Decidability of a Sound Set of Inference Rules for Computational Indistinguishability

ADRIEN KOUTSOS, INRIA Paris, France

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 [7, 8], 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.

CCS Concepts: • **Security and privacy → Logic and verification**; • **Theory of computation → Automated reasoning**.

Additional Key Words and Phrases: Security Protocols, Automated Deduction, Decision Procedure, Computational Indistinguishability

ACM Reference Format:
Adrien Koutsos. 2020. Decidability of a Sound Set of Inference Rules for Computational Indistinguishability. *ACM Trans. Comput. Logic* 1, 1, Article 1 (July 2020), 113 pages. https://doi.org/1

1 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 [2] or the TripleHandshake attack [15]. 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 [16], Tamarin [33] and Deepsec [21].

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: he can perform any probabilistic polynomial time computation. More formally, messages

Author's address: Adrien Koutsos, adrien.koutsos@inria.fr, INRIA Paris, Paris, France.

1529-3785/2020/7-ART1 $15.00
https://doi.org/1

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 [11] which relies on pRHL, and CryptoVerif [17] 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 [7, 8]. 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.

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 used; 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, [25] shows decidability of secrecy (a reachability property) for a bounded number of sessions. In [27], the authors show the decidability of a secrecy property for *depth-bounded* protocols, with an unbounded number of sessions, using Well-Structured Transition Systems [28]. Chrétien et al [22] 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 [20] 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 [24], 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. [1]). 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. [3] for reachability properties, or [4] for indistinguishability properties), we obtain a decidability result in the computational model.

Contributions. In this paper, we consider the Bana-Comon model for indistinguishability properties [8]. 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 [13]. Given a protocol and a security property, we can build, using a folding technique described in [8], 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.

Formally, our main result is the decidability of the problem:

Input: A ground formula ψ of the form $\vec{u} \sim \vec{v}$.
Question: Is $\mathsf{Ax} \wedge \neg\psi$ unsatisfiable?

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

All the formulas in Ax are Horn clauses, therefore to show the unsatisfiability of $\mathsf{Ax} \wedge \neg\psi$ we use resolution with a negative strategy: we see axioms in Ax as inference rules and look for a derivation of the goal ψ. 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[1] 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.

Game Transformations. Our result can be reinterpreted as the decidability of the problem of determining whether there exits a sequence of game transformations [14, 35] that allows to prove the security of a protocol with a bounded number of sessions. Indeed, one can associate to every axiom in Ax either a cryptographic assumption or a game transformation.

Each atomic axiom in Ax corresponds to an instantiation of the IND-CCA$_2$ game. For instance, in the simpler case of IND-CPA security of an encryption $\{\_\}^{\mathsf{n}}_{\mathsf{pk}}$, no polynomial-time adversary can distinguish between two cipher-texts, even if it chooses the two corresponding plain-texts. 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 atomic axiom:

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

[1]Meaning that any ground formula derivable using the Ax is derivable using the ordered strategy.

Similarly, non-atomic axioms correspond to cryptographic game transformations. E.g., the FA axiom:

$$\frac{\vec{u} \sim \vec{v}}{f(\vec{u}) \sim f(\vec{v})}\ \mathsf{FA}$$

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 ψ (which is of the form $\vec{u} \sim \vec{v}$) as the initial game, our result can be reformulated as showing the decidability of the following problem:

Input: An initial bounded 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.

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 CCA_2 axiom schema, by allowing instances of the CCA_2 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.

Applications. The Bana-Comon indistinguishability model has been used to analyse RFID protocols in [23], a variant of the AKA protocol in [31], a key-wrapping API [34] and an e-voting protocol [5]. 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.

Related Works. In [9], 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 [9] do not provide a 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.

In [30], the authors study proof automation in the UC framework [18]. 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 [10], 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, even though the proof system of [10] shares some rules with ours, e.g. the R, Dup and FA rules, 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 stem from the if_then_else_. Moreover, there are no rules corresponding to cryptographic assumptions, like our CCA_2 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 [10].

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

Outline. We recall the Bana-Comon logic in Section 2. In Section 3, we introduce the axioms we use, which include, in particular, the equality axioms R and the cryptographic axioms CCA_2. In Section 4, we prove that the set of equality axioms R can be defined using a convergent term rewriting system $\rightarrow_R$. We state the main result in Section 5, and depict the difficulties of the proof. We prove several rule commutations in Section 6, which allow use to obtain complete ordered strategy for our fragment. We study the shape of the terms appearing in derivations following this ordered strategy in Section 7, and prove some key properties of these terms. In Section 8, we give some proof cut eliminations, and describe the decision procedure. Finally, we conclude in Section 9. For space reasons, the majority of the technical definitions and proofs are omitted in the body of this paper, and given in an Electronic Appendix.

2 THE LOGIC

We recall here the logic introduced in [8]. 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 predicates $\{\sim_n |\ n \in \mathbb{N}\}$, which stand for the indistinguishability of two vectors of n terms. 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* violate. This yields a set of axioms Ax. We show that $\mathsf{Ax} \wedge \vec{u} \nsim \vec{v}$ 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.

2.1 Syntax

Terms. Terms are built upon a set of function symbols $\mathcal{F}$, a countable set of names $\mathcal{N}$[2] and a countable set of variables $\mathcal{X}$. This is a sorted logic with two sorts term and bool, where bool $\subseteq$ term.

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 pk(_), sk(_), encryption with random seed $\{\_\}^{\_}_{\_}$, decryption dec(_, _), if_then_else_, true, false, zero $\mathbf{0}$(_) and equality check eq(_, _). We give their types below:

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

Moreover all the names in $\mathcal{N}$ have sort term, and each variable in $\mathcal{X}$ comes with a sort. 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 $\mathcal{S}$.

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.

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 [8]):

- the domain term is interpreted as the set of probabilistic polynomial time Turing machines equipped with a working tape and two random tapes ρ_1, ρ_2: ρ_1 is for the protocol random values, while ρ_2 is for the adversary random samplings. Moreover its input is of length η, where η is the security parameter. bool is the restriction of term 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. This ensures that distinct names are interpreted as independent random variables.
- true, false, $\mathbf{0}$(_), eq(_, _), and if_then_else_ are interpreted as expected. For instance, 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 $(m_i)_{i \le n}$ in the interpretation domains, and for every inputs w, ρ_1, ρ_2:

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

  Observe that M_g cannot access directly the tape ρ_1.
- Protocol function symbols (i.e. $\mathcal{F} \backslash \mathcal{G}$) are interpreted as deterministic polynomial-time Turing machines. Their interpretations will be restricted using *implementation axioms* later.

[2] A name is a constant function symbols used to model random samplings.

- 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 $m_1, \ldots, m_n \approx m'_1, \ldots, m'_n$ iff for every PPTM $\mathcal{A}$ with random tape ρ_2:

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

  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. ◇

3 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 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 [19]). 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).

3.1 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. Before presenting the axioms, we define some subsets of the set of function symbols $\mathcal{F}$:

Definition 1. We let $\mathcal{F}_{\setminus \mathbf{0}}$, $\mathcal{F}_{\setminus \mathsf{if}}$ and $\mathcal{F}_{\setminus \mathsf{if},\mathbf{0}}$ be the subsets of $\mathcal{F}$ defined by:

$$\mathcal{F}_{\setminus \mathbf{0}} = \mathcal{F} \setminus \{\mathbf{0}(\_)\} \qquad \mathcal{F}_{\setminus \mathsf{if}} = \mathcal{F} \setminus \{\mathsf{if\_then\_else\_}\} \qquad \mathcal{F}_{\setminus \mathsf{if},\mathbf{0}} = \mathcal{F} \setminus \{\mathbf{0}(\_), \mathsf{if\_then\_else\_}\}$$

The equality and structural axioms we present below already appeared in the literature [6, 8, 23], sometimes with slightly different formulations.

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:

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

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

$$R_2 \begin{cases} f(\vec{u}, \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ y, \vec{v}) = \mathsf{if}\ b\ \mathsf{then}\ f(\vec{u}, x, \vec{v})\ \mathsf{else}\ f(\vec{u}, y, \vec{v}) & (f \in \mathcal{F}_{\setminus \mathsf{if}}) \\ \mathsf{if}\ (\mathsf{if}\ b\ \mathsf{then}\ a\ \mathsf{else}\ c)\ \mathsf{then}\ x\ \mathsf{else}\ y = \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ (\mathsf{if}\ c\ \mathsf{then}\ x\ \mathsf{else}\ y) \end{cases}$$

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

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

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

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 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 Figure 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 if_then_else_; and R_4 allows to change the order in which condition tests are performed.

We then introduce a recursive set of rules:

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

It turns out that there exists a convergent orientation $\rightarrow_R$ of $=_R$.[3] We describe how we orient equalities of R, and prove that the resulting term rewriting system is convergent, later, in Section 4. Still, we anticipate and give the outlines of the orientation now.

We let $R_{\leq 3}$ be $R_1 \cup R_2 \cup R_3$. By orienting $R_{\leq 3}$ from left to right, and carefully choosing an orientation for the ground instances of R_4, we can build a recursive term rewriting system $\rightarrow_R$ convergent on ground terms:

- First, we choose the orientation of the rules in R_4. This is done by using a Lexicographic Path Ordering [26] on the conditions, modified using a user-chosen total order $\succ_u$ on ground $R_{\leq 3}$-irreducible conditions that do not use the if_then_else_ function symbol.

[3] Actually, there are many such orientations, as we will see later.

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

$$\frac{\vec{u}_1, \vec{v}_1 \sim \vec{u}_2, \vec{v}_2}{f(\vec{u}_1), \vec{v}_1 \sim f(\vec{u}_2), \vec{v}_2}\ \text{FA} \quad \text{where } f \in \mathcal{F}_{\setminus \mathbf{0}} \qquad \frac{\vec{u}, t \sim \vec{v}, t'}{\vec{u}, t, t \sim \vec{v}, t', t'}\ \text{Dup} \qquad \frac{\vec{v} \sim \vec{u}}{\vec{u} \sim \vec{v}}\ \text{Sym}$$

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

Conventions: π is a permutation of $\{1, \ldots, n\}$.

Fig. 2. The Axioms Struct-Ax.

- Then, we show that the resulting term rewriting system is locally confluent and terminating *on ground terms*. We deduce that it is convergent using Newman's lemma.

Theorem 1. *There exists an orientation $\rightarrow_{R_4}$ of R_4 on ground terms such that the resulting term rewriting system $\rightarrow_R = \rightarrow_{R_{\leq 3}} \cup \rightarrow_{R_4}$ is convergent on ground terms.*

Proof. The proof is given in Section 4. □

Structural Axioms. We now describe the set of structural axioms Struct-Ax, which is given in Figure 2. We focus on the case study axiom CS, which is the most complicated one. It states that in order to show that:

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

it is sufficient to show that the then branches and the else branches are indistinguishable, *when giving to the adversary the value of the condition* (i.e. b on the left and b' on the right). We can do better, by considering simultaneously several terms starting with the same condition. We also allow some terms $\vec{w}$ and $\vec{w}'$ on the left and right to stay untouched. This yield the axiom:

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

This is the only axiom with more than one premise. To get decidability, we had to restrict this rule, by only considering instances where the conditions b and b' are if-free (i.e. in $\mathcal{T}(\mathcal{F}_{\setminus \text{if}}, \mathcal{N})$). This restriction is used in the decidability proof, but might be unnecessary: decidability or undecidability with the unrestricted rule is open.

Definition 2. A term t is if-free if it does not use the if_then_else_, i.e. $t \in \mathcal{T}(\mathcal{F}_{\setminus \text{if}}, \mathcal{N})$.

We let CS be the rule:

$$\frac{\vec{w}, b, (u_i)_i \sim \vec{w}', b', (u'_i)_i \qquad \vec{w}, b, (v_i)_i \sim \vec{w}', b', (v'_i)_i}{\vec{w}, (\text{if } b \text{ then } u_i \text{ else } v_i)_i \sim \vec{w}', (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_i}\ \text{CS} \quad \text{when } b, b' \in \mathcal{T}(\mathcal{F}_{\setminus \text{if}}, \mathcal{N}),$$

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, stating that to prove that $\vec{u} \sim \vec{v}$, it is sufficient to show the stronger formula $\vec{u}, t \sim \vec{v}, t'$; 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 Figure 2 are valid in any computational model in which the functional correctness assumptions R_1 on pairs and encryptions hold.*

Proof. The proof is straightforward, and can be found in [8]. □

Note that the validity of the axioms in Figure 1 follows from the validity of the R rule of Figure 2 (except for the R_1 axioms, which do not hold in any computational model).

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 [8]: in the CS axiom, we forbid the terms b and b' from containing conditions; 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.

3.2 Cryptographic Assumptions

We now show how cryptographic assumptions are translated into atomic 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 the IND-CCA$_2$ game (Indistinguishability against Chosen Ciphertexts Attacks [13]). First, the challenger computes a public/private key pair (pk(n), sk(n)) (using a uniformly sampled nonce n of length η), and sends pk(n) to the attacker. The adversary then has access to two oracles:

- A left-right oracle $O^b_{\mathsf{LR}}(\mathsf{n})$ that takes two messages m_0, m_1 as input and returns $\{m_b\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}$, where b is an internal bit uniformly sampled at the beginning by the challenger and n_r is a fresh nonce.
- A decryption oracle $O_{\mathsf{dec}}(\mathsf{n})$ that, given m, returns $\mathsf{dec}(m, \mathsf{sk(n)})$ if m was not the result of a previous O_{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 B). In this game, the adversary $\mathcal{A}$ tries to guess the bit b, and the advantage $\mathrm{Adv}^{\mathrm{CCA}_2}_{\mathcal{A}}(\eta)$ of $\mathcal{A}$ against this game is the quantity:

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

An encryption scheme is IND-CCA$_2$ if the advantage $\mathrm{Adv}^{\mathrm{CCA}_2}_{\mathcal{A}}(\eta)$ 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 O_{dec} after having called O_{LR}. An encryption scheme is IND-CCA$_1$ if $\mathrm{Adv}^{\mathrm{CCA}_1}_{\mathcal{A}}(\eta)$ is negligible for any adversary $\mathcal{A}$.

CCA$_1$ Axiom. Before introducing the CCA$_2$ axioms, we recall informally the CCA$_1$ axioms from [8]. First, we define a syntactic property on secret keys used as a side-condition of the CCA$_1$ axioms:

Definition 3. For every ground term t, we say that a secret key sk(n) appears only in *decryption position* in t if it appears only in subterms of t of the form dec(_, sk(n)).

We now define the CCA$_1$ axioms:

Definition 4. CCA$_1$ is the recursive set of atomic axioms:

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

where $t, u, v, \vec{w}$ are ground, and:

- n_r does not appear in $t, u, v, \vec{w}$.
- n appears only in pk(n) or sk(n) in $t, u, v, \vec{w}$.
- sk(n) does not appear in $t, \vec{w}$, and sk(n) appears only in decryption position in u, v.
- the terms u and v are always of the same length (see Remark 1 below).

