\text{l-frame}}_l^P(\beta_i)\theta_i)_{1 \le i \le n}$ is a sequence of pair-wise distinct terms. Tu use this, we prove that there there exists a sequence of substitutions $\theta_1, \dots, \theta_n$ such that:

$$\left\{\underline{\text{l-frame}}_l^P(\beta_1)\theta_1, \dots, \underline{\text{l-frame}}_l^P(\beta_n)\theta_n\right\} \subseteq \mathcal{B}(t, t')$$

where $\mathcal{B}(t, t')$ is a set of bounded size w.r.t. $|t| + |t'|$. Since the $(\text{l-frame}_l^P(\beta_i)\theta_i)_{1 \le i \le n}$ are pair-wise distinct, using a pigeon-hole argument we get that $n \le |\mathcal{B}(t, t')|$.

We outline the end of this sub-section. First, we define the set of terms $\mathcal{B}(t, t')$, and show the existence of the substitutions $(\theta_i)_i$. Then, we bound the size of $\mathcal{B}(t, t')$. Finally, we bound the number of nested basic conditional n using a pigeon-hole argument.

Definition 48. *Let u be an if-free term. We let $\zeta_{\mathcal{K}}(u)$ be the set of terms obtained from u by replacing some occurrences of $\mathbf{0}(\mathsf{dec}(w, \mathsf{sk}))$ by $\mathsf{dec}(w, \mathsf{sk})$ (where $\mathsf{sk} \in \mathcal{K}$), non-deterministically stopping at some encryptions. Formally:*

$$\zeta_{\mathcal{K}}(u) = \begin{cases} \{\mathsf{dec}(v, \mathsf{sk}) \mid w \in v \in \zeta_{\mathcal{K}}(w)\} & \text{if } u \equiv \mathbf{0}(\mathsf{dec}(w, \mathsf{sk})) \text{ and } \mathsf{sk} \in \mathcal{K} \\ \{u\} \cup \{\{v\}_{\mathsf{pk(n)}}^{\mathsf{n}_r} \mid v \in \zeta_{\mathcal{K}}(m)\} & \text{if } u \equiv \{m\}_{\mathsf{pk(n)}}^{\mathsf{n}_r} \text{ and } \mathsf{sk(n)} \in \mathcal{K} \\ \{f(v_1, \dots, v_n) \mid \forall i, v_i \in \zeta_{\mathcal{K}}(u_i)\} & \text{otherwise, where } u \equiv f(u_1, \dots, u_n) \end{cases}$$

Moreover, given a set of ground terms $\mathcal{S}$, we let $\textit{guards}_{\mathcal{K}}(\mathcal{S})$ be an over-approximation of the set of guards of terms in $\mathcal{S}$:

$$\textit{guards}_{\mathcal{K}}(\mathcal{S}) = \left\{ \textit{eq}(s, \alpha) \mid \textit{dec}(s, \textit{sk}(\textit{n})) \in \mathcal{S} \wedge \alpha \equiv \{\_\}_{\textit{pk}(\textit{n})}^{\_} \in \mathsf{st}(s) \wedge \textit{sk}(\textit{n}) \in \mathcal{K} \right\}$$

Definition 49. *Let $\mathcal{S}_{\mathsf{k}}(t)$ be the set of private keys appearing in $t \downarrow_R$, i.e. $\mathcal{S}_{\mathsf{k}}(t) = \{\textit{sk}(\textit{n}) \mid \textit{sk}(\textit{n}) \in \mathsf{st}(t \downarrow_R)\}$. For every term t, we let $\mathcal{B}(t)$ be the set:*

$$\mathcal{B}(t) = \bigcup_{\mathcal{K} \subseteq \mathcal{S}_{\mathsf{k}}(t)} \bigcup_{\substack{u \,\in\, \mathsf{st}(\mathsf{leave\text{-}st}(t \downarrow_R)) \\ \vee u \,\in\, \mathsf{st}(\mathsf{cond\text{-}st}(t \downarrow_R))}} \zeta_{\mathcal{K}}(u) \cup \textit{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u))$$

Moreover, we let $\mathcal{B}(t, t') = \mathcal{B}(t) \cup \mathcal{B}(t')$.

Proposition 34. *Let $P \vdash_{\alpha}^{\textit{npf}} t \sim t'$ and $l \in \textit{label}(P)$. Let β be a $\mathcal{S}_l^P$-normalized basic conditional. Then, for every $u \in \mathsf{leave\text{-}st}(\beta \downarrow_R)$, there exists θ such that $\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \in \zeta_{\mathcal{K}}(u)$.*

Proof. We show this by induction on $|\beta|$.

- If β is an encryption $\{m\}_{\mathsf{pk}}^{\mathsf{n}} \in \mathcal{E}_l^P$, then $\underline{\mathsf{l\text{-}frame}}_l^P(\beta) \equiv \{[]_{\beta,0}\}_{\mathsf{pk}}^{\mathsf{n}}$ and:

$$\mathsf{leave\text{-}st}(\beta \downarrow_R) = \left\{ \{v\}_{\mathsf{pk}}^{\mathsf{n}} \mid v \in \mathsf{leave\text{-}st}(m \downarrow_R) \right\}$$

  Let $u \in \mathsf{leave\text{-}st}(\beta \downarrow_R)$, there exists $u_m \in \mathsf{leave\text{-}st}(m \downarrow_R)$ such that $u \equiv \{u_m\}_{\mathsf{pk}}^{\mathsf{n}}$. Let θ be the substitution mapping $[]_{\beta,0}$ to u_m. Then:

$$\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \equiv \{u_m\}_{\mathsf{pk}}^{\mathsf{n}} \equiv u \in \zeta_{\mathcal{K}_l^P}(u)$$

- If β is a decryption oracle call in $\mathcal{D}_l^P$ for $\mathsf{dec}(s, \mathsf{sk})$, the:

$$\mathsf{leave\text{-}st}(\beta \downarrow_R) \subseteq \{\mathsf{dec}(u_s, \mathsf{sk}) \mid u_s \in \mathsf{leave\text{-}st}(s \downarrow_R)\} \cup \{\mathbf{0}(\mathsf{dec}(u_s, \mathsf{sk})) \mid u_s \in \mathsf{leave\text{-}st}(s \downarrow_R)\}$$

  Let $u \in \mathsf{leave\text{-}st}(\beta \downarrow_R)$, there exists $u_s \in \mathsf{leave\text{-}st}(s \downarrow_R)$ such that $u \equiv \mathsf{dec}(u_s, \mathsf{sk})$ or $u \equiv \mathbf{0}(\mathsf{dec}(u_s, \mathsf{sk}))$. Since s is a $\mathcal{S}_l^P$-normalized basic term, by induction hypothesis we have θ such that $\underline{\mathsf{l\text{-}frame}}_l^P(s)\theta \in \zeta_{\mathcal{K}_l^P}(u_s)$. Moreover:

$$\underline{\mathsf{l\text{-}frame}}_l^P(\beta) \equiv \mathsf{dec}(\underline{\mathsf{l\text{-}frame}}_l^P(s)\mu, \mathsf{sk})$$

  where μ is a renaming of hole variables. Let $\theta' = \mu^{-1}\theta$, then:

$$\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta' \equiv \mathsf{dec}(\underline{\mathsf{l\text{-}frame}}_l^P(s)\mu\mu^{-1}\theta, \mathsf{sk}) \equiv \mathsf{dec}(\underline{\mathsf{l\text{-}frame}}_l^P(s)\theta, \mathsf{sk}) \in \zeta_{\mathcal{K}_l^P}(u)$$