Proposition 2. *CCA_1 is valid in every computational model where the encryption scheme interpretation is IND-CCA_1.*

Proof. (*sketch*) For simplicity, we assume that $\vec{w}$ is empty. The proof is a reduction that, given a PPTM $\mathcal{A}$ that can distinguish between $t[\{u\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}]$ and $t[\{v\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}]$, 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 $[\![\{u\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}]\!]$ or $[\![\{v\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}]\!]$. 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. □

Remark 1. In the CCA_1 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).[4] To simplify the presentation, we omit all length constraints for now. We describe them later, in Appendix B.2. ⋄

CCA_2 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 5. Let $\alpha \equiv \{u\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}$ and $\alpha' \equiv \{v\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}$, then $\text{CCA}^{\mathsf{s}}_s$ is the recursive set of atomic axioms:

$$\vec{w}, t[\alpha], \text{if } \mathsf{eq}(t[\alpha], \alpha) \text{ then } \mathbf{0}(\mathsf{dec}(t[\alpha], \mathsf{sk(n)})) \quad \sim \quad \vec{w}, t[\alpha'], \text{if } \mathsf{eq}(t[\alpha'], \alpha') \text{ then } \mathbf{0}(\mathsf{dec}(t[\alpha'], \mathsf{sk(n)}))$$
$$\text{else } \mathsf{dec}(t[\alpha], \mathsf{sk(n)}) \qquad\qquad \text{else } \mathsf{dec}(t[\alpha'], \mathsf{sk(n)})$$

under the side-conditions of Definition 4, i.e. $t, u, v, \vec{w}$ are ground, and:

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

This axiom is valid whenever the encryption is IND-CCA_2.

Proposition 3. *CCA^{s}_s is valid in every computational model where the encryption scheme interpretation is IND-CCA_2.*

We do not prove validity of these axioms yet, as we actually use an extended version CCA_2 of $\text{CCA}^{\mathsf{s}}_s$ (given in Appendix B) where:

[4]Basically, we add to the CCA_1 axioms a premise ensuring that the lengths are the same, which we let the user under-axiomatize (in a restricted manner) through a length function Length(_) and a length equality predicate EQL(_, _).

- We allow for any number of calls to the left-right oracle, by adding a guard for each call. We use extra syntactic side-conditions to remove superfluous guards.
- The CCA_2 axiom schema is closed under alpha-renaming of names in $\mathcal{N}$.
- We restrict t to be without if_then_else_ and $\mathbf{0}$(_). This is needed in the completeness proof.
- Finally, the axioms allow for an arbitrary number of public/private key pairs to be used simultaneously (e.g., with keys $\mathsf{pk}(\mathsf{n})$ and $\mathsf{pk}(\mathsf{n}')$, we have the instance $\{A\}^{\mathsf{n}_0}_{\mathsf{pk}(\mathsf{n})}, \{B\}^{\mathsf{n}_1}_{\mathsf{pk}(\mathsf{n}')} \sim \{C\}^{\mathsf{n}_0}_{\mathsf{pk}(\mathsf{n})}, \{D\}^{\mathsf{n}_1}_{\mathsf{pk}(\mathsf{n}')}$), and an instance of the axiom can contain any number of interleaved left-right and decryption oracles calls. [5]

3.3 Transitivity

The last point is what allows us to eliminate transitivity in many examples. E.g, consider four encryptions, two of them (α and γ) using the public key $\mathsf{pk}(\mathsf{n})$, and the other two (β and δ) using the public key $\mathsf{pk}(\mathsf{n}')$:

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

Then the following formula is a valid instance of the CCA_2 axioms on, simultaneously, $\mathsf{pk}(\mathsf{n})$ and $\mathsf{pk}(\mathsf{n}')$:

$$\frac{}{\alpha, \beta \sim \gamma, \delta}\ \text{CCA}_2(\mathsf{pk}(\mathsf{n}), \mathsf{pk}(\mathsf{n}'))$$

However, proving the above formula using CCA_2 only on one key at a time, as in [8], requires transitivity:

$$\frac{\dfrac{}{\alpha, \beta \sim \alpha, \delta}\ \text{CCA}_2(\mathsf{pk}(\mathsf{n}')) \qquad \dfrac{}{\alpha, \delta \sim \gamma, \delta}\ \text{CCA}_2(\mathsf{pk}(\mathsf{n}))}{\alpha, \beta \sim \gamma, \delta}$$

When Transitivity is Needed. But, not surprisingly, transitivity can be necessary to complete a proof. Notably, this is the case when there are key-usability issues. For example, consider the toy scenario were an agent sends a message, say 0, encrypted with a key $\mathsf{pk}(\mathsf{n})$, and then encrypts the corresponding secret key $\mathsf{sk}(\mathsf{n})$, with some long-term key $\mathsf{pk}(\mathsf{n}_{\mathsf{lt}})$. To prove the secrecy of the message sent, consider to goal:

$$\{0\}^{\mathsf{n}^1_{\mathsf{e}}}_{\mathsf{pk}(\mathsf{n})}, \{\mathsf{sk}(\mathsf{n})\}^{\mathsf{n}^2_{\mathsf{e}}}_{\mathsf{pk}(\mathsf{n}_{\mathsf{lt}})} \sim \{1\}^{\mathsf{n}^1_{\mathsf{e}}}_{\mathsf{pk}(\mathsf{n})}, \{\mathsf{sk}(\mathsf{n})\}^{\mathsf{n}^2_{\mathsf{e}}}_{\mathsf{pk}(\mathsf{n}_{\mathsf{lt}})}$$

where we changed the message encrypted from 0 to 1. We cannot apply CCA_2 on $\mathsf{pk}(\mathsf{n})$ because the secret key appears outside a decryption position. And we cannot apply CCA_2 only on the key $\mathsf{pk}(\mathsf{n}_{\mathsf{lt}})$, because the first encryption is different on the left and the right (0 vs 1). There seems to be no proof using axioms Ax. But there is a proof if we add transitivity (given in Figure 3), by replacing the encrypted key $\mathsf{sk}(\mathsf{n})$ by $\mathsf{sk}(\mathsf{n}')$ (where n' is a fresh name) using CCA_2 on $\mathsf{pk}(\mathsf{n}_{\mathsf{lt}})$, and then applying CCA_2 on key $\mathsf{pk}(\mathsf{n})$.

3.4 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.

Still, our axioms are expressive enough to complete concrete proofs of security. We illustrate this on three examples: a proof of the simple formula from Example 1, a proof of a more complex

[5] Axioms for the IND-CCA_2 cryptographic assumption have already appeared in the literature, in [6]. These axioms are only for a single call to the left-right oracle, and a single key, while our axiom schema is more general. However, note that a more general axiom is not needed if you have transitivity (as in [6]), as we explain in Section 3.3.

$$\{0\}^{n_e^1}_{\mathsf{pk}(n)}, \{\mathsf{sk}(n)\}^{n_e^2}_{\mathsf{pk}(n_{lt})} \quad \overset{\mathrm{CCA}_2(\mathsf{pk}(n_{lt}))}{\sim} \quad \{0\}^{n_e^1}_{\mathsf{pk}(n)}, \{\mathsf{sk}(n')\}^{n_e^2}_{\mathsf{pk}(n_{lt})}$$

$$\not\sim \quad \mathrm{CCA}_2(\mathsf{pk}(n))$$

$$\{1\}^{n_e^1}_{\mathsf{pk}(n)}, \{\mathsf{sk}(n)\}^{n_e^2}_{\mathsf{pk}(n_{lt})} \quad \overset{\mathrm{CCA}_2(\mathsf{pk}(n_{lt}))}{\sim} \quad \{1\}^{n_e^1}_{\mathsf{pk}(n)}, \{\mathsf{sk}(n')\}^{n_e^2}_{\mathsf{pk}(n_{lt})}$$

Fig. 3. Derivation using the transitivity rule.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{0\}^{n_A}_{\mathsf{pk}_A}, \mathsf{pk}_B, n_B, \mathsf{sk}_B, n \sim \{1\}^{n_A}_{\mathsf{pk}_A}, \mathsf{pk}_B, n_B, \mathsf{sk}_B, n}\,\mathrm{CCA}_2(\mathsf{pk}_A, \mathsf{sk}_A)}{\{\{0\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}, \mathsf{sk}_B, n \sim \{\{1\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}, \mathsf{sk}_B, n}\,\mathrm{FA}}{\{\{0\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}, \mathsf{dec}(t_0, \mathsf{sk}_B), n \sim \{\{1\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}, \mathsf{dec}(t_1, \mathsf{sk}_B), n}\,\mathrm{FA}^{(2)} + \mathrm{Dup}}{\mathsf{eq}(t_0, \{\{0\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}), \langle \mathsf{dec}(t_0, \mathsf{sk}_B),\, n\rangle \sim \mathsf{eq}(t_1, \{\{1\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}), \langle \mathsf{dec}(t_1, \mathsf{sk}_B),\, n\rangle}\,\mathrm{FA}^{(3)} + \mathrm{Dup} \qquad \dfrac{\dfrac{\dfrac{}{\{\{0\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}, s_0, n \sim \{\{1\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}, s_1, n_A}\,\mathrm{CCA}_2(\mathsf{pk}_B, \mathsf{sk}_B)}{\mathsf{eq}(t_0, \{\{0\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}), \langle s_0,\, n\rangle \sim \mathsf{eq}(t_1, \{\{1\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}), \langle s_1,\, n_A\rangle}\,\mathrm{FA}^{(3)} + \mathrm{Dup}}{\begin{array}{c}\text{if } \mathsf{eq}(t_0, \{\{0\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}) \text{ then } \langle \mathsf{dec}(t_0, \mathsf{sk}_B),\, n\rangle \sim \text{if } \mathsf{eq}(t_1, \{\{1\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}) \text{ then } \langle \mathsf{dec}(t_1, \mathsf{sk}_B),\, n\rangle \\ \text{else } \langle s_0,\, n\rangle \qquad\qquad\qquad\qquad \text{else } \langle s_1,\, n_A\rangle\end{array}}\,\mathrm{CS}$$

$$\dfrac{\begin{array}{c}\langle \mathsf{dec}(t_0, \mathsf{sk}_B),\, n\rangle \sim \text{if } \mathsf{eq}(t_1, \{\{1\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}) \text{ then } \langle \mathsf{dec}(t_1, \mathsf{sk}_B),\, n\rangle \\ \text{else } \langle \mathsf{dec}(t_1, \mathsf{sk}_B),\, n_A\rangle\end{array}}{}\,R$$

Remark: in the right branch, when we apply the CCA$_2$ axiom to:

$$\{\{0\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}, s_0, n \sim \{\{1\}^{n_A}_{\mathsf{pk}_A}\}^{n_B}_{\mathsf{pk}_B}, s_1, n_A$$

we need to alpha-rename n_A by n on the right side of the formula. This is not a problem, as the extended CCA$_2$ axiom schema, given in Appendix B, is closed under alpha-renaming of names.

Fig. 4. Derivation of ϕ.

formula and a proof of the security of one round of the NSL protocol [32]. Of course, such proofs can be found automatically using our decision procedure.

In this section, and everywhere else in the paper, we describe derivations starting from the conclusion, and moving up the proof tree.

Example 2. We prove the formula below, where $g() \in \mathcal{G}$ is an adversarial function symbol:

$$\text{if } g() \text{ then } n_0 \text{ else } n_1 \sim n$$

First, we introduce a condition $g()$ 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

alpha-renaming axiom (this axiom is subsumed by CCA_2, therefore we do not include it in Ax).

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

Example 3 (Introduction on One Side). We give here an example of formula that cannot be proved without introducing a condition in a CS application and applying the CCA_2 axiom on different public/private key pair in each branch. We use the keys $\mathsf{pk}_A \equiv \mathsf{pk}(\mathsf{n}_0)$, $\mathsf{sk}_A \equiv \mathsf{sk}(\mathsf{n}_0)$, $\mathsf{pk}_B \equiv \mathsf{pk}(\mathsf{n}_1)$ and $\mathsf{sk}_B \equiv \mathsf{sk}(\mathsf{n}_1)$. Let $t_m \equiv g(\{\{m\}^{\mathsf{n}_A}_{\mathsf{pk}_A}\}^{\mathsf{n}_B}_{\mathsf{pk}_B})$. Consider the formula below:

$$\begin{aligned}\phi \ \equiv\ \langle \mathsf{dec}(t_0, \mathsf{sk}_B)\,,\ \mathsf{n}\rangle \sim \text{if } \mathsf{eq}(t_1, \{\{1\}^{\mathsf{n}_A}_{\mathsf{pk}_A}\}^{\mathsf{n}_B}_{\mathsf{pk}_B}) \text{ then } \langle \mathsf{dec}(t_1, \mathsf{sk}_B)\,,\ \mathsf{n}\rangle \\ \text{else } \langle \mathsf{dec}(t_1, \mathsf{sk}_B)\,,\ \mathsf{n}_A\rangle\end{aligned}$$

We would like to apply the CCA_2 axiom. The problem is that, on the else branch of the right term, the encryption randomness n_A is leaked. Therefore the only way to prove the else branch is to use the fact that the decryption $\mathsf{dec}(t_1, \mathsf{sk}_B)$ is under the correct guard and to apply CCA_2 on keys $(\mathsf{pk}_B, \mathsf{sk}_B)$. For the then branch, we can directly use the CCA_2 axioms on keys $(\mathsf{pk}_A, \mathsf{sk}_A)$. Let s_m be the following term:

$$s_m \ \equiv\ \text{if } \mathsf{eq}(t_m, \{\{m\}^{\mathsf{n}_A}_{\mathsf{pk}_A}\}^{\mathsf{n}_B}_{\mathsf{pk}_B}) \text{ then } \mathbf{0}(\mathsf{dec}(t_m, \mathsf{sk}_B)) \text{ else } \mathsf{dec}(t_m, \mathsf{sk}_B)$$

We give the derivation of ϕ in Figure 4. ◇

Example 4 (Proof of NSL). We consider a simple setting with an initiator A and a different respondent B playing one session each, and no key server (see Figure 5).

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

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

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:

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

During an execution of the protocol, the adversary has several choices. First, it decides whether to interact with A or B first. 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 $\{\langle \mathsf{n}_\mathsf{A}\,,\ \mathsf{A}\rangle\}^{\mathsf{n}_0}_{\mathsf{pk}_B}$ 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 $\mathbf{x}_\mathsf{B} \equiv \{\langle \mathsf{n}_\mathsf{A}\,,\ \mathsf{A}\rangle\}^{\mathsf{n}_0}_{\mathsf{pk}_B}$, and B answers:

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

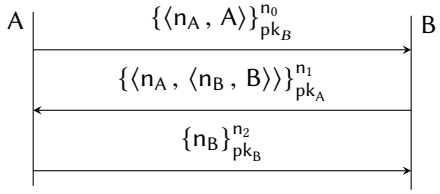


Fig. 5. The NSL protocol.