- Otherwise, $\beta \equiv f(\beta_1, \ldots, \beta_n)$ where, for every $1 \le i \le n$, β_i is a $\mathcal{S}_l^P$-normalized basic term. Then, using the fact that β is a $\mathcal{S}_l^P$-normalized basic term, we check that:

$$\mathsf{leave\text{-}st}(\beta \downarrow_R) \subseteq \{f(v_1, \ldots, v_n) \mid \forall i, v_i \in \mathsf{leave\text{-}st}(\beta_i \downarrow_R)\}$$

  Let $u \in \mathsf{leave\text{-}st}(\beta \downarrow_R)$, there exists $v_1, \ldots, v_n$ such that for every $1 \le i \le n$ $v_i \in \mathsf{leave\text{-}st}(\beta_i \downarrow_R)$ and $u \equiv f(v_1, \ldots, v_n)$. By induction hypothesis, there exists $\theta_1, \ldots, \theta_n$ such that for every $1 \le i \le n$:

$$\underline{\mathsf{l\text{-}frame}}_l^P(\beta_i)\theta_i \in \zeta_{\mathcal{K}_l^P}(v_i)$$

  For very $1 \le i \le n$, let μ_i be the lifting substitution such that, for every $\alpha \in \mathcal{E}_l^P$ and position p, $\mu_i([]_{\alpha,p}) \equiv []_{\alpha, i \cdot p}$. Then:

$$\underline{\mathsf{l\text{-}frame}}_l^P(\beta) \equiv f(\underline{\mathsf{l\text{-}frame}}_l^P(\beta_1)\mu_1, \ldots, \underline{\mathsf{l\text{-}frame}}_l^P(\beta_n)\mu_n)$$

  Observe that the substitutions $(\mu_i\theta_i)_{1 \le i \le n}$ have disjoint domains. Let $\theta = \mu_1\theta_1 \ldots \mu_n\theta_n$. Then:

$$\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \equiv f(\underline{\mathsf{l\text{-}frame}}_l^P(\beta_1)\mu_1\theta_1, \ldots, \underline{\mathsf{l\text{-}frame}}_l^P(\beta_n)\mu_n\theta_n)$$

  We know that f cannot be the function symbol $\mathbf{0}(\_)$ (since FA cannot be applied on $\mathbf{0}(\_)$. It follows that:

$$f(\underline{\mathsf{l\text{-}frame}}_l^P(\beta_1)\mu_1\theta_1, \ldots, \underline{\mathsf{l\text{-}frame}}_l^P(\beta_n)\mu_n\theta_n) \in \zeta_{\mathcal{K}_l^P}(u)$$ ∎

We lift the previous result to α-bounded conditionals.

Lemma 19. *Let $P \vdash_{\alpha}^{\textit{npf}} t \sim t'$, l a branch label in $\textit{label}(P)$, $\textit{h}$ a proof index and $\beta \in (\le_{\textit{bt}}^{\textit{h},\textit{l}} (t, P) \cup \textit{cs-path}^{\textit{h},\textit{l}}(t, P))$. If β is (t, P)-α-bounded then there exists a substitution θ s.t. $\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \in \mathcal{B}(t, t')$.*

Proof. We prove this by induction on the well-founded order underlying the inductive definition of (t, P)-α-bounded terms.

- **Base case:** Assume $\mathsf{h} = \epsilon$ and $\mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \mathsf{st}(t \downarrow_R) \neq \emptyset$. Let $u \in \mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \mathsf{st}(t \downarrow_R)$, we have u in R-normal form and if-free, therefore $u \in \mathsf{st}(\mathsf{leave\text{-}st}(t \downarrow_R) \cup \mathsf{cond\text{-}st}(t \downarrow_R))$. Moreover, by Proposition 34, there exists θ such that $\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \in \zeta_{\mathcal{K}_l^P}(u)$. Hence $\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \in \mathcal{B}(t, t')$.
- **Base case:** Assume $\mathsf{h} = \epsilon$ and there exists β' such that:

$$(\beta, \beta')\ (\leq_{\mathsf{l}\sim\mathsf{l}}^{\epsilon,l} \cup \leq_{\mathsf{c}\sim\mathsf{c}}^{\epsilon,l} \cup \leq_{\mathsf{cs}\sim\mathsf{cs}}^{\epsilon})\ (t \sim t', P) \quad \text{and} \quad \mathsf{leave\text{-}st}(\beta' \downarrow_R) \cap \mathsf{st}(t' \downarrow_R) \neq \emptyset$$

  By Proposition 31 we know that $\mathsf{l\text{-}frame}_l^P(\beta) \equiv \mathsf{r\text{-}frame}_l^P(\beta')$. By Proposition 32, we deduce that $\underline{\mathsf{l\text{-}frame}}_l^P(\beta) \equiv \underline{\mathsf{r\text{-}frame}}_l^P(\beta')$. From the previous case we know that there exists θ such that $\underline{\mathsf{r\text{-}frame}}_l^P(\beta')\theta \in \mathcal{B}(t')$. Therefore $\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \in \mathcal{B}(t')$.
- **Inductive case, same label:** Assume $\beta \in \mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P)$ and that there exists $\varepsilon \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P)$ such that ε is (t, P)-α-bounded and $\beta \in \mathsf{leave\text{-}st}(\varepsilon \downarrow_R)$. By induction hypothesis we have θ such that $\underline{\mathsf{l\text{-}frame}}_l^P(\varepsilon)\theta \in \mathcal{B}(t, t')$. We know that β is if-free and in R-normal form and that ε is a $\mathcal{S}_l^P$-normalized basic term. Therefore, by Proposition 28, we have $\mathsf{l\text{-}frame}_l^P(\beta) \equiv \mathsf{l\text{-}frame}_l^P(\varepsilon)$. Hence, using Proposition 32, $\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \in \mathcal{B}(t, t')$.
- **Inductive case, different labels:** Similar to the previous case.
- **Inductive case, guard:** If there exists $\varepsilon \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P)$ such that:
  - $\varepsilon \equiv B[\vec{w}, (\alpha_i)_i, (\mathsf{dec}_j)_j]$ is (t, P)-α-bounded.
  - β is a guard of a $\mathcal{S}_l^P$-decryption oracle call $d \in (\mathsf{dec}_j)_j$.