- 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 condition $\mathsf{eq}(g(\phi_1), t_\mathsf{B}[\mathbf{x}_\mathsf{B}])$ to ensure that this message is forged. The answer from A is then:

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

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 goal:

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

We let δ be the guarded decryption that will be used in the CCA_2 axiom:

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

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 CCA_2 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):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\phi_0, t_\mathsf{B}[\mathbf{x}_\mathsf{B}], \mathsf{n_A}, \delta, e_\delta \sim \phi_0, t'_\mathsf{B}[\mathbf{x}_\mathsf{B}], \mathsf{n_A}, \delta', e'_{\delta'}}\ \mathrm{CCA}_2}{\phi_0, t_\mathsf{B}[\mathbf{x}_\mathsf{B}], a_\delta, b_\delta, e_\delta \sim \phi_0, t'_\mathsf{B}[\mathbf{x}_\mathsf{B}], a'_{\delta'}, b'_{\delta'}, e'_{\delta'}}\ (\mathsf{FA}+\mathsf{Dup})^*}{\phi_0, t_\mathsf{B}[\mathbf{x}_\mathsf{B}], s_\delta \sim \phi_0, t'_\mathsf{B}[\mathbf{x}_\mathsf{B}], s'_{\delta'}}\ (\mathsf{FA}+\mathsf{Dup})^*}{\phi_0, t_\mathsf{B}[\mathbf{x}_\mathsf{B}], s \sim \phi_0, t'_\mathsf{B}[\mathbf{x}_\mathsf{B}], s'}\ R$$

◇

Remark 2. The process of computing the formula in (2) from the protocol description can be done automatically, using a simple procedure similar to the folding procedure from [8]. 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 condition $\mathsf{eq}(g(\phi_1), t_\mathsf{B}[\mathbf{x}_\mathsf{B}])$ (which does not contain nested conditions, as required by the CS side-condition). ◇

3.5 Comparison with Other Axiomatizations

We compare our core axiomatization (i.e. excluding cryptographic axioms) with the one introduced in [6].[6] Because we want a decidable axiomatization, we restricted or removed some axioms. Therefore, unsurprisingly, the axiomatization in [6] is more powerful that ours.

Some of our axioms are restricted versions of the axioms in [6]. First, the `IfBranch` axiom of [6] is exactly our CS axiom, without the restriction to if-free conditions. Second, they used a axiom `EqCong`, which states that the relation = (which is syntactic sugar for eq(_, _) ∼ true) is a congruence relation:

$$\frac{}{\mathsf{eq}(x,x) \sim \mathsf{true}} \qquad \frac{}{\mathsf{eq}(x,y) \sim \mathsf{eq}(y,x)} \qquad \frac{\mathsf{eq}(x,y) \sim \mathsf{true} \quad \mathsf{eq}(y,z) \sim \mathsf{true}}{\mathsf{eq}(x,z) \sim \mathsf{true}} \qquad \frac{\vec{u}, C[y] \sim \vec{v} \quad \mathsf{eq}(x,y) \sim \mathsf{true}}{\vec{u}, C[x] \sim \vec{v}}$$

This axiom (actually axioms) allow to do equational reasoning inside the logic, using the ∼ predicate and the eq(_, _) function symbol. Our version of this axiom, R, does the equational reasoning outside the logic, using a term rewriting system $=_R$. This allows us to use rewriting, while avoiding the general congruence axioms, which are problematic. Indeed, handling automatically the equality transitivity axiom presents the same problem as for the ∼ transitivity axiom: it introduces an arbitrarily large intermediate term z. With our TRS based axiom R, we have better control over the rewritings used in proofs, which will allow us to bound their size. But the `EqCong` axiom is, a priori, more general than R, as it allows to use the other ∼ axioms to prove an equality $\mathsf{eq}(x,y) \sim \mathsf{true}$, while we only allow for a fixed set of equalities in $=_R$.

Finally, some axioms presented in [6] are missing from our axiomatization:

$$\frac{\vec{u} \sim \vec{v} \quad \vec{v} \sim \vec{w}}{\vec{u} \sim \vec{w}}\ \texttt{Trans} \qquad \frac{}{\mathsf{true} \not\sim \mathsf{false}}\ \texttt{TFDist} \qquad \frac{\vec{u} \sim \vec{v} \quad \mathit{if}\ n \notin \mathit{st}(\vec{u}),\ n' \notin \mathit{st}(\vec{v})}{\vec{u}, \mathsf{n} \sim \vec{v}, \mathsf{n}'}\ \texttt{FreshInd}$$

$$\frac{\mathit{if}\ n \notin \mathit{st}(t)}{\mathsf{eq}(\mathsf{n}, t) \sim \mathsf{false}}\ \texttt{FreshNEq} \qquad \frac{}{\mathsf{eq}(\mathsf{if}\ b\ \mathsf{then}\ u[b]\ \mathsf{else}\ v[b],\ \ \mathsf{if}\ b\ \mathsf{then}\ u[\mathsf{true}]\ \mathsf{else}\ v[\mathsf{false}]) \sim \mathsf{true}}\ \texttt{IfEval}$$

We already explained why we excluded the transitivity axiom. The `TFDist` axiom is simple, and could maybe be added to our set of axioms. Since we closed our atomic axioms under alpha-renaming, we do not need `FreshInd`: indeed, we can just keep the names n and n′ throughout the proof until we apply an atomic axiom.[7] `FreshNEq` states that if n is independent from t (i.e. $\mathsf{n} \notin \mathsf{st}(t)$), then the probability that n and t are equal is negligible. We do not know if decidability is preserved if we add this axiom. Finally, `IfEval` allows to replace a condition by true (resp. false) in the then (resp. else) branch of an if_then_else_. Actually, the authors of [6] shows that, using `IfEval` and the other axioms, the following rule is admissible:

$$\frac{\mathsf{if}\ \mathsf{eq}(u,v)\ \mathsf{then}\ t[v]\ \mathsf{else}\ s \sim w}{\mathsf{if}\ \mathsf{eq}(u,v)\ \mathsf{then}\ t[u]\ \mathsf{else}\ s \sim w} \tag{4}$$

This means that when we rewrite a term t into a term s, we can use additional equalities coming from the conditions we are in the then branch of. Moreover, the equality condition $\mathsf{eq}(u,v)$ could have been introduced earlier, e.g. using the rewriting $t[u] =_r \mathsf{if}\ \mathsf{eq}(u,v)\ \mathsf{then}\ t[u]\ \mathsf{else}\ t[u]$. For these reasons, automatically handling the rule in (4) seems very challenging.

[6] There are other axiomatization of the BC indistinguishability logic in the literature, notably [34]. While the core axioms presented in [34] and [6] are different, they have the same expressive power (we refer the reader to [6] for equivalence proofs between different axiomatizations). The only exception is the *case disjunction axiom* of [34], which we believe cannot be expressed using [6]'s axiomatization.

[7] Actually, we believe that $\mathsf{Ax} \cup$ `FreshInd` is equivalent to Ax, though we did not prove it.

4 THE TERM REWRITING SYSTEM R

In this section, we explain how we orient $=_R$ to obtain a convergent (terminating and confluent) term rewriting system. First, we recall the definition of a Lexicographic Path Ordering [26].

Definition 6. Let $\succ_f$ be a precedence over function symbols (i.e. an order over function symbols). The lexicographic path ordering $\succ$ associated with $\succ_f$ is the relation defined by:

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

where $\succ_{lex}$ is the lexicographic ordering obtained from $\succ$.

Proposition 4 (Dershowitz, Jouannaud [26]). *If $\succ_f$ is a total precedence, then the corresponding LPO $\succ$ is a reduction ordering: $\succ$ is a well-founded order closed under substitutions and context application. Moreover, $\succ$ is a total ordering on ground terms.*

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). We define the lexicographic path ordering $\succ$ on ground terms using $\succ_f$.

Definition 7. Let $\succ$ be the lexicographic path ordering on $\mathcal{T}(\mathcal{F}, \mathcal{N})$ using precedence $\succ_f$.

Now, we want to have some leeway in the ordering of terms. We do this by letting $\succ_u$ be an arbitrary total order on if-free conditions that are $R_{\leq 3}$-irreducible. We define the extension $\succ_u^{\text{lpo}}$ of $\succ_u$ to arbitrary ground conditions. Basically, $\succ_u^{\text{lpo}}$ compares if-free $R_{\leq 3}$-irreducible conditions using $\succ_u$; conditions that are *not* if-free or *not* $R_{\leq 3}$-irreducible are compared using $\succ$; and we choose the behavior of $\succ_u^{\text{lpo}}$ on cross-cases (i.e. one if-free $R_{\leq 3}$-irreducible condition and one *not* if-free or *not* $R_{\leq 3}$-irreducible) so as to have a pre-order.

Definition 8. For any total ordering $\succ_u$ on ground if-free $R_{\leq 3}$-irreducible terms, we let $\succ_u^{\text{lpo}}$ be the relation defined on ground terms by:

$$b \succ_u^{\text{lpo}} a = \begin{cases} b \succ_u a & \text{if } a \text{ and } b \text{ are if-free and } R_{\leq 3}\text{-irreducible} \\ b \succ a & \text{if } a \text{ and } b \text{ are not if-free or not } R_{\leq 3}\text{-irreducible} \\ \text{true} & \text{if } a \text{ is if-free and } R_{\leq 3}\text{-irreducible, and } b \text{ is not} \\ \text{false} & \text{if } b \text{ is if-free and } R_{\leq 3}\text{-irreducible, and } a \text{ is not} \end{cases}$$

Note that, since $\succ$ and $\succ_u$ are total orders on ground terms, $\succ_u^{\text{lpo}}$ is also a total order on ground terms. We then order R_4 using $\succ_u^{\text{lpo}}$.

Definition 9. For any total ordering $\succ_u$ on ground if-free $R_{\leq 3}$-irreducible terms, we let $\rightarrow_{R_4^{\succ_u}}$ be the ordering of R_4 on ground terms defined by:

$$\text{if } b \text{ then (if } a \text{ then } x \text{ else } y) \text{ else } z \rightarrow \text{if } a \text{ then (if } b \text{ then } x \text{ else } z) \text{ else (if } b \text{ then } y \text{ else } z) \qquad (\text{when } b \succ_u^{\text{lpo}} a)$$

$$\text{if } b \text{ then } x \text{ else (if } a \text{ then } y \text{ else } z) \rightarrow \text{if } a \text{ then (if } b \text{ then } x \text{ else } y) \text{ else (if } b \text{ then } x \text{ else } z) \qquad (\text{when } b \succ_u^{\text{lpo}} a)$$

Since $\succ_u^{\text{lpo}}$ is a total order on ground terms, any ground instance of the equalities in R_4 is ordered by $\rightarrow_{R_4^{\succ_u}}$, from left to right or right to left.

Moreover, we 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 Figure 1 on ground terms. When the ordering $\succ_u$ is irrelevant, we write $\rightarrow_R$ instead of $\rightarrow_{R^{\succ_u}}$. We state the convergence theorem.

Theorem 2. *For all $\succ_u$, the term rewriting system $\rightarrow_{R^{\succ_u}}$ defined on ground terms is a convergent orientation of R, i.e. $=_R$ and $(\rightarrow_{R^{\succ_u}} \cup \rightarrow_{R^{\succ_u}}^{-1})^*$ are the same relations over the set of ground terms, and $\rightarrow_{R^{\succ_u}}^*$ is a convergent relation.*

Observe that this result subsumes Theorem 1.

Proof. Using Newman's lemma [29], we only need to prove that $\rightarrow_{R^{\succ_u}}$ is locally confluent and terminating.

Local Confluence (see Appendix A for details). 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 manually checked that every critical pair is joinable. This is done in Appendix A.

Termination. To prove termination, we let $\mathcal{F}_{\mathsf{term}}$ be the signature $\mathcal{F}$ to which we added a symbol $\mathsf{if}_b(\,,\,)$ for every if-free $R_{\leq 3}$-irreducible condition b:

$$\mathcal{F}_{\mathsf{term}} = \mathcal{F} \cup \left\{\mathsf{if}_b(\,,\,) \mid b \in \mathcal{T}(\mathcal{F}_{\setminus \mathsf{if}}, \mathcal{N}), b \text{ is a } R_{\leq 3}\text{-irreducible condition}\right\}$$

This yields an infinite countable signature. We extend the precedence $\succ_f$ to $\mathcal{F}_{\mathsf{term}} \cup \mathcal{N}$ 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 $\rightarrow_{R'}$ on $\mathcal{T}(\mathcal{F}_{\mathsf{term}}, \mathcal{N})$, defined by removing $\rightarrow_{R_4}$ from $\rightarrow_R$ and adding all the rules in Figure 6:

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

One can easily (but tediously) check that $\succ$ is compatible with $\rightarrow_{R'}$: the only non-trivial cases are the cases in $\rightarrow_{R_2}$ (the first rule is decreasing because $f \succ_f$ if_then_else_, the second rule using the lexicographic order), in $\rightarrow_{R_2'}$ (same arguments than for R_2) and the cases in $\rightarrow_{R_4^0}, \rightarrow_{R_4^1}, \rightarrow_{R_4^2}$ (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 $\rightarrow_{R'}$ is a terminating term rewriting systems on ground terms.

To conclude, one just has to observe that for every ground terms u, v and integer n, if $u \rightarrow_R^{(n)} v$ then there exist u', v' such that $u \rightarrow_{R^i}^! u'$, $v \rightarrow_{R^i}^! v'$ and $u' \rightarrow_{R'}^{(\geq n)} v'$. We depict this graphically in the diagram in Figure 7 We prove this easily by induction on n. Since $\rightarrow_{R'}$ is terminating on ground terms, and since any infinite sequence for $\rightarrow_R$ can be translated into an infinite sequence for $\rightarrow_{R'}$, it follows that $\rightarrow_R$ is terminating on ground terms. □

Contexts. As usual, a context $D[]_{\vec{x}}$ (written D when there is no confusion) on a signature $\mathcal{S}$ is a term in $\mathcal{T}(\mathcal{S}, \{[]_y \mid y \in \vec{x}\})$ where $\vec{x}$ are distinct special variables called holes. Given a context $D[]_{\vec{x}}$ and terms $\vec{t}$ with $|\vec{t}| = n$, we let $D[\vec{t}\,]$ be the term obtained from $D[]_{\vec{x}}$ by substituting all occurrences of the hole $[]_{x_i}$ by $\vec{t}_i$ (for every i).