  By induction hypothesis there exists θ such that $\underline{\mathsf{l\text{-}frame}}_l^P(\varepsilon)\theta \in \mathcal{B}(t, t')$. Moreover let $(\mathsf{pk}_i)_i$ and $(\mathsf{n}_i)_i$ be such that $\forall i, \alpha_i \equiv \{\_\}_{\mathsf{pk}_i}^{\mathsf{n}_i}$. Then:

$$\mathsf{l\text{-}frame}_l^P(\varepsilon) \equiv B\left[\vec{w}, \left(\{[]_{\alpha_i}\}_{\mathsf{pk}_i}^{\mathsf{n}_i}\right)_i, \left(\mathsf{l\text{-}frame}_l^P(\mathsf{dec}_j)\right)_j\right]$$

  Therefore there exists a renaming of hole variables μ such that $\underline{\mathsf{l\text{-}frame}}_l^P(d)\mu\theta \in \mathsf{st}(\underline{\mathsf{l\text{-}frame}}_l^P(\varepsilon)\theta)$. Since $\mathcal{B}(t, t')$ is closed under st, this implies that:

$$\underline{\mathsf{l\text{-}frame}}_l^P(d)\mu\theta \in \mathcal{B}(t, t')$$

  d is of the form $\mathsf{dec}(s, \mathsf{sk})$ where $\mathsf{sk} \in \mathcal{K}$. Since members of $\mathsf{guards}_{\mathcal{K}}(\_)$ are of the form $\mathsf{eq}(\_, \_)$, we know that there exists some $u \in \mathsf{st}(\mathsf{leave\text{-}st}(t \downarrow_R) \cup \mathsf{cond\text{-}st}(t \downarrow_R))$ such that $\underline{\mathsf{l\text{-}frame}}_l^P(d)\mu\theta \in \zeta_{\mathcal{K}}(u)$. Since β is a guard of d, β is of the form $\mathsf{eq}(s, \alpha)$ where α is an encryption under key pk (corresponding to sk) and randomness n appearing directly in s. It follows that:

$$\mathsf{l\text{-}frame}_l^P(d) \equiv \mathsf{dec}(\mathsf{l\text{-}frame}_l^P(s), \mathsf{sk}) \qquad \mathsf{l\text{-}frame}_l^P(\beta) \equiv \mathsf{eq}(\mathsf{l\text{-}frame}_l^P(s), \{[]_\alpha\}_{\mathsf{pk}}^{\mathsf{n}})$$

  Since α appears directly in s, and since $\underline{\mathsf{l\text{-}frame}}_l^P(d)\mu\theta \in \zeta_{\mathcal{K}}(u)$, there exists θ' such that:

$$\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta' \in \mathsf{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u)) \subseteq \mathcal{B}(t, t') \qquad \blacksquare$$

We now bound the size of $\mathcal{B}(t)$.

Proposition 35. *For every term t, for every $u \in \mathcal{B}(t)$, we have $|u| \leq |t \downarrow_R|$. Moreover:*

$$|\mathcal{B}(t)| \leq |t \downarrow_R|^2.2^{|t \downarrow_R|}$$

Proof. An over-approximation of the set of terms $\zeta_{\mathcal{K}}(u)$ is obtained from u by choosing a subset of positions of u where decryptions over keys in $\mathcal{K}$ occur, and removing $\mathbf{0}$ before the subterms at these positions (if there is one). Hence each element of $\zeta_{\mathcal{K}}(u)$ is of size at most $|u|$. Moreover, for every $u \in \mathsf{st}(\mathsf{leave\text{-}st}(t \downarrow_R) \cup \mathsf{cond\text{-}st}(t \downarrow_R))$, we have $u \in \mathsf{st}(t \downarrow_R)$, and therefore $|u| \leq |t \downarrow_R|$. Therefore the set $\zeta_{\mathcal{K}}(u)$ contains terms of size at most $|t \downarrow_R|$.

Let $\mathsf{dec}(s, \mathsf{sk}) \in \zeta_{\mathcal{K}}(u)$, then $|\mathsf{dec}(s, \mathsf{sk})| = |s| + 3$ and for every α appearing in s:

$$|\mathsf{eq}(s, \alpha)| = |s| + |\alpha| + 1 \leq 2|s| + 1 \leq 2|\mathsf{dec}(s, \mathsf{sk})| \leq 2|t \downarrow_R|$$

Hence the set $\mathsf{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u))$ contains terms of size at most $2|t \downarrow_R|$. We deduce that for every $v \in \mathcal{B}(t)$, $|v| \leq 2|t \downarrow_R|$. Moreover, by upper-bounding the positions of $\mathsf{dec}(s, \mathsf{sk})$ where an encryption might be, there are at most $|s| - 1 \leq |t \downarrow_R| - 1$ such α, independently of the set of keys $\mathcal{K}$. It follows that:

$$\left| \bigcup_{\mathcal{K} \subseteq \mathcal{S}_{\mathsf{k}}(t)} \mathsf{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u)) \right| \leq |\zeta_{\mathcal{K}}(u)|.(|t \downarrow_R| - 1)$$

Independently of the set of keys $\mathcal{K}$ chosen, we have at most $|\mathsf{st}(t\downarrow_R)| \leq |t\downarrow_R|$ choices for u, and the set $\bigcup_{\mathcal{K}\subseteq\mathcal{S}_{\mathsf{k}}(t)} \zeta_{\mathcal{K}}(u)$ contains at most $2^{|u|} \leq 2^{|t\downarrow_R|}$ elements (we choose the positions where we remove **0**s). Hence:

$$\begin{aligned}\Big|\bigcup_{\mathcal{K}\subseteq\mathcal{S}_{\mathsf{k}}(t)} \zeta_{\mathcal{K}}(u) \cup \mathsf{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u))\Big| &\leq \Big|\bigcup_{\mathcal{K}\subseteq\mathcal{S}_{\mathsf{k}}(t)} \zeta_{\mathcal{K}}(u)\Big| + \Big|\bigcup_{\mathcal{K}\subseteq\mathcal{S}_{\mathsf{k}}(t)} \mathsf{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u))\Big| \\ &\leq |\zeta_{\mathcal{K}}(u)| + (|t\downarrow_R| - 1).|\zeta_{\mathcal{K}}(u)| \leq |t\downarrow_R|.2^{|t\downarrow_R|}\end{aligned}$$

By consequence:

$$|\mathcal{B}(t)| \leq |t\downarrow_R|.|t\downarrow_R|.2^{|t\downarrow_R|} \leq |t\downarrow_R|^2.2^{|t\downarrow_R|}$$

■

Finally, we apply a pigeon-hole argument to bound the number of nested basic terms.