Often, we want to distinguish between holes that contain "internal" conditions, and holes that contain terms appearing at the leaves. To do this we introduce the notion of if-context. An if-context $D[]_{\vec{x} \diamond \vec{y}}$ is a context using only the if_then_else_ function symbol and two sets of holes variables: $\vec{x}$ is for conditions and $\vec{y}$ is for leaves.

$$\rightarrow_{R_2'} \left\{ f(\vec{u}, \mathsf{if}_b(x\,,\,y), \vec{v}) \rightarrow \mathsf{if}_b(f(\vec{u}, x, \vec{v})\,,\, f(\vec{u}, y, \vec{v})) \quad (f \in \mathcal{F}_{\setminus \mathsf{if}}) \right.$$

$$\rightarrow_{R_3'} \begin{cases} \mathsf{if}_{\mathsf{true}}(x\,,\,y) \rightarrow x \\ \mathsf{if}_{\mathsf{false}}(x\,,\,y) \rightarrow y \\ \mathsf{if}_b(x\,,\,x) \rightarrow x \\ \mathsf{if}_b((\mathsf{if}_b(x\,,\,y))\,,\,z) \rightarrow \mathsf{if}_b(x\,,\,z) \\ \mathsf{if}_b(x\,,\,(\mathsf{if}_b(y\,,\,z))) \rightarrow \mathsf{if}_b(x\,,\,z) \end{cases}$$

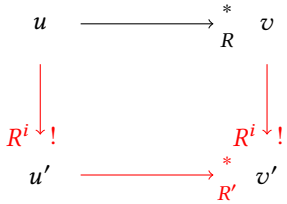
$$\rightarrow_{R_4^0} \begin{cases} \text{if } b \text{ then (if } a \text{ then } x \text{ else } y) \text{ else } z \rightarrow \text{if } a \text{ then (if } b \text{ then } x \text{ else } z) \text{ else (if } b \text{ then } y \text{ else } z) \\ \qquad\qquad (b \succ a,\ a,b \text{ not if-free or not } R_{\leq 3}\text{-irreducible}) \\ \text{if } b \text{ then } x \text{ else (if } a \text{ then } y \text{ else } z) \rightarrow \text{if } a \text{ then (if } b \text{ then } x \text{ else } y) \text{ else (if } b \text{ then } x \text{ else } z) \\ \qquad\qquad (b \succ a,\ \ a,b \text{ not if-free or not } R_{\leq 3}\text{-irreducible}) \end{cases}$$

$$\rightarrow_{R_4^1} \begin{cases} \text{if } b \text{ then } (\mathsf{if}_a(x\,,\,y)) \text{ else } z \rightarrow \mathsf{if}_a((\text{if } b \text{ then } x \text{ else } z)\,,\,(\text{if } b \text{ then } y \text{ else } z)) \\ \qquad\qquad (b \text{ not if-free or not } R_{\leq 3}\text{-irreducible}) \\ \text{if } b \text{ then } x \text{ else } (\mathsf{if}_a(y\,,\,z)) \rightarrow \mathsf{if}_a((\text{if } b \text{ then } x \text{ else } y)\,,\,(\text{if } b \text{ then } x \text{ else } z)) \\ \qquad\qquad (b \text{ not if-free or not } R_{\leq 3}\text{-irreducible}) \end{cases}$$

$$\rightarrow_{R_4^2} \begin{cases} \mathsf{if}_b((\mathsf{if}_a(x\,,\,y))\,,\,z) \rightarrow \mathsf{if}_a((\mathsf{if}_b(x\,,\,z))\,,\,(\mathsf{if}_b(y\,,\,z))) & (b \succ_u a) \\ \mathsf{if}_b(x\,,\,(\mathsf{if}_a(y\,,\,z))) \rightarrow \mathsf{if}_a((\mathsf{if}_b(x\,,\,y))\,,\,(\mathsf{if}_b(x\,,\,z))) & (b \succ_u a) \end{cases}$$

$$\rightarrow_{R^i} \left\{ \text{if } b \text{ then } u \text{ else } v \rightarrow \mathsf{if}_b(u\,,\,v) \qquad (b \text{ if-free and } R_{\leq 3}\text{-irreducible}) \right.$$

Fig. 6. The Relations $\rightarrow_{R_2'}, \rightarrow_{R_3'}, \rightarrow_{R_4^0}, \rightarrow_{R_4^1}, \rightarrow_{R_4^2}$ and $\rightarrow_{R^i}$ used for termination



Conventions: black edges stand for universal quantifications, red edges for existentials.

Fig. 7. Diagram Used in the Proof of Theorem 2.

Definition 10. For all distinct variables $\vec{x}, \vec{y}$, an if-context $D[]_{\vec{x} \diamond \vec{y}}$ is a context in:

$$\mathcal{T}(\mathsf{if\_then\_else\_}, \{[]_z \mid z \in \vec{x} \cup \vec{y}\})$$

such that for all position p, $D_{|p} \equiv \text{if } b \text{ then } u \text{ else } v$ implies:

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

Example 5. Let $\vec{x} = x_1, x_2, x_3$ and $\vec{y} = y_1, y_2, y_3, y_4$, we can define the if-context $D[]_{\vec{x} \diamond \vec{y}}$:

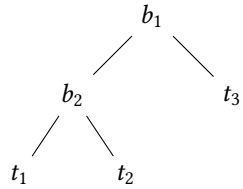
$$\begin{array}{c} \mathsf{if}\ []_{x_1}\ \mathsf{then} \begin{pmatrix} \mathsf{if}\ []_{x_2}\ \mathsf{then}\ \mathsf{if}\ []_{x_1}\ \mathsf{then}\ []_{y_1}\ \mathsf{else}\ []_{y_2} \\ \mathsf{else}\ []_{y_3} \end{pmatrix} \\ \mathsf{else}\ \left(\mathsf{if}\ []_{x_3}\ \mathsf{then}\ []_{y_2}\ \mathsf{else}\ []_{y_4}\right) \end{array}$$ ◇

The normal form of term t by $\to_{R^{\succ_u}}$ is of the form $C[\vec{b} \diamond \vec{u}]$, where $\vec{b}, \vec{u}$ are if-free terms in R-normal form. We are going to call $\vec{b}$ the conditions of $t \downarrow_{R^{\succ_u}}$, and $\vec{u}$ its leaves.

Definition 11. For every if-free terms $\vec{b}, \vec{u}$, if t is the term $C[\vec{b} \diamond \vec{u}]$ then we let cond-st(t) be the set of conditions $\vec{b}$, and leave-st(t) be the set of terms $\vec{u}$.

Example 6. Let b_1, b_2, t_1, t_2, t_3 be if-free terms, and let s be the following term (we give the labelled tree representation of s on the right):

$$\mathsf{if}\ b_1\ \mathsf{then}\ \mathsf{if}\ b_2\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2\ \mathsf{else}\ t_3$$



Then cond-st$(s) = \{b_1, b_2\}$ and leave-st$(s) = \{t_1, t_2, t_3\}$. ◇

Interestingly, the leaves and conditions of $t \downarrow_{R^{\succ_u}}$ do not depend on the order $\succ_u$ on ground conditions. Formally:

Proposition 5. *Let $\succ_u$ and $\succ'_u$ be two total orderings on if-free $R_{\leq 3}$-irreducible conditions. Then for every ground term t we have:*

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

Proof. Let C, C' be two if-contexts such that $t \downarrow_{R^{\succ_u}} \equiv C[\vec{b} \diamond \vec{u}]$ and $t \downarrow_{R^{\succ'_u}} \equiv C'[\vec{b}\,' \diamond \vec{u}\,']$ where:

$$\vec{b} = \text{leave-st}(t \downarrow_{R^{\succ_u}}) \qquad \vec{u} = \text{cond-st}(t \downarrow_{R^{\succ_u}}) \qquad \vec{b}\,' = \text{leave-st}(t \downarrow_{R^{\succ'_u}}) \qquad \vec{u}\,' = \text{cond-st}(t \downarrow_{R^{\succ'_u}})$$

We know that $C[\vec{b} \diamond \vec{u}] \to^*_{R^{\succ'_u}} C'[\vec{b}\,' \diamond \vec{u}\,']$. Since the terms $\vec{b}, \vec{u}, \vec{b}\,'$ and $\vec{u}\,'$ are if-free and in R-normal form, we can only apply the rules:

$$\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ x \to x$$
$$\mathsf{if\ true\ then}\ x\ \mathsf{else}\ y \to x$$
$$\mathsf{if\ false\ then}\ x\ \mathsf{else}\ y \to y$$
$$\mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ z \to \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z$$
$$\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ y\ \mathsf{else}\ z) \to \mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z$$
$$\mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ z \to \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ y\ \mathsf{else}\ z) \qquad (\text{when } b \succ_u^{\text{lpo}} a)$$
$$\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ (\mathsf{if}\ a\ \mathsf{then}\ y\ \mathsf{else}\ z) \to \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ y)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ x\ \mathsf{else}\ z) \qquad (\text{when } b \succ_u^{\text{lpo}} a)$$

Moreover, if a term $C_1[\vec{a}_1 \diamond \vec{v}_1]$ can be rewritten in one step into $C_2[\vec{a}_2 \diamond \vec{v}_2]$ using one of the rules above then $\vec{a}_2 \subseteq \vec{a}_1$ and $\vec{v}_2 \subseteq \vec{v}_1$. Hence, by induction, $\vec{b}\,' \subseteq \vec{b}$ and $\vec{u}\,' \subseteq \vec{u}$. Similarly, since $C'[\vec{b}\,' \diamond \vec{u}\,'] \to^*_{R^{\succ_u}} C[\vec{b} \diamond \vec{u}]$, we get that $\vec{b} \subseteq \vec{b}\,'$ and $\vec{u} \subseteq \vec{u}\,'$. We deduce that $\vec{b} \equiv \vec{b}\,'$ and $\vec{u} \equiv \vec{u}\,'$. □

By consequence, for any term u, the sets leave-st($t \downarrow_R$) and cond-st($t \downarrow_R$) are always well-defined, by taking an arbitrary ordering of if-free $R_{\leq 3}$-irreducible conditions.

5 MAIN RESULT AND DIFFICULTIES

We let Ax be the conjunction of Struct-Ax and CCA_2. We now state our main result.

Theorem (Main Result). *The following problem is decidable:*
Input: *A ground formula $\vec{u} \sim \vec{v}$.*
Question: *Is 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 CCA_2 axiom schema is closed under Sym. Therefore we can remove Perm and Sym from the set of rules. Observe that CS, FA, Dup and CCA_2 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, to obtain a complete and terminating strategy for Ax, we need to bound the size of the terms introduced when applying the R rule. The main result of this paper is a characterization of unnecessary rewritings, which yields a bound on the size of the premises of a useful R application. We will deduce from this an upper-bound on the minimal proof of a formula, if it exists.

First, we define a way of describing fragments of our logic:

Definition 12. For every formula ϕ, we write $P \vdash \phi$ if P is a proof of ϕ.

Definition 13. 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 $\mathcal{L}$.

Example 7. The derivations in Example 2 and Figure 4 are, respectively, in the fragments:

$$\mathfrak{F}\left(R \cdot \mathsf{CS} \cdot \mathsf{Refl}\right) \qquad \text{and} \qquad \mathfrak{F}\left(R \cdot \mathsf{CS} \cdot (\mathsf{FA} + \mathsf{Dup})^* \cdot \text{CCA}_2\right) \qquad \diamond$$

Necessary Introductions. As we saw in Example 2, it might be necessary to use R in the "wrong direction", typically to introduce new conditions. 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 conditions.

- First, to match the shape of the term on the other side, like $g()$ in Example 2:

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

  In this case, the introduced condition is exactly the condition that appeared on the other side of $\sim$. With more complex examples this may not be the case. Nonetheless, an introduced condition is always bounded by the condition it matches.
- Second, we might introduce a guard in order to fit to the definition of safe decryptions in the CCA_2 axioms, as in (3) in Example 4. 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.

These two (informally defined) conditions are actually sufficient: any other rewriting is a unnecessary detour. We illustrate this on an example:

Example 8 (Cut Elimination). We consider a proof of $s \sim t$ where the CS rule is applied on two conditions that have just been introduced by the R rule:

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

Here, the condition 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 $s \sim t$. ◇

The idea is that we can extract a proof of $s \sim t$ from any proof of $a, s \sim b, t$. We prove this by showing that Restr applications can be eliminated.

Lemma 1 (Restr Elimination). *For any set of atomic axioms U closed under Restr, if $P \vdash \vec{u} \sim \vec{v}$ with P in the fragment:*

$$\mathfrak{F}\left((CS + R + FA + Dup + U + Restr)^*\right)$$

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.*
- *if P is in a fragment $\mathfrak{F}(\mathcal{L})$ where $\mathcal{L}$ is closed by sub-words then P' is in $\mathfrak{F}(\mathcal{L})$.*

Proof. We do a proof by induction on the height of the derivation P of $\vec{u} \sim \vec{v}$. More precisely, we prove that for any height n and formula $\vec{u} \sim \vec{v}$, for any derivation P of $\vec{u} \sim \vec{v}$ in the fragment:

$$\mathfrak{F}\left((\mathsf{CS} + R + \mathsf{FA} + \mathsf{Dup} + \mathsf{U} + \mathsf{Restr})^*\right)$$

such that P is of height n, there exists a derivation P' with no Restr of $\vec{u} \sim \vec{v}$ of height no larger than n.

Assume that we have a derivation P of $\vec{u} \sim \vec{v}$ where the last rule applied is Restr:

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

We discriminate on the second last rule applied:

- If it is a atomic axiom in U, we conclude using the fact that U is 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:

$$\dfrac{\dfrac{\vdots\ (A)}{\vec{u}, \vec{u}', \vec{t} \sim \vec{v}, \vec{v}', \vec{t}'}\ \mathsf{FA}}{f(\vec{u}), \vec{u}', \vec{t} \sim f(\vec{v}), \vec{v}', \vec{t}'}\ \mathsf{Restr}$$

  To conclude, we apply the induction hypothesis to extract a proof of $\vec{u}, \vec{u}' \sim \vec{v}, \vec{v}'$ in the wanted fragment from (A). We conclude by applying the FA rule:

$$\dfrac{\vdots\ (A)}{\vec{u}, \vec{u}', \vec{t} \sim \vec{v}, \vec{v}', \vec{t}'}\ \mathsf{Restr} \quad \overset{\text{ind. hyp.}}{\Longrightarrow} \quad \vec{u}, \vec{u}' \sim \vec{v}, \vec{v}' \ (A') \quad \overset{\text{apply FA}}{\Longrightarrow} \quad \dfrac{\vdots\ (A')}{\vec{u}, \vec{u}' \sim \vec{v}, \vec{v}'}\ \mathsf{FA}$$

- If it is a FA axiom and $\vec{t}$ is involved in this function application then P is of the form:

$$\dfrac{\dfrac{\dfrac{\vdots\ (A)}{\vec{u}, \vec{u}', \vec{u}'' \sim \vec{v}, \vec{v}', \vec{v}''}\ \mathsf{FA}}{\vec{u}, \vec{u}', f(\vec{u}'') \sim \vec{v}, \vec{v}', f(\vec{v}'')}\ \mathsf{FA}}{\vec{u} \sim \vec{v}}\ \mathsf{Restr}$$

By applying the induction hypothesis, we extract a proof of $\vec{u} \sim \vec{v}$ in the wanted fragment:

$$\begin{array}{c}\vdots\ (A)\\ \vec{u}, \vec{u}\,', \vec{u}\,'' \sim \vec{v}, \vec{v}\,', \vec{v}\,''\end{array} \quad \overset{\text{ind. hyp.}}{\Longrightarrow} \quad \begin{array}{c}\vdots\ (A')\\ \vec{u} \sim \vec{v}\end{array}$$

- If it is CS:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\ (A)\\ \vec{w}_0, \vec{w}_1, b, (u_i)_{i\in I\cup J} \sim \vec{w}\,'_0, \vec{w}\,'_1, b', (u'_i)_{i\in I\cup J}\end{array} \quad \begin{array}{c}\vdots\ (B)\\ \vec{w}_0, \vec{w}_1, b, (v_i)_{i\in I\cup J} \sim \vec{w}\,'_0, \vec{w}\,'_1, b', (v'_i)_{i\in I\cup J}\end{array}}{\vec{w}_0, \vec{w}_1, (\text{if } b \text{ then } u_i \text{ else } v_i)_{i\in I\cup J} \sim \vec{w}\,'_0, \vec{w}\,'_1, (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_{i\in I\cup J}}\ \text{CS}}{\vec{w}_0, (\text{if } b \text{ then } u_i \text{ else } v_i)_{i\in I} \sim \vec{w}\,'_0, (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_{i\in I}}\ \text{Restr}$$

We apply the induction hypothesis twice:

$$\dfrac{\begin{array}{c}\vdots\ (A)\\ \vec{w}_0, \vec{w}_1, b, (u_i)_{i\in I\cup J} \sim \vec{w}\,'_0, \vec{w}\,'_1, b', (u'_i)_{i\in I\cup J}\end{array}}{\vec{w}_0, b, (u_i)_{i\in I} \sim \vec{w}\,'_0, b', (u'_i)_{i\in I}}\ \text{Restr} \quad \overset{\text{ind. hyp.}}{\Longrightarrow} \quad \begin{array}{c}\vdots\ (A')\\ \vec{w}_0, b, (u_i)_{i\in I} \sim \vec{w}\,'_0, b', (u'_i)_{i\in I}\end{array}$$

$$\dfrac{\begin{array}{c}\vdots\ (B)\\ \vec{w}_0, \vec{w}_1, b, (v_i)_{i\in I\cup J} \sim \vec{w}\,'_0, \vec{w}\,'_1, b', (v'_i)_{i\in I\cup J}\end{array}}{\vec{w}_0, b, (v_i)_{i\in I} \sim \vec{w}\,'_0, b', (v'_i)_{i\in I}}\ \text{Restr} \quad \overset{\text{ind. hyp.}}{\Longrightarrow} \quad \begin{array}{c}\vdots\ (B')\\ \vec{w}_0, b, (v_i)_{i\in I} \sim \vec{w}\,'_0, b', (v'_i)_{i\in I}\end{array}$$

We obtain the derivation:

$$\dfrac{\begin{array}{c}\vdots\ (A')\\ \vec{w}_0, b, (u_i)_{i\in I} \sim \vec{w}\,'_0, b', (u'_i)_{i\in I}\end{array} \quad \begin{array}{c}\vdots\ (B')\\ \vec{w}_0, b, (v_i)_{i\in I} \sim \vec{w}\,'_0, b', (v'_i)_{i\in I}\end{array}}{\vec{w}_0, (\text{if } b \text{ then } u_i \text{ else } v_i)_{i\in I} \sim \vec{w}\,'_0, (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_{i\in I}}\ \text{CS}$$

- The Dup and R axioms are trivial to handle.

Remark that in the local proof rewritings above, we never changed the order of the rule applications, but only removed some rules. It follows that if P is in a fragment $\mathfrak{F}(\mathcal{L})$ where $\mathcal{L}$ is closed by subwords, then the proof P' obtained using the Restr elimination procedure is in $\mathfrak{F}(\mathcal{L})$. □

Remark 3. In the proof, we need the atomic axioms U to be closed under Restr. Therefore, we are going to close the CCA_2 axiom schema under Restr. This adds a new difficulty: let CCA_2^a be the axiom schema before closing it under Restr, then a formula $\vec{u} \sim \vec{v}$ is an instance of CCA_2 if and only if there exists $\vec{u}', \vec{v}'$ such that $\vec{u}, \vec{u}' \sim \vec{v}, \vec{v}'$ is an instance of CCA_2^a:

$$\left\{ \dfrac{}{\vec{u} \sim \vec{v}}\ \text{CCA}_2 \right\} = \left\{ \exists \vec{u}', \vec{v}' \text{ s.t. } \dfrac{\dfrac{}{\vec{u}, \vec{u}' \sim \vec{v}, \vec{v}'}\ \text{CCA}_2^a}{\vec{u} \sim \vec{v}}\ \text{Restr} \right\}$$

Here, $\vec{u}', \vec{v}'$ can be of arbitrary size. This is problematic, as it means that to check whether $\vec{u} \sim \vec{v}$ is a valid instance of CCA_2, we have to guess two arbitrarily large vectors of terms $\vec{u}', \vec{v}'$.

We solve this in Appendix B.1: we show that there always exists $\vec{u}', \vec{v}'$ of of polynomial size w.r.t. $\vec{u}, \vec{v}$. We deduce a NP procedure to check whether a ground formula $\vec{u} \sim \vec{v}$ is an instance of CCA_2: we guess $\vec{u}', \vec{v}'$ of polynomial size, and we check that $\vec{u}, \vec{u}' \sim \vec{v}, \vec{v}'$ is an instance of CCA_2^a (which can be done in polynomial time). ◇

Using this lemma, we can deal with Example 8 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 condition followed by a CS application. For example the condition being matched could have been itself introduced earlier to match another condition, which itself was introduced to match a third condition etc.

Example 9. 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. Consider the derivation:

$$\dfrac{\dfrac{\vdots\ (A)}{a_1, b_2, b_3, u_4, w_5, u_6, v_7 \sim d_1, c_2, d_3, s_4, t_5, r_6, p_7}\ \mathsf{FA}^{(3)}}{\text{if } a \text{ then } u \text{ else } v \sim \text{if } c \text{ then } s \text{ else } t}\ R$$

where $p \equiv$ if c then s else t. Here the conditions 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 8 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 the axioms soundness, this means that in every appropriate computational model, $[\![b, b]\!] \approx [\![c, d]\!]$. Therefore, 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, $[\![c]\!]_{\eta,\rho} = [\![d]\!]_{\eta,\rho}$, except for a negligible number of random tapes ρ. ◇

A First Key Lemma. A natural question is to ask whether the semantic equality between $[\![c]\!]$ and $[\![d]\!]$ implies a syntactic equality.[8] While this is not the case in general, there are fragments of our logic in which this holds. To define such a fragment, we annotate the rules FA by the function symbol involved, and we let $\mathsf{FA_s} = \{\mathsf{FA}_f \mid f \in \mathcal{F}_{\setminus \mathsf{if},\mathbf{0}}\}$ be the restriction of FA to function symbols different from if_then_else_. Formulas that can be proven in the fragment $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathrm{CCA}_2)$ have a particular shape, which is completely characterized by the rules applied in the derivation:

Proposition 6. *For all $b, b' \in \mathcal{T}(\mathcal{F}, \mathcal{N})$, if $b \sim b'$ is in the fragment $\mathfrak{F}(FA_s^* \cdot Dup^* \cdot CCA_2)$ then $b \equiv C[\vec{w}, (\alpha_i)_i, (dec_j)_j]$, $b' \equiv C[\vec{w}, (\alpha'_i)_i, (dec'_j)_j]$ and the CCA_2 instance applied is (up-to α-renaming):*

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

where $(\alpha_i, \alpha'_i)_i$ are the encryption oracle calls and $(dec_j, dec'_j)_j$ are the decryption oracle calls.

Proof. This is easy immediate by induction on the proof derivation. □

Using this characterization, we proof the following key lemma:

Lemma 2. *For all b, b', b'', if $b, b \sim b', b''$ is in the fragment $\mathfrak{F}(FA_s^* \cdot Dup^* \cdot CCA_2)$ then $b' \equiv b''$.*

Proof. From Proposition 6 we have:

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

[8] We say that $[\![c]\!]$ and $[\![d]\!]$ are semantically equal if the bit-strings distributions $[\![c]\!]$ and $[\![d]\!]$ are equal, except for a negligible number of samplings.

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:

- 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 a hole position in both terms.
- The hole at $b_{|p}$ is mapped to an encryption oracle call $\{m\}^{\mathsf{n_e}}_{\mathsf{pk(n)}}$ in b and $\{m'\}^{\mathsf{n_e}}_{\mathsf{pk(n)}}$ in b'. Since $\{m\}^{\mathsf{n_e}}_{\mathsf{pk(n)}}$ is an encryption in the CCA_2 application, we know from the freshness side-condition that $\mathsf{n_e}$ does not appear in $\vec{w}^{\,r}$. But since $C^r_{|p}$ is not a hole, the proof of $b, b \sim b', b''$ includes the sub-proof:

$$\dfrac{\dfrac{}{\ldots, \mathsf{n_e} \sim \ldots, \mathsf{n'_e}}\ \mathrm{CCA}_2 \\ \vdots \\ \ldots, m, \mathsf{pk(n)}, \mathsf{n_e} \sim \ldots, m', \mathsf{pk(n)}, \mathsf{n_e}}{\ldots, \{m\}^{\mathsf{n_e}}_{\mathsf{pk(n)}} \sim \ldots, \{m'\}^{\mathsf{n_e}}_{\mathsf{pk(n)}}}\ \mathsf{FA}$$

  Since $\mathsf{n_e}$ is a name in $\mathcal{N}$ and cannot be modified by any rules in $\{R, \mathsf{FA_s}, \mathsf{Dup}\}$. Therefore $\mathsf{n_e} \in \vec{w}^{\,r}$. This contradict the freshness side-condition. Absurd.
- If the hole at $b_{|p}$ is mapped to a decryption oracle call $\mathsf{dec}^l_{i_0}$ in b. Since $C^r_{|p}$ is not a hole, and since function applications on $\mathsf{FA_s}$ cannot be applied on the if_then_else_ function symbols we know that there exists m, m' such that $\mathsf{dec}^l_{i_0} \equiv \mathsf{dec}(m, sk(\mathsf{n}))$ and $\mathsf{dec}'^l_{i_0} \equiv \mathsf{dec}(m', sk(\mathsf{n}))$. Moreover, since $\mathsf{dec}^l_{i_0}$ is a decryption in the CCA_2 application, we know from the key-usability side-condition that $\mathsf{sk(n)}$ appears only in decryption position in $\vec{w}^{\,r}$. Then the reasoning we have in the previous cases applies here. Indeed, we know that $C^r_{|p}$ is not a hole, hence the proof of $b, b \sim b', b''$ includes one of the following sub-proofs:

$$\dfrac{\dfrac{}{\ldots, \mathsf{sk(n)} \sim \ldots, \mathsf{sk(n)}}\ \mathrm{CCA}_2 \\ \vdots \\ \ldots, m, \mathsf{sk(n)} \sim \ldots, m', \mathsf{sk(n)}}{\ldots, \mathsf{dec}(m, \mathsf{sk(n)}) \sim \ldots, \mathsf{dec}(m', \mathsf{sk(n)})}\ \mathsf{FA}$$

  or

$$\dfrac{\dfrac{\dfrac{}{\ldots, m, \mathsf{n} \sim \ldots, m', \mathsf{n}}\ \mathrm{CCA}_2 \\ \vdots \\ \ldots, m, \mathsf{n} \sim \ldots, m', \mathsf{n}}{\ldots, m, \mathsf{sk(n)} \sim \ldots, m', \mathsf{sk(n)}}\ \mathsf{FA_{sk}} \\ \vdots \\ \ldots, m, \mathsf{sk(n)} \sim \ldots, m', \mathsf{sk(n)}}{\ldots, \mathsf{dec}(m, \mathsf{sk(n)}) \sim \ldots, \mathsf{dec}(m', \mathsf{sk(n)})}\ \mathsf{FA}$$

  Hence either $\mathsf{n} \in \vec{w}^{\,r}$ or $\mathsf{sk(n)} \in \vec{w}^{\,r}$. Absurd. □

Using this lemma, we can deal with Example 9 whenever the proof of $a_1, b_2, b_3 \sim d_1, c_2, d_3$ lies in the fragment $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathrm{CCA}_2)$. By applying the lemma on $b_2, b_3 \sim c_2, d_3$ we obtain that $c \equiv d$. By applying the lemma a second time on $a_1, b_2 \sim d_1, c_2$ (since $d \equiv c$) we deduce $a \equiv b$. Hence:

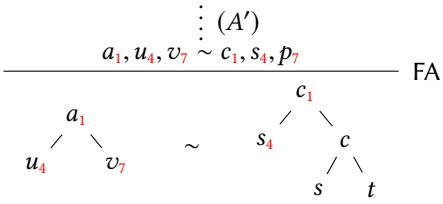
$$a_1, b_2, b_3, u_4, w_5, u_6, v_7 \sim d_1, c_2, d_3, s_4, t_5, r_6, p_7 \ \equiv\ a_1, a_2, a_3, u_4, w_5, u_6, v_7 \sim c_1, c_2, c_3, s_4, t_5, r_6, p_7$$

Therefore, using Lemma 1, we can extract a proof:

$$\dfrac{\vdots\ (A')}{a_1, u_4, v_7 \sim c_1, s_4, p_7}$$

Where, we recall, $p \equiv \text{if } c \text{ then } s \text{ else } t$. Hence we have the cut elimination:

$$\dfrac{\dfrac{\vdots\ (A')}{a_1, u_4, v_7 \sim c_1, s_4, p_7}\ \mathsf{FA}}{\text{if } a \text{ then } u \text{ else } v \sim \text{if } c \text{ then } s \text{ else } t}\ R$$



Notice that all sub-terms above are bounded, although the condition c appears twice on the right.

Proof Sketch. We sketch the outline of the remaining of the completeness 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 condition appearing in the proof is α-bounded. Intuitively a condition 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 space is finite. This yields an effective algorithm to decide our problem.

6 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, through two rule commutations lemmas, and a proof cut elimination. In the next section, we show how this restricts the shapes of the terms appearing in a proof.

6.1 Rule Commutations

Everything in this subsection applies to any set U of atomic axioms closed under Restr. We specialize to CCA_2 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:

Lemma 3. *The following rule commutations are correct:*

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

Proof. The commutations can be found in Figure 8. □

Using these rules, we obtain a first restriction.

Lemma 4. *For any set of atomic axioms U closed under Restr, the ordered strategy:*

$$\mathfrak{F}((CS + R)^* \cdot FA^* \cdot Dup^* \cdot U)$$

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

Delay FA.

- $\mathsf{FA} \cdot \mathsf{CS} \Rightarrow R \cdot \mathsf{CS} \cdot \mathsf{FA}$:

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

Can be rewritten into:

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

- $\mathsf{FA} \cdot R \Rightarrow R \cdot \mathsf{FA}$:

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

Delay Dup.

- $\mathsf{Dup} \cdot R \Rightarrow R \cdot \mathsf{Dup}$.
  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 $t' =_R s'$:

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

- $\mathsf{Dup} \cdot \mathsf{FA} \Rightarrow \mathsf{FA}^* \cdot \mathsf{Dup}$.
  Similarly if the FA rules does not involve a duplicated term then this is trivial. Otherwise:

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

- $\mathsf{Dup} \cdot \mathsf{CS} \Rightarrow \mathsf{CS} \cdot \mathsf{Dup}$. Commutation of Dup with CS is similar.