Lemma 20. *Let $P \vdash^{\mathsf{npf}}_{\alpha} t \sim t'$. Let l be a branch label in $\mathsf{label}(P)$, h a proof index. Let $(\beta_i)_{i\leq n}$ such that for all i, $\beta_i \leq^{\mathsf{h},l}_{\mathsf{bt}} (t,P)$. If $\beta_1 <_{\mathsf{st}} \cdots <_{\mathsf{st}} \beta_n$ then $n \leq |\mathcal{B}(t,t')|$.*

Proof. For every $i \neq j$, we know, using Proposition 29, that $\mathsf{l\text{-}frame}^P_l(\beta_i) \not\equiv \mathsf{l\text{-}frame}^P_l(\beta_j)$. By Proposition 32, we deduce that $\underline{\mathsf{l\text{-}frame}}^P_l(\beta_i) \not\equiv \underline{\mathsf{l\text{-}frame}}^P_l(\beta_j)$. Since $P \vdash^{\mathsf{npf}}_{\alpha} t \sim t'$, we know that for every i, β_i is (t,P)-α-bounded. Using Lemma 19, we deduce that for every i, there exists a substitution θ_i such that:

$$\underline{\mathsf{l\text{-}frame}}^P_l(\beta_i)\theta_i \in \mathcal{B}(t,t')$$

Using the contrapositive of Proposition 33, we have that for every $i \neq j$:

$$\underline{\mathsf{l\text{-}frame}}^P_l(\beta_i)\theta_i \not\equiv \underline{\mathsf{l\text{-}frame}}^P_l(\beta_j)\theta_j$$

Therefore, by a pigeon-hole argument, $n \leq |\mathcal{B}(t,t')|$. ■

C. Candidate Sequences

Let $P \vdash^{\mathsf{npf}}_{\alpha} t \sim t'$. For all $n \leq |\mathcal{B}(t,t')|$, we are going to define the set $\mathcal{U}_n$ of normalized basic terms that may appear in P using n nested basic terms. We then show that these sets are finite and recursive, and give an upper-bound on their size which does not depend on n. This allows us to conclude by showing that the existence of a proof using our (complete) strategy is decidable.

Definition 50. *An α-context C is a context such that all holes appear below the encryption function symbol, with proper randomness and encryption key. More precisely, for every position $p \in \mathsf{pos}(C)$, if $C_{|p} \equiv []$ then $p = p' \cdot 0$ and there exist two nonces n, n_r such that $C_{|p'} \equiv \{[]\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}$.*

Moreover, we require that every hole appears at most once.

Remark 11. For every $\beta \leq^{\mathsf{h},l}_{\mathsf{bt}} (t,P)$, the context $\mathsf{l\text{-}frame}^P_l(\beta)$ is an α-context.

Let t and t' be two ground terms. We now define what is a *valid candidate sequence* $(\mathcal{U}_n, \mathcal{A}_n)_{n\in\mathbb{N}}$ for t, t'. Basically, $\mathcal{U}_n$ corresponds to basic terms at nested depth n that could appear, on the left, in a proof of $\vdash^{\mathsf{npf}}_{\alpha} t \sim t'$, while $\mathcal{A}_n$ is the set of left encryptions oracle calls built using basic terms in $\mathcal{U}_{n-1}$.

Definition 51. *Let t, t' be two terms. A sequence of pairs of sets of ground terms $(\mathcal{U}_n, \mathcal{A}_n)_{n\in\mathbb{N}}$ is a* valid candidate sequence *for t, t' if:*

- *$\mathcal{U}_0 = \mathcal{B}(t,t')$ and $\mathcal{A}_0 = \emptyset$.*
- *For $n \geq 0$, $\mathcal{A}_{n+1}$ can be any set of terms that satisfies the following constraints (with the convention that $\mathcal{A}_{-1} = \emptyset$): $\mathcal{A}_{n+1}$ contains $\mathcal{A}_n$, and for all $\alpha \in \mathcal{A}_{n+1}\backslash\mathcal{A}_n$, $\alpha \equiv \{D[\vec{b} \diamond \vec{u}]\}^{\mathsf{n}_r}_{\mathsf{pk}(\mathsf{n}_p)}$ where:*
  - *$\vec{b} \cup \vec{u}$ are in $\mathcal{U}_{n-1}$ and there exists $\{\_\}^{\mathsf{n}_r}_{\_} \in \mathsf{st}(t\downarrow_R) \cup \mathsf{st}(t'\downarrow_R)$.*
  - *for every branch $\vec{\rho} \subseteq \vec{b}$ of $D[\vec{b} \diamond \vec{u}]$, $\vec{\rho}$ does not contain duplicates.*
  - *$\mathcal{A}_n$ does not contain any terms of the form $\{\_\}^{\mathsf{n}_r}_{\_}$.*
- *For $n > 0$, we let $\mathcal{U}_{n+1}$ is the set of term defined from $\mathcal{U}_n$ and $\mathcal{A}_n$ as follows: $\mathcal{U}_{n+1}$ contains $\mathcal{U}_n$, plus any element that can be obtained through the following construction:*
  - *Take a α-context C such that there exists θ with $C\theta \in \mathcal{B}(t,t')$.*
  - *Let $[]_1, \ldots, []_a$ be the variables of C, and let $\alpha_1, \ldots, \alpha_a$ be encryptions in $\mathcal{A}_n$. For all $1 \leq k \leq a$, let s_i be such that $\{s_i\}^{\_}_{\_} \equiv \alpha_i \in \mathcal{A}_n$.*
  - *Let $v_0 \equiv C[(s_i)_{1\leq i\leq a}]$. Then let v be the term obtained from v_0 as follows: take positions $p_1, \cdots, p_o \in \mathsf{pos}(C)$ such that for all $1 \leq i \leq o$, $C_{|p_i} \equiv \mathsf{dec}(\_, \mathsf{sk}_i)$ (where sk_i is a valid private key, i.e. of the form $\mathsf{sk}(\mathsf{n}_i)$); for every $1 \leq i \leq o$, replace in v_0 the subterm $\mathsf{dec}(s, \mathsf{sk})$ at position p by $D[\vec{g} \diamond \vec{w}]$, where $\vec{g}$ are terms in $\mathcal{U}_n$ of the form $\mathsf{eq}(s, \alpha)$ (with $\alpha \equiv \{\_\}^{\mathsf{n}_\alpha}_{\_} \in \mathcal{A}_n$ and α directly appears in s) and $\forall w \in \vec{w}$, $w \equiv \mathsf{dec}(s, \mathsf{sk})$ or $w \equiv \mathbf{0}(\mathsf{dec}(s, \mathsf{sk}))$.*

Proposition 36. *Let $P \vdash^{\mathsf{npf}}_{\alpha} t \sim t'$. For $l \in \mathsf{label}(P)$, there exists a valid candidate sequence $(\mathcal{U}_n, \mathcal{A}_n)_{n\in\mathbb{N}}$ for t, t' such that:*

$$\bigcup_{h} \leq^{h,l}_{bt} (t, P) \subseteq \bigcup_{n<|\mathcal{B}(t,t')|} \mathcal{U}_n \qquad \textit{and} \qquad \bigcup_{h} \textit{cs-path}^{h,l}(t, P) \subseteq \bigcup_{n<|\mathcal{B}(t,t')|} \text{leave-st}\,(\mathcal{U}_n \downarrow_R)$$