Fig. 8. Function Application and Duplicate Rules Commutations

Proof. Using Lemma 3, 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. □

Example 10. We give an example of such a proof rewriting:

$$\dfrac{\dfrac{\dfrac{x \sim z}{\pi_1(\langle x\,,\,y\rangle) \sim z}\,R}{g(\pi_1(\langle x\,,\,y\rangle)) \sim g(z)}\,\mathsf{FA}}{g(\pi_1(\langle x\,,\,y\rangle)), g(\pi_1(\langle x\,,\,y\rangle)) \sim g(z), g(z)}\,\mathsf{Dup} \quad\Rightarrow\quad \dfrac{\dfrac{\dfrac{\dfrac{x \sim z}{x, x \sim z, z}\,\mathsf{Dup}}{x, g(x) \sim z, g(z)}\,\mathsf{FA}}{g(x), g(x) \sim g(z), g(z)}\,\mathsf{FA}}{g(\pi_1(\langle x\,,\,y\rangle)), g(\pi_1(\langle x\,,\,y\rangle)) \sim g(z), g(z)}\,R \qquad \diamond$$

Splitting the FA Rule. To go further, we split the function application rules FA as follows: if the deconstructed symbol is if_then_else_ then we denote the function application by $\mathsf{FA}(b, b')$, where b, b' are the involved conditions; if the deconstructed symbol f is in $\mathcal{F}_{\setminus \mathsf{if},\mathbf{0}}$, then we denote the function application by FA_f. We give below the two new rules:

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

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

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:

$$\dfrac{\dfrac{a_1, u_1 \sim c_1, s_1}{a, u \sim c, s}\,R \qquad \dfrac{a_2, v_1 \sim c_2, t_1}{a, v \sim c, t}\,R}{\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ v \sim \mathsf{if}\ c\ \mathsf{then}\ s\ \mathsf{else}\ t}\,\mathsf{CS} \tag{5}$$

we can commute the rewritings on u, v, s and t, but not on a and c because they appear twice in the premises, and a_1 and a_2 may be different (same for c_1 and c_2).

We solve this by adding new rules to track relations between branches. We first give simplified versions. For every if-free ground conditions a and c in R-normal form, we introduce the rules:

$$\dfrac{\vec{u}, C\big[\boxed{a}\boxed{a}_a\big] \sim \vec{v}, C'\big[\boxed{c}\boxed{c}_c\big]}{\vec{u}, C[a] \sim \vec{v}, C'[c]}\,\mathsf{2Box^s} \qquad \dfrac{a_1, u \sim c_1, s \qquad a_2, v \sim c_2, t}{\mathsf{if}\ \boxed{a_1}\boxed{a_2}_a\ \mathsf{then}\ u\ \mathsf{else}\ v \sim \mathsf{if}\ \boxed{c_1}\boxed{c_2}_c\ \mathsf{then}\ s\ \mathsf{else}\ t}\,\mathsf{CS}^{\mathsf{s}}_{\square}$$

where $\boxed{\ }\boxed{\ }_a$ is a new symbol of sort $\mathsf{bool}^2 \to \mathsf{bool}$, and of fixed semantics: it ignores its arguments and has the semantics $[\![a]\!]$. Intuitively, $\boxed{a_1}\boxed{a_2}_a$ stands for the condition a, and a_1, a_2 are, respectively, the left and right versions of a. Then, using these rules, we can rewrite the derivation in (5):

$$\dfrac{\dfrac{\dfrac{a_1, u_1 \sim c_1, s_1 \qquad a_2, v_1 \sim c_2, t_1}{\mathsf{if}\ \boxed{a_1}\boxed{a_2}_a\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \sim \mathsf{if}\ \boxed{c_1}\boxed{c_2}_c\ \mathsf{then}\ s_1\ \mathsf{else}\ t_1}\,\mathsf{CS}^{\mathsf{b}}_{\square}}{\mathsf{if}\ \boxed{a}\boxed{a}_a\ \mathsf{then}\ u\ \mathsf{else}\ v \sim \mathsf{if}\ \boxed{c}\boxed{c}_c\ \mathsf{then}\ s\ \mathsf{else}\ t}\,R}{\mathsf{if}\ a\ \mathsf{then}\ u\ \mathsf{else}\ v \sim \mathsf{if}\ c\ \mathsf{then}\ s\ \mathsf{else}\ t}\,\mathsf{2Box^s}$$

The $\mathsf{2Box^s}$ allows to introduce two versions of a and c, which can be independently rewritten. Using this, we can do both rewritings before applying the $\mathsf{CS}^{\mathsf{s}}_{\square}$ rule.

Let us define formally the unrestricted rules. First, we denote $\mathcal{B}$ the set of new function symbols.

Definition 14. We let $\mathcal{B}$ be the set of function symbols:

$$\mathcal{F} \cup \{\boxed{\_}\boxed{\_}_b \mid b \text{ if-free ground condition}\}$$

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 set of equalities R_2 is *not* extended to $\mathcal{B}$. For example we have:

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

The R rule is replaced by $R_\Box$ which has an extra side-condition: $R_\Box$ can rewrite $\vec{w}, u[s]$ into $\vec{w}, u[t]$ as long as $\vec{w}, u[s]$'s boxed conditions $\left\{ \boxed{a \mid c}_b \in \mathrm{st}(\vec{w}, u[s]) \right\}$ contain t's boxed conditions $\left\{ \boxed{a \mid c}_b \in \mathrm{st}(t) \right\}$.

Definition 15. We let $R_\Box$ be the following axiom schema:

$$\frac{\vec{w}, u[t] \sim \vec{v}}{\vec{w}, u[s] \sim \vec{v}}\ R_\Box \qquad \text{when } s =_R t \text{ and } \{ \boxed{a \mid c}_b \in \mathrm{st}(t) \} \subseteq \{ \boxed{a \mid c}_b \in \mathrm{st}(\vec{w}, u[s]) \}$$

The side-condition ensures that no new arbitrary $\boxed{a \mid c}_b$ is introduced. New boxed conditions are only introduced through the 2Box rule. Similarly, the FA axiom is *not* extended to $\mathcal{B}$: boxed conditions can only be open using the $\mathrm{CS}_\Box$ rule.

Example 11. We give two examples of valid application of the $R_\Box$ rules. The first $R_\Box$ application is valid because we do not introduce any boxed condition on the left, and because we remove a boxed condition on the right. The second $R_\Box$ application is valid because the introduced boxed condition already appears in the conclusion:

$$\frac{\begin{array}{r}\mathsf{if}\ \mathsf{eq}(g(\{0\}^{\mathsf{n}}_{\mathsf{pk}}), \{0\}^{\mathsf{n}}_{\mathsf{pk}})\ \mathsf{then}\ \mathsf{dec}(g(\{0\}^{\mathsf{n}}_{\mathsf{pk}}), \mathsf{sk}) \sim v \\ \mathsf{else}\ \mathsf{dec}(g(\{0\}^{\mathsf{n}}_{\mathsf{pk}}), \mathsf{sk})\end{array}}{\mathsf{dec}(g(\{0\}^{\mathsf{n}}_{\mathsf{pk}}), \mathsf{sk}) \sim \mathsf{if}\ \boxed{a \mid c}_b\ \mathsf{then}\ v\ \mathsf{else}\ v}\ R_\Box \qquad \frac{\begin{array}{r}\mathsf{if}\ \boxed{a \mid c}_b\ \mathsf{then}\ (\mathsf{if}\ \boxed{a \mid c}_b\ \mathsf{then}\ u\ \mathsf{else}\ w) \sim t \\ \mathsf{else}\ v\end{array}}{\mathsf{if}\ \boxed{a \mid c}_b\ \mathsf{then}\ u\ \mathsf{else}\ v \sim t}\ R_\Box \quad \diamond$$

When boxing a condition c, we want the term $c_\Box$ indexing the box $\boxed{c \mid c}_{c_\Box}$ to characterize c's semantics in a proof invariant way. By consequence, we replace all boxes $\boxed{a_1 \mid a_2}_a$ in c by a, and we normalize the resulting term. Formally, we introduce the following erasure function which removes boxed condition:

Definition 16. We let 2erase be the function defined on if-free ground terms by:

$$\mathrm{2erase}(t) \equiv \begin{cases} \mathrm{2erase}(b) & \text{if } t \equiv \boxed{b_1 \mid b_2}_b \\ \mathsf{n} & \text{if } t \equiv \mathsf{n} \text{ and } \mathsf{n} \in \mathcal{N} \\ f(\mathrm{2erase}(t_1), \ldots, \mathrm{2erase}(t_n)) & \text{if } t \equiv f(t_1, \ldots, t_n) \text{ and } f \neq \mathsf{if\_then\_else\_} \end{cases}$$

Example 12. We give an example with a term containing only one boxed condition $\boxed{a \mid c}_b$:

$$\mathrm{2erase}\left(\mathsf{eq}(\mathsf{if}\ \boxed{a \mid c}_b\ \mathsf{then}\ u\ \mathsf{else}\ v, \mathsf{A})\right) \equiv \mathsf{eq}(\mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v, \mathsf{A}) \quad \diamond$$

This function is used to define the full (not simplified) versions of 2Box and $\mathrm{CS}_\Box$:

Definition 17. We let 2Box and $\mathrm{CS}_\Box$ be the axioms:

$$\frac{\vec{u}, C\left[\boxed{a \mid a}_{\mathrm{2erase}(a)\downarrow_R}\right] \sim \vec{u}', C'\left[\boxed{a' \mid a'}_{\mathrm{2erase}(a')\downarrow_R}\right]}{\vec{u}, C[a] \sim \vec{u}', C'[a']}\ \mathrm{2Box} \qquad \text{when } a, a' \in \mathcal{T}(\mathcal{F}_{\setminus \mathsf{if}} \cup \mathcal{B}, \mathcal{N})$$

$$\frac{\vec{w}, a_1, (u_i)_i \sim \vec{w}', a_1', (u_i')_i \qquad \vec{w}, a_2, (v_i)_i \sim \vec{w}', a_2', (v_i')_i}{\vec{w}, \left(\mathsf{if}\ \boxed{a_1 \mid a_2}_a\ \mathsf{then}\ u_i\ \mathsf{else}\ v_i\right)_i \sim \vec{w}', \left(\mathsf{if}\ \boxed{a_1' \mid a_2'}_{a'}\ \mathsf{then}\ u_i'\ \mathsf{else}\ v_i'\right)_i}\ \mathrm{CS}_\Box \qquad \text{when } a, a' \in \mathcal{T}(\mathcal{F}_{\setminus \mathsf{if}}, \mathcal{N})$$

Remark that for the $\mathrm{CS}_\Box$ rule to be sound we need $[\![a_1]\!]$, $[\![a_2]\!]$ and $[\![a]\!]$ to be equal, up to a negligible number of samplings (same for a_1', a_2' and a). This is not enforced by the rules, so it has to be an invariant of our strategy.

Definition 18. A term t is *well-formed* if and only if for every $\boxed{a \mid c}_b \in \mathrm{st}(t)$, $a =_R c =_R b$. We lift this to formulas as expected.

Proposition 7. *The following rules preserve well-formedness:*

$$R_\Box,\ 2Box,\ CS_\Box,\ FA_s,\ \{FA(b,b')\},\ Dup$$

Besides, $R_\Box$, $CS_\Box$ and 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. □

Remark 4. We extend cond-st to terms in $\mathcal{T}(\mathcal{B},\mathcal{N})$ in a non-obvious way, by erasing all boxes. Formally, for all $t \in \mathcal{T}(\mathcal{B},\mathcal{N})$, we let:

$$\text{cond-st}(t) = \text{cond-st}(\text{2erase}(t))$$ ◇

Ordered Strategy. We can now give the new rule commutations.

Lemma 5. *The following rule commutations are correct:*

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

Proof. The rule commutations can be found in Figure 9. □

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

Lemma 6. *For any set of atomic axioms U closed under Restr, the ordered strategy:*

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

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

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 $\mathsf{2Box} \cdot \mathsf{CS}_\Box$. 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. □

6.2 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 conditions b and b' to forbid any further applications of $\{\mathsf{FA}(b,b')\}$ to them.

Example 13. 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:

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

- $\mathsf{FA_s} \cdot \mathsf{FA}(b, b') \;\Rightarrow\; R \cdot \mathsf{FA}(b, b') \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}$

$$\dfrac{\dfrac{\vec{u}, \vec{v}, b, s, t \sim \vec{u}', \vec{v}', b', s', t'}{\vec{u}, \vec{v}, \text{if } b \text{ then } s \text{ else } t}\ \mathsf{FA}(b, b')}{\begin{array}{l}\vec{u}, f(\vec{v}, \text{if } b \text{ then } s \text{ else } t) \\ \sim\ \vec{u}', f(\vec{v}', \text{if } b' \text{ then } s' \text{ else } t')\end{array}}\ \mathsf{FA}_f \quad\Rightarrow\quad \dfrac{\dfrac{\dfrac{\vec{u}, b, s, \vec{v}, t \sim \vec{u}', b', s', \vec{v}', t'}{\vec{u}, b, \vec{v}, s, \vec{v}, t \sim \vec{u}', b', \vec{v}', s', \vec{v}', t'}\ \mathsf{Dup}}{\vec{u}, b, f(\vec{v}, s), f(\vec{v}, t) \sim \vec{u}', b', f(\vec{v}', s'), f(\vec{v}', t')}\ \mathsf{FA}_f^{(2)}\ \mathsf{FA}(b, b')}{\begin{array}{l}\vec{u}, \text{if } b \text{ then } f(\vec{v}, s) \text{ else } f(\vec{v}, t) \\ \sim\ \vec{u}', \text{if } b' \text{ then } f(\vec{v}', s') \text{ else } f(\vec{v}', t') \\ \hline \vec{u}, f(\vec{v}, \text{if } b \text{ then } s \text{ else } t) \\ \sim\ \vec{u}', f(\vec{v}', \text{if } b' \text{ then } s' \text{ else } t')\end{array}}\ R$$

- $\mathsf{CS}_\square \cdot R_\square \;\Rightarrow\; R_\square \cdot \mathsf{CS}_\square$:

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

can be rewritten into:

$$\dfrac{\begin{array}{l}(w_j^1)_j, b_1, (u_i^1)_i \sim (w_j'^1)_j, b_1', (u_i'^1)_i \quad (w_j^2)_j, b_2, (v_i^1)_i \sim (w_j'^1)_j, b_2', (v_i'^1)_i\ \mathsf{CS}_\square \\ \hline \begin{array}{rl} & (\text{if } \boxed{b_1 \mid b_2}_b \text{ then } w_j^1 \text{ else } w_j^2)_j, (\text{if } \boxed{b_1 \mid b_2}_b \text{ then } u_i^1 \text{ else } v_i^1)_i \\ \sim & (\text{if } \boxed{b_1' \mid b_2'}_{b'} \text{ then } w_j'^1 \text{ else } w_j'^2)_j, (\text{if } \boxed{b_1' \mid b_2'}_{b'} \text{ then } u_i'^1 \text{ else } v_i'^1)_i \end{array}\end{array}\ R_\square}{(w_j)_j, (\text{if } \boxed{a_1 \mid a_2}_b \text{ then } u_i \text{ else } v_i)_i \sim (w_j')_j, (\text{if } \boxed{a_1' \mid a_2'}_{b'} \text{ then } u_i' \text{ else } v_i')_i}$$

- $\mathsf{CS}_\square \cdot \mathsf{2Box} \;\Rightarrow\; R_\square \cdot \mathsf{2Box} \cdot \mathsf{CS}_\square$. Let $b, b' \in \mathcal{T}(\mathcal{F}_{\setminus \mathsf{if}} \cup \mathcal{B}, \mathcal{N})$, and let:

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

Then the following proof:

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

can be rewritten into:

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

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

Fig. 9. Function Application and Boxed Case Study Rules Commutations

For this, we define a new function symbol $\overline{\ }$ arity one, which allows to freeze a condition and prevent applications of $\{\mathsf{FA}(b, b')\}$. Basically, when we apply a rule in $\{\mathsf{FA}(b, b')\}$ on the conditions b_1 and b_2:

$$b_1 \equiv \mathsf{if}\ a_1\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \qquad\qquad b_2 \equiv \mathsf{if}\ a_2\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2$$

We replace, in the premise, a_1 by $\overline{a_1}$ in b_1 and a_2 by $\overline{a_2}$ in b_2. Then, we show that we can restrict ourselves to proofs where we never apply FA on a frozen if_then_else_ condition.