Proof. First, we show that there exists a valid candidate sequence such that the inclusion holds when taking the union over $\mathbb{N}$ on the right, and s.t. for every n, $\mathcal{A}_n$ contains only valid encryptions in $\mathcal{E}^P_l$, i.e.:

$$\mathcal{S} = \bigcup_{\mathsf{h}} \leq^{\mathsf{h},l}_{\mathsf{bt}} (t, P) \subseteq \bigcup_{n<+\infty} \mathcal{U}_n \qquad \text{and} \qquad \bigcup_{n\in\mathbb{N}} \mathcal{A}_n \subseteq \mathcal{E}^P_l \tag{26}$$

Before starting the construction of the valid candidate sequence, we make some observations: if one fixes $(\mathcal{A}_n)_{n\in\mathbb{N}}$, there is at most one sequence $(\mathcal{U}_n)_{n\in\mathbb{N}}$ such that $(\mathcal{U}_n, \mathcal{A}_n)_{n\in\mathbb{N}}$ is a valid candidate sequence.

Moreover this sequence is non-decreasing in $(\mathcal{A}_n)_{n\in\mathbb{N}}$. More precisely, if $(\mathcal{U}_n, \mathcal{A}_n)_{n\in\mathbb{N}}$ and $(\mathcal{U}'_n, \mathcal{A}'_n)_{n\in\mathbb{N}}$ are valid candidate sequences such that for every n, $\mathcal{A}_n \subseteq \mathcal{A}'_n$, then for every n, $\mathcal{U}_n \subseteq \mathcal{U}'_n$.

We now describe a procedure that recursively construct $\mathcal{S}' \subseteq \mathcal{S}$ and a valid candidate sequence $(\mathcal{U}_n, \mathcal{A}_n)_{n\in\mathbb{N}}$ such that $\mathcal{S}'$ is a subset of $\bigcup_{n\leq+\infty} \mathcal{U}_n$ (eventually, we will show that $\mathcal{S}' = \mathcal{S}$). Moreover we require $(\mathcal{A}_n)_{n\in\mathbb{N}}$ to be minimal in the following sense: if $\alpha \equiv C[\vec{b} \diamond \vec{u}]$ is in $\mathcal{A}_{n+1}\backslash\mathcal{A}_n$ then there exists $v \in \vec{b} \cup \vec{u}$ such that $v \in \mathcal{U}_n\backslash\mathcal{U}_{n-1}$ (in other words, we add new encryptions in $\mathcal{A}_n$ as soon as we can).

Initially we take $\mathcal{A}_n = \emptyset$ for every n, $(\mathcal{U}_n)_{n\in\mathbb{N}}$ such that $(\mathcal{U}_n, \mathcal{A}_n)_{n\in\mathbb{N}}$ is a valid candidate sequence and $\mathcal{S}' = \emptyset$. While $\mathcal{S}' \neq \mathcal{S}$, we pick an element β in $\mathcal{S}\backslash\mathcal{S}'$ such that β is minimal for $<_{\mathsf{st}}$ in $\mathcal{S}\backslash\mathcal{S}'$. Then we add β to $\mathcal{S}'$ and update $(\mathcal{A}_n)_{n\in\mathbb{N}}$ as follows:

a) Case 1: If β is minimal for $<_{\mathsf{st}}$ in $\mathcal{S}$, we have β of the form $B[\vec{w}, (\alpha_i)_{i\in I}, (\mathsf{dec}_j)_{j\in J}]$. By minimality of β, we have $I = \emptyset$ and for all $j \in J$, dec_j has no encryptions in $\mathcal{E}^P_l$, and by consequence no guards. It follows that β is if-free and in R-normal form, hence $\text{l-frame}^P_l(\beta) \equiv \beta$. By consequence, using Lemma 19, we get that $\beta \in \mathcal{B}(t, t') = \mathcal{U}_0$ (since $\mathcal{U}_0$ does not depends on the sets $(\mathcal{A}_n)_{n\in\mathbb{N}}$).

b) Case 2: Let β such that for all $\beta' <_{\mathsf{st}} \beta$, $\beta' \in \mathcal{S}'$. Since $\mathcal{S}' \subseteq \cup_{n\in\mathbb{N}}\mathcal{U}_n$, and since $\{\beta' \mid \beta' <_{\mathsf{st}} \beta\}$ is finite, there exists n_m such that:

$$\{\beta' \mid \beta' <_{\mathsf{st}} \beta\} \cap \left(\leq^{\mathsf{h},l}_{\mathsf{bt}} (t, P) \cup \text{cs-path}^{\mathsf{h},l}(t, P)\right) \subseteq \bigcup_{0\leq n\leq n_m} \mathcal{U}_n$$

From Lemma 19 we have a substitution θ such that:

$$\underline{\text{l-frame}}^P_l(\beta)\theta \in \mathcal{B}(t, t')$$

We then just need to show that we can obtain β from $\underline{\text{l-frame}}^P_l(\beta)$ using the procedure defining $\mathcal{U}_{n_m+1}$:

- For all encryption $\alpha \equiv \{m\}^{\mathsf{n}}_{\mathsf{pk}} \in \mathsf{st}(\beta)\cap\mathcal{E}^P_l$, we know that $m \equiv C[\vec{b}\diamond\vec{u}]$ where $\vec{b}, \vec{u} <_{\mathsf{st}} \beta$. Hence $\vec{b}, \vec{u}$ are in $\cup_{0\leq n\leq n_m}\mathcal{U}_n$. We then have two cases:
  - either $\cup_{n\in\mathbb{N}}\mathcal{A}_n$ already contains an encryption α' with randomness n. Since $\cup_{n\in\mathbb{N}}\mathcal{A}_n \subseteq \mathcal{E}^P_l$, and using the side-condition of the $\mathsf{CCA2}$ application, we know that $\alpha \equiv \alpha' \in \cup_{n\in\mathbb{N}}\mathcal{A}_n$. By minimality of the $(\mathcal{A}_n)_{n\in\mathbb{N}}$ we know that $\alpha \in \mathcal{A}_{n_m+1}$.
  - or $\cup_{n\in\mathbb{N}}\mathcal{A}_n$ does not contain an encryption with randomness n. Then we simply add α to $\mathcal{A}_{n'}$, where $n' \leq n_m + 1$ is the smallest possible: we know that there exists such a n' since adding α to $\mathcal{A}_n$ yields, after completion of the $(\mathcal{U}_n)_{n\in\mathbb{N}}$, a valid candidate sequence (one can check that for all branch $\vec{\rho}$ of $C[\vec{b} \diamond \vec{u}]$, $\vec{\rho}$ does not contain duplicates, using the third bullet point of the definition of $\vdash^{\mathsf{npf}}_{\alpha}$).

  Then we replace in $\underline{\text{l-frame}}^P_l(\beta)$ the holes $[]_{\alpha,\_}$ by $\{C[\vec{b} \diamond \vec{u}]\}^{\mathsf{n}}_{\mathsf{pk}}$. This produce a term v_0.
- Finally we also replace in v_0 every occurrence of $\mathsf{dec}(\_, \mathsf{sk})$ or $\mathbf{0}(\mathsf{dec}(\_, \mathsf{sk}))$ in $\mathsf{st}(\text{l-frame}^P_l(\beta))$ by the corresponding $\mathcal{S}^P_l$-decryption oracle call, which is possible since the guards $\vec{g}$ of this decryption oracle calls are such that $\vec{g} <_{\mathsf{st}} \beta$, hence are in $\cup_{0\leq n\leq n_m}\mathcal{U}_n$.

c) Conclusion: We show that when $\mathcal{S} = \mathcal{S}'$ we have:

$$\mathcal{S} \cap \bigcup_{n<+\infty} \mathcal{U}_n \;=\; \mathcal{S} \cap \bigcup_{n<|\mathcal{B}(t,t')|} \mathcal{U}_n \tag{27}$$

Assume that $\mathcal{S}\cap\mathcal{U}_{|\mathcal{B}(t,t')|-1} \subsetneq \mathcal{S}\cap\mathcal{U}_{|\mathcal{B}(t,t')|}$, take $\beta \in \mathcal{S}\cap(\mathcal{U}_{|\mathcal{B}(t,t')|}\backslash\mathcal{U}_{|\mathcal{B}(t,t')|-1})$. We know that $\beta \equiv B[\vec{w}, (\alpha_i)_i, (\mathsf{dec}_j)_j]$ and that there is an encryption α in $(\alpha_i)_i$ or in the encryptions of the $(\mathsf{dec}_j)_j$ such that $\alpha \in \mathcal{A}_{|\mathcal{B}(t,t')|-1}\backslash\mathcal{A}_{|\mathcal{B}(t,t')|-2}$ (otherwise β would be in $\mathcal{S}\cap\mathcal{U}_{|\mathcal{B}(t,t')|-1}$). Let $\alpha \equiv \{C[\vec{b} \diamond \vec{u}]\}^{\mathsf{n}}_{\mathsf{pk}}$, by minimality of the $(\mathcal{A}_n)_{n\in\mathbb{N}}$ we know that there is some $v \in \vec{b}\cup\vec{u}$ such that $v \in \mathcal{U}_{|\mathcal{B}(t,t')|-1}\backslash\mathcal{U}_{|\mathcal{B}(t,t')|-2}$. Since β is in $\mathcal{S}$ and since v is a $\mathcal{S}^P_l$-normalized basic term appearing in β we know that $v \in \mathcal{S}$. Let $\beta_0 \equiv \beta$, $\beta_1 \equiv v$, we have $v \in \mathcal{S}\cap(\mathcal{U}_{|\mathcal{B}(t,t')|-1}\backslash\mathcal{U}_{|\mathcal{B}(t,t')|-2})$. By induction we can build a sequence of terms

β_n, for $n \in \{0, \dots, |\mathcal{B}(t,t')|\}$ such that for all $0 \leq n \leq |\mathcal{B}(t,t')|$, $\beta_n \in \mathcal{S} \cap (\mathcal{U}_{|\mathcal{B}(t,t')|-i} \backslash \mathcal{U}_{|\mathcal{B}(t,t')|-(i+1)})$ and $\beta_{n+1} <_{\mathsf{st}} \beta_n$ (with the convention $\mathcal{U}_{-1} = \emptyset$). We built a sequence of terms in $\mathcal{S}$, strictly ordered by $<_{\mathsf{st}}$ and of length $|\mathcal{B}(t,t')| + 1$. This contradicts Lemma 20. Absurd.

To finish, it remains to show that:

$$\bigcup_{\mathsf{h}} \mathsf{cs\text{-}path}^{\mathsf{h},l}(t,P) \subseteq \bigcup_{n < |\mathcal{B}(t,t')|} \mathsf{leave\text{-}st}\left(\mathcal{U}_n \downarrow_R\right)$$

Let b in $\bigcup_{\mathsf{h}} \mathsf{cs\text{-}path}^{\mathsf{h},l}(t,P)$. Using Proposition 30 we know that there exists $\gamma \leq_{\mathsf{bt}}^{\mathsf{h}',l} (t,P)$ such that $b \in \mathsf{leave\text{-}st}(\gamma \downarrow_R)$. Since $\gamma \in \bigcup_{n < |\mathcal{B}(t,t')|} \mathcal{U}_n \downarrow_R$, we have $b \in \bigcup_{n<|\mathcal{B}(t,t')|} \mathsf{leave\text{-}st}\left(\mathcal{U}_n \downarrow_R\right)$. ∎

Proposition 37. *For all terms u, let $\mathcal{C}_u$ be the set of α-contexts:*

$$\mathcal{C}_u = \{C \mid \exists \theta.\, C\theta \equiv u \wedge \textit{every hole appears at most once}\}$$

and $\mathcal{C}_u^\alpha$ be $\mathcal{C}_u$ quotiented by the α-renaming of holes relation. Then $|\mathcal{C}_u^\alpha| \leq 2^{|u|}$.

Proof. The set of contexts $\mathcal{C}_u^\alpha$ can be injected in the subsets of positions of u as follows: for every context C, associate to C the set of positions of u such that $C_{|p}$ is a hole. This is invariant by α-renaming and uniquely characterizes C modulo hole renaming. It follows that there are less element of $\mathcal{C}_u^\alpha$ than subsets of $\mathsf{pos}(u)$, i.e. $2^{|\mathsf{pos}(u)|} = 2^{|u|}$. ∎

Proposition 38. *Let t and t' be two ground terms, $N = |t \downarrow_R| + |t' \downarrow_R|$. For every valid candidate sequence $(\mathcal{U}_n, \mathcal{A}_n)_{n \in \mathbb{N}}$ and $n \in \mathbb{N}$:*

$$|\mathcal{A}_n| \leq N \qquad\qquad |\mathcal{U}_n| \leq N^2.2^{3.N}$$

Proof. For every n, $\mathcal{A}_n$ contains only terms of the form $\alpha \equiv \{m\}_{\mathsf{pk}}^{\mathsf{n}_r}$, where $\{\_\}_{\_}^{\mathsf{n}_r} \in \mathsf{st}(t \downarrow_R) \cup \mathsf{st}(t' \downarrow_R)$. Moreover, $\mathcal{A}_n$ cannot contain two encryptions using the same randomness. Therefore $|\mathcal{A}_n| \leq N$.