Definition 19. Let $\overline{\ }$ be a new function symbol of arity one. For every ground term s, we let $\widetilde{s}$ be:

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

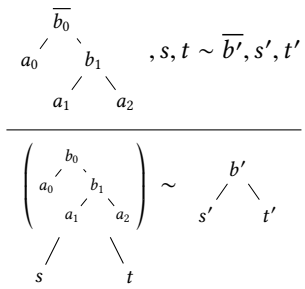
Moreover we replace every $\mathsf{FA}(b_1, b_2)$ by $\mathsf{BFA}(b_1, b_2)$, which freezes conditions b_1 and b_2.

Definition 20. We let BFA be the rule:

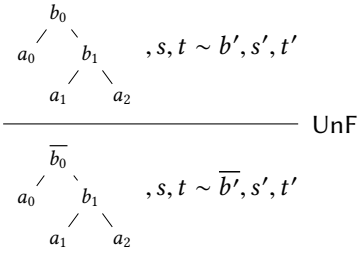
$$\frac{\vec{w}_1, \widetilde{b_1}, u_1, v_1 \sim \vec{w}_2, \widetilde{b_2}, u_2, v_2}{\vec{w}_1, \mathsf{if}\ b_1\ \mathsf{then}\ u_1\ \mathsf{else}\ v_1 \sim \vec{w}_2, \mathsf{if}\ b_2\ \mathsf{then}\ u_2\ \mathsf{else}\ v_2}\ \mathsf{BFA}(b_1, b_2)$$

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

Example 14. If the conditions b' is if-free then:



and



◇

We can extend the Restr elimination procedure of Lemma 1 to deal with the new rules $\mathsf{CS}_\Box$ and 2Box (but not $R_\Box$):

Lemma 7. *For any set of atomic axioms U closed under Restr, if $P \vdash \vec{u} \sim \vec{v}$ with P in the fragment:*

$$\mathfrak{F}\left((CS_\Box + 2Box + FA + Dup + U + Restr)^*\right)$$

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.*
- *if P is in a fragment $\mathfrak{F}(\mathcal{L})$ where $\mathcal{L}$ is closed by sub-words then P' is in $\mathfrak{F}(\mathcal{L})$.*

Proof. This is the same proof than for Lemma 1, without the R case and replacing the CS axiom by the $\mathsf{CS}_\Box$ axiom. Note that the 2Box rule is trivial to handle. □

We can state the following ordered strategy lemma:

Lemma 8. *For any set of atomic axioms U closed under Restr, the ordered strategy:*

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

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

Basically, the proof consists in eliminating all proof cuts of the shape given in Example 13. The cut elimination is simple, though voluminous. Before starting the proof, we define the induction ordering used in the proof.

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).

Example 15. Assume that $b_1 \equiv \text{if } b \text{ then } a \text{ else } c$ and $b_2 \equiv \text{if } b' \text{ then } a' \text{ else } c'$. Let P_1 be the proof:

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

And P_2 be the derivation:

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

Observe that P_1 and P_2 are two different derivations of the same formula. P_1 and P_2 are respectively interpreted as the multi-sets:

$$\{(b_1, b_2), (b, b')\} \qquad \text{and} \qquad \{(b, b'), (a, a'), (c, c')\}$$

Remark that when interpreting the derivation as multi-sets, we unfroze the conditions. The conditions 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 $(b_1, b_2) \succ_\times (c, c')$. Hence:

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

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

Proof of Lemma 8. First we are going to show a cut elimination strategy to get rid of the deconstruction of frozen conditions introduced by:

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

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

$b_1 \equiv$ if b then a else c and $b_2 \equiv$ if b' then a' else c', we know that our proof has the following shape:

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

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

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots\ (A_3)}{\vec{x}, \widetilde{b}, \widetilde{a}, \widetilde{c}, \vec{y} \sim \vec{x}', \widetilde{b'}, \widetilde{a'}, \widetilde{c'}, \vec{y}'}}{\vec{w}_1, \widetilde{b}, \widetilde{a}, \widetilde{c}, u_1, v_1 \sim \vec{w}_2, \widetilde{b'}, \widetilde{a'}, \widetilde{c'}, u_2, v_2}\ \mathsf{Dup}}{\vec{w}_1, \widetilde{b}, \widetilde{a}, u_1, v_1, \widetilde{c}, u_1, v_1 \sim \vec{w}_2, \widetilde{b'}, \widetilde{a'}, u_2, v_2, \widetilde{c'}, u_2, v_2}\ \mathsf{BFA}(c, c')}{\vec{w}_1, \widetilde{b}, \widetilde{a}, u_1, v_1, \text{if } c \text{ then } u_1 \text{ else } v_1 \sim \vec{w}_2, \widetilde{b'}, \widetilde{a'}, u_2, v_2, \text{if } c' \text{ then } u_2 \text{ else } v_2}\ \mathsf{BFA}(a, a')}{\vec{w}_1, \widetilde{b}, \text{if } a \text{ then } u_1 \text{ else } v_1, \text{if } c \text{ then } u_1 \text{ else } v_1 \sim \vec{w}_2, \widetilde{b'}, \text{if } a' \text{ then } u_2 \text{ else } v_2, \text{if } c' \text{ then } u_2 \text{ else } v_2}\ \mathsf{BFA}(b, b')$$

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

One can check that A_1 remains the same in the second proof tree since the hole in C is not in a condition 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 and P_2 are, respectively, the multi-sets:

$$M \cup \{(b_1, b_2), (b, b')\} \qquad \text{and} \qquad M \cup \{(b, b'), (a, a'), (c, c')\}$$

Therefore P_2 is a strictly smaller proof of $u \sim v$ than P_1 (this is almost the same multi-sets than in Example 15). Absurd. □

7 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 $(2\mathsf{Box} + R_\Box)^*$ part have a particular form, that we call proof form. We also show properties of this restricted shape that allow more cut eliminations.

7.1 Shape of the Terms

Most of the completeness results shown before are for any set of atomic axioms closed under Restr. We now specialize these results to CCA_2, to get some further restrictions.

When applying the atomic axioms CCA_2, we would like to require that terms are in R-normal form, e.g. to avoid the application of CCA_2 to terms with an unbounded component, such as $\pi_1(\langle u\,,\, v\rangle)$. Unfortunately, the side-conditions of CCA_2 are not stable by R. E.g., consider the CCA_2 instance:

$$\frac{}{\{\mathsf{if}\ \mathsf{eq}(g(\mathsf{n}_u),\mathsf{n}_u)\ \mathsf{then}\ A\ \mathsf{else}\ B\}^{\mathsf{n}_r}_{\mathsf{pk(n)}} \sim \{C\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}}\ \text{CCA}_2$$

The R-normal form of the left term is:

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

which cannot be used in a valid CCA_2 instance, since the condition $\mathsf{eq}(g(\mathsf{n}_u),\mathsf{n}_u)$ 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 condition branching below their encryption. This normalization strategy preserves the shape of the terms required by the CCA_2 axiom, as well as its side-conditions. In other word, if $\vec{u} \sim \vec{v}$ is a valid CCA_2 instance then its normalization $\vec{u}_{\mathsf{n}} \sim \vec{v}_{\mathsf{n}}$ is also a valid CCA_2 instance. We illustrate this on an example. The term:

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

is normalized as follows:

$$\left\{\begin{array}{l}\mathsf{if}\ b\ \mathsf{then}\ \mathsf{if}\ a\ \mathsf{then}\ \{\mathsf{if}\ d\ \mathsf{then}\ u\ \mathsf{else}\ v\}^{\mathsf{n}_1}_{\mathsf{pk}}\ \mathsf{else}\ w\\ \qquad\mathsf{else}\ \mathsf{if}\ c\ \mathsf{then}\ \{\mathsf{if}\ d\ \mathsf{then}\ u\ \mathsf{else}\ v\}^{\mathsf{n}_1}_{\mathsf{pk}}\ \mathsf{else}\ w\end{array}\right\}^{\mathsf{n}_2}_{\mathsf{pk}} \tag{6}$$

Observe that CCA_2 side-conditions are preserved. For example, the condition on occurrences of encryption randomness in (6) holds: e.g. the randomness n_1 is only used for the encryption $\{\mathsf{if}\ d\ \mathsf{then}\ u\ \mathsf{else}\ v\}^{\mathsf{n}_1}_{\mathsf{pk}}$.

Basic Terms. We omit the rewriting strategy for now (C.f. Appendix D). Instead, we describe the final shape of the terms, and prove some of their properties terms. We let $\mathcal{A}_{\succ}$ be the ordered strategy from Lemma 8, and we define several restriction of $\mathcal{A}_{\succ}$:

$$\mathfrak{F}\big((2\mathsf{Box} + R_{\square})^* \cdot \mathsf{CS}^*_{\square} \cdot \{\overline{\mathsf{BFA}}(b,b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \text{CCA}_2\big) \qquad (\mathcal{A}_{\succ})$$

$$\mathfrak{F}\big(\mathsf{CS}^*_{\square} \cdot \{\overline{\mathsf{BFA}}(b,b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \text{CCA}_2\big) \qquad (\mathcal{A}_{\mathsf{CS}_{\square}})$$

$$\mathfrak{F}\big(\{\overline{\mathsf{BFA}}(b,b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \text{CCA}_2\big) \qquad (\mathcal{A}_{\overline{\mathsf{BFA}}})$$

$$\mathfrak{F}\big(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \text{CCA}_2\big) \qquad (\mathcal{A}_{\mathsf{FA_s}})$$

The rule $\mathsf{CS}_{\square}$ is the only branching rule, therefore, after applying all the $\mathsf{CS}_{\square}$ rules, we can associate to each branch l of the proof a left CCA_2 trace $\mathcal{S}_l = (\mathcal{K}_l, \mathcal{R}_l, \mathcal{E}_l, \mathcal{D}_l)$ of the CCA_2 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 on the left side. We use $\mathcal{S}_l$ to define the normal form of the terms appearing, on the left, in branch using the CCA_2 instance $\phi \sim \psi$. 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 $\mathcal{F}_{\setminus \mathsf{if},\mathbf{0}}$ and well-formed encryptions and decryptions.
- *$\mathcal{S}_l$-normalized simple terms* are combinations of normalized basic terms using if_then_else_.

We give only the definition of $\mathcal{S}_l$-normalized basic terms (the full definitions are in Appendix D). A *$\mathcal{S}_l$-basic term* is a term build using *$\mathcal{S}_l$-encryption oracle calls*, *$\mathcal{S}_l$-decryption oracle calls*, function symbols in $\mathcal{F}_{\setminus \mathsf{if},\mathbf{0}}$ 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 CCA_2 randomness side-conditions.

- We do not decrypt terms using secret keys in $\mathcal{K}$.

To enforce the two conditions above, we define two syntactic side-conditions.

Definition 21. For every ground terms $\vec{u}, \vec{v}$, we let $\mathsf{fresh}(\vec{u}; \vec{v})$ holds if and only if no term in $\vec{v}$ is a subterm of a term in $\vec{v}$, i.e.:

$$\{u \mid u \in \vec{u}\} \cap \mathsf{st}(\vec{v}) = \emptyset$$

Definition 22. For every ground terms $\vec{u}$ and set of secret keys $\mathcal{K}$, we let $\mathsf{nodec}(\mathcal{K}, \vec{u})$ holds if and only if for every for every $\mathsf{sk}(\mathsf{n}) \in \mathcal{K}$, for every $u \in \vec{u}$, the occurrences of n in u are in subterms of the form $\mathsf{pk}(\mathsf{n})$.

Using this, we define what is a $\mathcal{S}_l$*-basic term.*

Definition 23. A $\mathcal{S}_l$*-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 $\mathcal{S}_l$*-encryption oracle calls.*
- $(\mathsf{dec}_k)_k$ are $\mathcal{S}_l$*-decryption oracle calls.*

A $\mathcal{S}_l$*-basic condition* is a $\mathcal{S}_l$*-basic term* of sort bool.

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

Definition 24. A $\mathcal{S}_l$-basic term is *normalized* if it is 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 $\mathcal{S}_l$*-normalized basic condition* is a $\mathcal{S}_l$*-normalized basic term* of sort bool.

Eager Reduction. We state here an crucial property of the $\mathcal{A}_{\mathsf{FA}_s} = \mathfrak{F}(\mathsf{FA}_s^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA}_2)$ fragment, which deals with the following proof cut: when trying to prove that $u \sim u'$ holds, one may rewrite u and u' into, respectively, $\pi_1(\langle u\,,\, v\rangle)$ and $\pi_1(\langle u'\,,\, v'\rangle)$, using R. The problem is that v and v' are arbitrary large terms. E.g. this is the case in the following proof:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\ (P)\\ u, v \sim u', v'\end{array}}{\pi_1(\langle u\,,\, v\rangle) \sim \pi_1(\langle u'\,,\, v'\rangle)}\ \mathsf{FA}_{\pi_1} \cdot \mathsf{FA}_{\langle\,,\,\rangle}}{u \sim u'}\ R$$

Of course there is a shortcut here: P is a proof of $u, v \sim u', v'$, hence by Restr we have a proof of $u \sim u'$. Using the Restr elimination procedure (Lemma 1), we obtain a proof P_{cut} of $u \sim u'$ such that P_{cut} is no larger than P. By generalizing this proof cut elimination, we show that if we have a proof $P \vdash_{\mathcal{A}_{\mathsf{FA}_s}} \beta \sim \beta'$ where β and β' are *basic terms*, then we can rewrite β and β' into *normalized basic terms* γ, γ' such that there exists P' no larger than P with $P' \vdash_{\mathcal{A}_{\mathsf{FA}_s}} \gamma \sim \gamma'$.

Lemma 9 (Informal). *Let $P \vdash_{\mathcal{A}_{FA_s}} \beta \sim \beta'$ where β and β' are* basic terms. *Then there exist $\gamma =_R \beta$ and $\gamma' =_R \beta'$ such that:*

- *γ and γ' are* normalized basic terms.
- *There exists P' such that $P' \vdash_{\mathcal{A}_{FA_s}} \gamma \sim \gamma'$, and P' contains less FA_s rules than P.*

This is stated and shown formally later, in Appendix C.2.

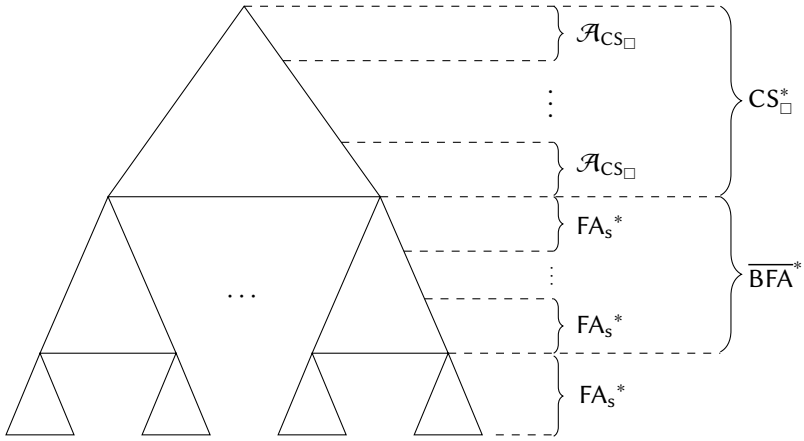


Fig. 10. The Shape of the Term is Determined by the Proof.