For every n, the only leeway we have while constructing the terms in $\mathcal{U}_n$ is in the choice of the α-context C, as the content of the encryptions is determined by $\mathcal{A}_{n-1}$, and the guards that are added are determined by $\mathcal{U}_{n-1}$. The α-context C is picked in the following set:

$$\bigcup_{u \in \mathcal{B}(t,t')} \mathcal{C}_u^\alpha$$

which, using Proposition 35 and Proposition 37, we can bound by:

$$\Big| \bigcup_{u \in \mathcal{B}(t,t')} \mathcal{C}_u^\alpha \Big| \quad\leq\quad \sum_{u \in \mathcal{B}(t,t')} |\mathcal{C}_u^\alpha| \quad\leq\quad \sum_{u \in \mathcal{B}(t,t')} 2^{2.N} \quad\leq\quad N^2.2^N.2^{2.N} \quad=\quad N^2.2^{3.N}$$ ∎

Proposition 39. *Let t, t' be two ground terms and $N = |t \downarrow_R| + |t' \downarrow_R|$. For every valid candidate sequence $(\mathcal{U}_n, \mathcal{A}_n)_{n \in \mathbb{N}}$ and $n \in \mathbb{N}$:*

$$\forall u \in \bigcup_{n<|\mathcal{B}(t,t')|} \mathcal{U}_n,\ |u| \leq 2^{Q(N)\,.\,2^{4.N}}$$

Where $Q(X)$ is a polynomial of degree 4.

Proof. Even though there are at most $|\mathcal{B}(t,t')|.N^2.2^{3.N}$ distinct basic terms appearing in branch l at proof index h, these terms may be much larger. Let U_n (resp. A_n) be an upper bound on the size of a term in $\mathcal{U}_n$ (resp. $\mathcal{A}_n$). Then for every $0 \leq n < |\mathcal{B}(t,t')|$ and $\alpha \in \mathcal{A}_{n+1} \backslash \mathcal{A}_n$, α is of the form $\{C[\vec{b} \diamond \vec{u}]\}_{\mathsf{pk}}^{\mathsf{n}}$, where $\vec{b}, \vec{u}$ are in $\mathcal{U}_n$ and C is such that no term appears twice on the same branch. Recall that we call branch the ordered list of *inner conditionals*, which does not include the final leaf. If follows that C is of depth at most $|\mathcal{U}_n| + 1$, and therefore has at most $2^{|\mathcal{U}_n|+2} - 1$ conditional and leaf terms. To bound $|C[\vec{b} \diamond \vec{u}]|$, we need to bound the size of each of its internal and leaf terms, which we do using U_n. We get:

$$\left|C[\vec{b} \diamond \vec{u}]\right| \leq |C| + |C|\,.\,U_n \leq 2.|C|\,.\,U_n \leq 2^{|\mathcal{U}_n|+3}\,.\,U_n$$

since U_n is greater than 1 (terms can not be of size 0). Therefore $|\alpha| \leq 4 + 2^{|\mathcal{U}_n|+3}\,.\,U_n$. Using the bound from Proposition 38, we can take:

$$A_n = 4 + 2^{N^2.2^{3.N}+3}\,.\,U_n$$

Now let $u \equiv C[(\alpha_i)_{i \in I}, (\mathsf{dec}_j)_{j \in J}]$ in $\mathcal{U}_{n+1} \backslash \mathcal{U}_n$. We know that $\forall i \in I, |\alpha_i| \leq A_n$. There are at most $|C|$ hole occurrences in C, hence $|I| \leq |C|$ and $|J| \leq |C|$. To bound $|u|$, we also need to bound the size of the decryption guards. There are at most N guards for each decryption (since only element of $\mathcal{A}_n$ may be guarded, and $|\mathcal{A}_n| \leq N$), and each guard is in

$\mathcal{U}_n$, so of size bounded by U_n. Moreover, guarded decryptions have at most $N+1$ leaf, where each life is of size at most $|C[(\alpha_i)_{i\in I}, ([])_{j\in J}]| + 1 \leq |C| + |I|.A_n + 1$. Hence every decryption's size is upper-bounded by:

$$N + N.U_n + (N+1).(|C| + |I|.A_n + 1)$$

Finally $|C|$ is such that there there exists θ such that $C\theta \in \mathcal{B}(t,t')$, hence $|C| \leq 2.N$ using Proposition 35. Hence, assuming $U_n \geq N$ (which will be the case):

$$\begin{aligned} |C[(\alpha_i)_{i\in I}, (\mathsf{dec}_j)_{j\in J}]| &\leq |C| + |I|.A_n + |J|.(N + N.U_n + (N+1).(|C| + |I|.A_n + 1)) \\ &\leq 2N + 2N.A_n + 2N.(N + N.U_n + (N+1).(2N + 2N.A_n + 1)) \end{aligned}$$

Seen as a multi-variate polynomial in N, A_n and U_n, we have only monomials N, $N.A_n$, N^2, $N^2.U_n$, N^3 and $N^3.A_n$. Hence there exists a constant L such that:

$$u \leq L.N^3(A_n + U_n) \leq L.N^3(4 + 2^{N^2.2^{3.N}+3}.U_n + U_n)$$

Hence there exists some polynomial Q_0 of degree two such that $u \leq 2^{Q_0(N).2^{3N}}.U_n$. We let $U_0 = N$, and $U_{n+1} = 2^{Q_0(N).2^{3N}}.U_n$. Then:

$$U_{|\mathcal{B}(t,t')|-1} \leq 2^{|\mathcal{B}(t,t')|.Q_0(N).2^{3N}}.U_n \leq 2^{N^2.2^N.Q_0(N).2^{3N}}.U_n \leq 2^{N^2.Q_0(N).2^{4N}}.U_n$$

Hence we have a polynomial $Q(N) = N^2.Q_0(N)$, which is of degree four. ∎

Corollary 2. *Let $P \vdash^{\mathsf{npf}}_{\alpha} t \sim t'$ and $N = |\mathcal{B}(t,t')|$. For $l \in \mathsf{label}(P)$ and for all proof index h:*

$$\forall u \in \left(\leq^{\mathsf{h},l}_{\mathsf{bt}} (t,P) \cup \textit{cs-path}^{\mathsf{h},l}(t,P) \right),\ |u| \leq 2^{Q(N)\,.\,2^{4.N}}$$

Proof. Direct consequence of Proposition 36 and Proposition 39. ∎

To conclude, we only need to bound the number of nested $\mathsf{CS}_\square$ conditionals.

Proposition 40. *Let $P \vdash^{\mathsf{npf}}_{\alpha} t \sim t'$ and $(\mathsf{h}_i)_{1\leq i\leq n}$ be a sequence of indices of P such that for every $1 \leq i < n$, $\mathsf{h}_{i+1} \in \textit{cs-pos}_P(\mathsf{h}_i)$ and $\mathsf{h}_1 = \epsilon$. Then $n \leq |\mathcal{B}(t,t')| + 1$. Moreover $|\mathsf{label}(P)| \leq 2^{|\mathcal{B}(t,t')|}$.*

Proof. Let $l \in \mathsf{label}(P)$ be such that $\mathsf{h}_n \in \mathsf{h\text{-}branch}(l)$. The proof consists in building an increasing sequence of $\mathcal{S}^P_l$-normalized basic terms $\beta_1 <_{\mathsf{st}} \cdots <_{\mathsf{st}} \beta_m$ from $(\mathsf{h}_i)_{1\leq i\leq n}$ of length $m \geq n$. We then concludes using Lemma 20.

If $\mathsf{h}_n \neq \epsilon$, then h_n is of the form $h^n_{\mathsf{x}_n}$. We know that $\mathsf{extract}_{\mathsf{x}_n}(h^n, P)$ is a proof of $b^n \sim b'^n$ in $\mathcal{A}_{\mathsf{CS}_\square}$. Moreover $b^n \downarrow_R$ is in $\mathsf{cs\text{-}path}^{\mathsf{h}_{n-1},l}(t,P)$ and is (t,P)-α-bounded. Be definition of (t,P)-α-bounded terms, we know that there exists $(\beta_{n,j})_{1\leq j\leq k_n}$ (with $k_n \geq 1$) such that:

- for all $1 \leq j \leq k_n$, $\beta_{n,j} \leq^{\mathsf{h}_{n-1},l}_{\mathsf{bt}} (t,P)$.
- $b^n \downarrow_R \in \mathsf{leave\text{-}st}(\beta_{n,1} \downarrow_R)$.
- $\beta_{n,k_n} \leq^{\mathsf{h}_{n-1},l}_{\mathsf{l}} (t,P)$.
- for all $1 \leq j < k_n$, $\beta_{n,j}$ is a guard of a decryption in $\beta_{n,j+1}$, and therefore $\beta_{n,j} <_{\mathsf{st}} \beta_{n,j+1}$.

If $\mathsf{h}_{n-1} \neq \epsilon$, then since $\beta_{n,k_n} \leq^{\mathsf{h}_{n-1},l}_{\mathsf{l}} (t,P)$ is (t,P)-α-bounded, and since for any $\beta \leq^{\mathsf{h}_{n-1},l}_{\mathsf{bt}} (t,P)$, $\beta_{n,j}$ is not a guard of β, we know that we are in the inductive case with different labels of the definition of (t,P)-α-bounded terms. Therefore there exists $b^{n-1} \in \mathsf{cs\text{-}path}^{\mathsf{h}_{n-2},l}(t,P)$ such that $b^{n-1} \in \mathsf{leave\text{-}st}(\beta_{n,k_n})$.

We then iterate this process until we reach ϵ, building sequences $(\beta_{i,j})_{1<i\leq n, 1\leq j\leq k_i}$ and $(b^i)_{1<i\leq n}$. Since for all i, $b^{i-1} \in \mathsf{leave\text{-}st}(\beta_{i,k_i} \downarrow_R)$ and $b^{i-1} \in \mathsf{leave\text{-}st}(\beta_{i-1,1} \downarrow_R)$ we know, using Proposition 16, that $\beta_{i,k_i} \equiv \beta_{i-1,1}$. Therefore we have:

$$\beta_{n,1} <_{\mathsf{st}} \cdots <_{\mathsf{st}} \beta_{n,k_n} \equiv \beta_{n-1,1} <_{\mathsf{st}} \cdots <_{\mathsf{st}} \beta_{n-1,k_{n-1}} \cdots <_{\mathsf{st}} \beta_{3,k_3} \equiv \beta_{2,1} <_{\mathsf{st}} \cdots <_{\mathsf{st}} \beta_{2,k_2}$$

Moreover, for all i we have $k_i \geq 1$, therefore we built an increasing sequence of $\mathcal{S}^P_l$-normalized basic terms of length at least $n-1$. It follows, using Lemma 20, that $n - 1 \leq |\mathcal{B}(t,t')|$.

To upper-bound $|\mathsf{label}(P)|$, we only need to observe that we cannot have two $\mathsf{CS}_\square$ applications on the same conditional in a given branch. Consider the binary tree associated to the $\mathsf{CS}_\square$ applications in P, labelled by the corresponding $\mathsf{CS}_\square$ conditionals (say, on the left). Then this tree is of depth at most $|\mathcal{B}(t,t')|$, and therefore has at most $2^{|\mathcal{B}(t,t')|}$ leaves. ∎

Theorem (Main Result)**.** *The following problem is decidable:*
Input: *A ground formula $\vec{u} \sim \vec{v}$.*
Question: *Is $\mathsf{Ax} \wedge \vec{u} \not\sim \vec{v}$ unsatisfiable?*

Proof. Let $\vec{u} = u_1, \ldots, u_n$, $\vec{v} = v_1, \ldots, v_n$ and:

$$t \equiv \langle u_1\,,\, \langle \ldots\,,\, \langle u_{n-1}\,,\, u_n \rangle\rangle\rangle \qquad\qquad t' \equiv \langle v_1\,,\, \langle \ldots\,,\, \langle v_{n-1}\,,\, v_n \rangle\rangle\rangle$$

Using the $\mathsf{FA}_{\langle\_,\_\rangle}$ axiom, we know that if $\vec{u} \sim \vec{v}$ is derivable then $t \sim t'$ is derivable. Conversely, we show that $t \sim t'$ is derivable then $\vec{u} \sim \vec{v}$ is derivable. For every $3 \leq i \leq n$, let $\rho_i[]$ be the i-th projection defined using π_1 and π_2 by:

$$\forall n > i \geq 1, \rho_i \equiv \pi_1(\pi_2^{i-1}([])) \qquad\qquad \rho_n[] \equiv \pi_2^{n-1}([])$$

Then:

$$\dfrac{\dfrac{t \sim t'}{(\rho_i[t])_{1 \leq i \leq n} \sim (\rho_i[t'])_{1 \leq i \leq n}}\ \mathsf{FA}^*}{\vec{u} \sim \vec{v}}\ R$$

Hence $t \sim t'$ is derivable iff $\vec{u} \sim \vec{v}$ is derivable. Moreover, the corresponding proof of $\vec{u} \sim \vec{v}$ is of polynomial size in the size of the proof of $t \sim t'$. Therefore w.l.o.g. we can focus on the case $|\vec{u}| = |\vec{v}| = 1$.

Let $N = |\mathsf{st}(t\downarrow_R)| + |\mathsf{st}(t'\downarrow_R)|$. Using Proposition 40, we have bounded the number of branches of the proof tree (by $2^{N^2.2^N}$), and the number of nested $\mathsf{CS}_\square$ conditionals. For every branch, we non-deterministically guesses a set of α-bounded basic terms that can appear in a proof P of $P \vdash^{\mathsf{npf}}_{\alpha} t \sim t'$ using the valid candidate sequence algorithm (in polynomial time in $\mathcal{O}(N.2^{3.N}.2^{Q(N).2^{4.N}})$, using Proposition 38 and Proposition 39). Then the procedure guesses the rule applications, and checks that the candidate derivation is a valid proof. This is done in polynomial time in the size of the candidate derivation. Remark that to check whether the leaves are valid $\mathsf{CCA2}$ instances we use the polynomial-time algorithm describe in Proposition 9. Finally, since $|t\downarrow_R|$ is at most exponential with respect to $|t|$, this yields a 3-NEXPTIME decision procedure that shows the decidability of our problem. ∎