Normalized Proof Form. Every application of $\mathsf{CS}_\Box$:

$$\frac{a_1, u \sim b_1, s \qquad a_2, v \sim b_2, t}{\text{if } \boxed{a_1}\boxed{a_2}_a \text{ then } u \text{ else } v \sim \text{if } \boxed{b_1}\boxed{b_2}_b \text{ then } s \text{ else } t}\ \mathsf{CS}_\Box$$

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 $\mathcal{A}_{\mathsf{CS}_\Box}$. Therefore, we can check that terms after $(2\mathsf{Box} + R_\Box)^*$ are of the form informally described in Figure 10. 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 10. *Every formula in $\mathfrak{F}((CS + FA + R + Dup + CCA_2)^*)$ is provable using the strategy $\vdash^{npf}$.*

Proof. (*sketch*) The full proof is in Appendix D. First, we rewrite terms by pulling conditions upward without crossing an encryption function symbol, and without modifying decryption guards. This yield a proof P where every term s appearing at the conclusion of a sub-proof of P in the fragment $\mathcal{A}_{\mathsf{FA}_s}$ is a $\mathcal{S}_l$-basic term. Using Lemma 9, we know that there exists a smaller proof P' of the same formula such that every term s appearing at the conclusion of a sub-proof of P' in the fragment $\mathcal{A}_{\mathsf{FA}_s}$ is a $\mathcal{S}_l$-*normalized* basic term. □

7.2 Key Properties

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 8. *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 E. 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:

- If $\beta_{|p}$ is in $\vec{w}$, we simply change U to include everything up to p.
- 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 $\vec{w}$.
- 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 CCA_2 side-conditions). □

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 conditions of constant size, but b could be of any size (and we are trying to bound all conditions appearing in a proof). Also, the else branch of a true condition can contain anything and is, a priori, not bounded by the proof conclusion.

Proposition 9. *Let b an if-free condition 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 G, and relies on Proposition 8. □

8 BOUNDING THE PROOF AND DECISION PROCEDURE

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

$\overline{\mathit{BFA}}$ Rule. We already used this cut elimination to deal with Example 9 for conditions involved in $\overline{\mathsf{BFA}}$ applications. The cuts we want to eliminate are of the form:

$$\frac{a_1, a_2, u_3, v_4, w_5 \sim b_1, c_2, r_3, s_4, t_5}{\begin{array}{ccc} a_1 & & b_1 \\ a_2 \quad w_5 & \sim & c_2 \quad t_5 \\ u_3 \quad v_4 & & r_3 \quad s_4 \\ \underbrace{\qquad\qquad}_{\sigma} & & \underbrace{\qquad\qquad}_{\tau} \end{array}} \overline{\mathsf{BFA}}^{(2)} \tag{7}$$

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 $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA}_2)$. From Lemma 2 we get that $b \equiv c$. We then replace (7) by:

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

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

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

$$a_i^\Box \equiv \boxed{a_i^l}\boxed{a_i^r}_a \ (i \in \{1,2\}) \quad b_1^\Box \equiv \boxed{b_1^l}\boxed{b_1^r}_b \quad c_2^\Box \equiv \boxed{c_2^l}\boxed{c_2^r}_c$$

Consider the following cut:

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

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:

$$\mathfrak{F}(\mathsf{CS}_\Box^* \cdot \{\overline{\mathsf{BFA}}(b, b')\}^* \cdot \mathsf{UnF} \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA}_2)$$

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

Proof. (*sketch*) The full proof is given in Appendix G. It uses Proposition 9 to obtain a proof P' of $a, a' \sim b, c$ without any false and true, and also relies on Proposition 8 and Lemma 2. □

We now deal with the cut above. Using Lemma 11, 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:

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

where (A') is extracted from (A) by Lemma 7. Finally, to get a proof in $\mathcal{A}_{\succ}$, we commute the $R_\Box$ rewriting to the beginning.

8.1 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.

Spurious Conditions. A condition b without if_then_else_ and in R-normal form is said to be *spurious* in t if, when R-normalizing t, the condition b disappears. Formally, b is spurious in t if $b \notin \text{cond-st}(t \downarrow_R)$. E.g., the condition $\mathsf{eq}(\mathsf{n}_0, \mathsf{n}_1)$ 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 condition β, which may not be if-free, is spurious in t if all its leaf terms are spurious in t (i.e. $\text{leave-st}(\beta \downarrow_R) \cap \text{cond-st}(t \downarrow_R) = \emptyset$). As we saw in Example 2, we may need to introduce spurious basic conditions to carry out a proof. Still, we need to bound such terms. To do this, we characterize the basic conditions that *cannot* be removed: basically, a basic condition 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 condition of $t \sim t'$ (indeed, we cannot remove a decryption's guards, as this would not yield a valid CCA_2 instance).

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

Lemma 12. *$\vdash^{npf}_{\alpha}$ is complete with respect to $\vdash^{npf}$.*

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

Given a sequence of nested basic conditions $\beta_1 <_{\text{st}} \cdots <_{\text{st}} \beta_n$, (where $u <_{\text{st}} v$ iff $u \not\equiv v$ and $u \in \text{st}(v)$), 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 8).

Since the β_is are all pair-wise distinct (as $<_{\text{st}}$ 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 conditions, we show a triple exponential upper-bound in $|t| + |t'|$ on the size of the basic conditions appearing in a cut-free proof of $t \sim t'$.

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 $P \vdash^{\text{npf}}_{\alpha} t \sim t'$ (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 $Ax \wedge \vec{u} \nsim \vec{v}$ unsatisfiable?*

8.2 The Pure Case

In this section, we specialize the decision procedure to the pure case, where there is no cryptographic axioms: we replace the CCA_2 axiom schema by the reflexivity axiom modulo alpha-renaming. Two vectors of terms $\vec{u}$ and $\vec{v}$ are alpha-equal, which we write $\vec{u} \equiv_\alpha \vec{v}$, if and only if there exists a bijection $\theta : \mathcal{N} \to \mathcal{N}$ from names to names such that $\vec{u} \equiv \vec{v}\theta$. Then, Refl_α is the axiom schema:

$$\frac{}{\vec{u} \sim \vec{v}}\ \mathsf{Refl}_\alpha \qquad \text{when } \vec{u} \equiv_\alpha \vec{v}\theta$$

We let $\mathsf{Ax}_{\mathsf{pure}}$ be the set of axioms obtained from Ax by removing the CCA_2 axiom schema, and adding the Refl_α axiom schema, for reflexivity module alpha-renaming of names. The following lemma gives a necessary condition on t and $'$ for the formula $t \sim t$ to be derivable using $\mathsf{Ax}_{\mathsf{pure}}$. Basically, t and t' must be R-equal to the same simple term s, modulo alpha-renaming of each basic terms in s (note that the renaming can be different for each basic term).

Lemma 13. *For any ground terms t, t', if there exists a proof P using axioms Ax_{pure} of $t \sim t'$, then:*

$$t =_R u \equiv C[(s_1, \ldots, s_n) \diamond (s_{n+1}, \ldots, s_m)] \qquad \text{and} \qquad t' =_R u' \equiv C[(s'_1, \ldots, s'_n) \diamond (s'_{n+1}, \ldots, s'_m)]$$

where $C[]$ is an if-context and:

- *for every $1 \leq i \leq m$, s_i is an if-free term in R-normal form, such that:*

$$s_i \equiv_\alpha s'_i \quad \text{and} \quad s_i \in st(t\downarrow_R) \cup st(t'\downarrow_R) \ \text{ or } \ s'_i \in st(t\downarrow_R) \cup st(t'\downarrow_R)$$

  Moreover, the names of s_i and s'_i all appear in $st(t) \cup st(t')$.
- *there exists a derivation of $u \sim u'$ in $\mathfrak{F}(CS^* \cdot \{BFA(b, b')\}^* \cdot FA_s{}^* \cdot Dup^* \cdot Refl_\alpha)$.*
- *No condition appears twice on the same branch of u (resp. u').*

Proof (sketch). First, we note that since the CCA_2 axiom schema already allow for reflexivity module alpha-renaming of names, we have $\mathsf{Ax}_{\mathsf{pure}} \subseteq \mathsf{Ax}$. Hence, from Lemma 12, we can restrict ourselves to proofs in $\vdash_\alpha^{\text{npf}}$. To conclude, we look at proofs in $\vdash_\alpha^{\text{npf}}$ that only use reflexivity modulo alpha-renaming, to deduce the shape of t and t'. The details are given in Appendix H. □

Example 16. let $g() \in \mathcal{G}$ and $\mathsf{n}, \mathsf{n}', \mathsf{n}_1, \mathsf{n}_2 \in \mathcal{N}$. Consider the goal $t \sim t'$ where:

$$t \equiv \text{if } g(\mathsf{n}) \text{ then } \langle \mathsf{n}, \mathsf{n}_1 \rangle \qquad \text{and } t' \equiv \langle \mathsf{n}, \mathsf{n}' \rangle$$
$$\text{else } \langle \mathsf{n}, \mathsf{n}_2 \rangle$$

Then we have the derivation:

$$\dfrac{\dfrac{\dfrac{\text{where } \theta = \{\mathsf{n}' \mapsto \mathsf{n}_1\}}{g(\mathsf{n}), \langle \mathsf{n}, \mathsf{n}_1 \rangle \sim g(\mathsf{n}), \langle \mathsf{n}, \mathsf{n}' \rangle}\ \mathsf{Refl}_\alpha \qquad \dfrac{\text{where } \theta = \{\mathsf{n}' \mapsto \mathsf{n}_2\}}{g(\mathsf{n}), \langle \mathsf{n}, \mathsf{n}_2 \rangle \sim g(\mathsf{n}), \langle \mathsf{n}, \mathsf{n}' \rangle}\ \mathsf{Refl}_\alpha}{\text{if } g(\mathsf{n}) \text{ then } \langle \mathsf{n}, \mathsf{n}_1 \rangle \quad \text{if } g(\mathsf{n}) \text{ then } \langle \mathsf{n}, \mathsf{n}' \rangle \atop \text{else } \langle \mathsf{n}, \mathsf{n}_2 \rangle \quad \sim \quad \text{else } \langle \mathsf{n}, \mathsf{n}' \rangle}\ \text{CS}}{t \sim t'}\ R$$

Using the notations of Lemma 13, we have $t =_R C[s_1 \diamond (s_2, s_3)]$ and $t =_R C[s'_1 \diamond (s'_2, s'_3)]$ where:

$$C[x \diamond (y, z)] \equiv \text{if } x \text{ then } y \text{ else } z \qquad s_1 \equiv g(\mathsf{n}) \equiv s'_1$$

$$s'_2 \equiv \langle \mathsf{n}, \mathsf{n}' \rangle \text{ and } s_2 \equiv s'_2\{\mathsf{n}' \mapsto \mathsf{n}_1\} \qquad s'_3 \equiv \langle \mathsf{n}, \mathsf{n}' \rangle \text{ and } s_3 \equiv s'_3\{\mathsf{n}' \mapsto \mathsf{n}_2\}$$

Remark that we have a different substitution in each branch. ◇

Simplifying P. Consider a derivation of $t \sim t'$ of the form given by Lemma 13. Since all conditions appearing in the derivation are if-free, we can replace every $\mathsf{BFA}(b, b')$ application by a CS application. Moreover, we can assume that all Refl_α applications are on formulas containing no function symbols (i.e. only names), and that there are no Dup rule applications, by repeating the following proof rewritings:

$$\frac{}{\vec{u}, f(\vec{v}) \sim \vec{u}', f(\vec{v}')}\ \mathsf{Refl}_\alpha \quad \Longrightarrow \quad \frac{\dfrac{}{\vec{u}, \vec{v} \sim \vec{u}', \vec{v}'}\ \mathsf{Refl}_\alpha}{\vec{u}, f(\vec{v}) \sim \vec{u}', f(\vec{v}')}\ \mathsf{FA_s}$$

$$\frac{\dfrac{}{\vec{u}, v \sim \vec{u}', v'}\ \mathsf{Refl}_\alpha}{\vec{u}, v, v \sim \vec{u}', v', v'}\ \mathsf{Dup} \quad \Longrightarrow \quad \frac{}{\vec{u}, v, v \sim \vec{u}', v', v'}\ \mathsf{Refl}_\alpha$$

Therefore, we can assume that P is in $\mathfrak{F}(R \cdot \mathsf{CS}^* \cdot \mathsf{FA_s}^* \cdot \mathsf{Refl}_\alpha)$, where leaves contain only names.

Decision Procedure for the Pure Case. We let $w = C[(\sigma_1, \dots, \sigma_n) \diamond (\sigma_{n+1}, \dots, \sigma_m)]$ be the term obtained from u and u' by having, for every $1 \leq i \leq m$, σ_i be a term in $\{s_i; s_i'\} \cap (\mathsf{st}(t\downarrow_R) \cup \mathsf{st}(t'\downarrow_R))$ (which is always possible). To obtain back u (resp. u') from w, we just need to know the alpha-renaming used in each branch.

In every branch of w, either at least half the conditions belong to u, or at least half the conditions belong to u'. Moreover, every branch of u and u' does not contain the same condition twice. Consequently, at least half of w conditions do not appear twice. Since w's conditions are all members of $\mathsf{st}(t\downarrow_R) \cup \mathsf{st}(t'\downarrow_R)$, it follows that that each branch of w contains at most $2.N$ conditions, where N is the cardinal of $\mathsf{st}(t\downarrow_R) \cup \mathsf{st}(t'\downarrow_R)$. We deduce that w is a binary tree of depth at most $2.N + 1$ ($2.N$ different conditions plus the leaf), labeled by terms in $\mathsf{st}(t\downarrow_R) \cup \mathsf{st}(t'\downarrow_R)$. Therefore, w is of exponential size in N.

Since w is of exponential size in N, it has at most exponentially many leaves. Therefore, the proof P has at most exponentially many application of Refl_α. To guess an application of Refl_α, we just need to guess a renaming of the names of t' into the names of t (indeed, u and u' use only names of t and t'). Upper-bounding this by the number of functions from the names of t' into the names of t, each renaming is of size at most $|t|^{|t'|}$. Consequently, we can guess all Refl_α applications in time $|t|^{|t'|}$ times the number of leaves of w (which is exponential in N).

Putting everything together, we obtain the following non-deterministic decision procedure to decide if $t \sim t'$ can be derived using $\mathsf{Ax_{pure}}$:

- Guess the term w, by non-deterministically guessing a tree of depth at most $2.N + 1$, and a labeling of the tree using $\mathsf{st}(t\downarrow_R) \cup \mathsf{st}(t'\downarrow_R)$. This can be done in exponential time in N.
- Guess all the alpha-renaming, again in exponential time in N. This allow us to compute the terms u and u'.
- Check that $t =_R u$. This can be done in polynomial time in $|u| + |t\downarrow_R|$.[9] Similarly, we check that $t' =_R u'$. If both checks succeed, we know that $t \sim t'$ can be derived using $\mathsf{Ax_{pure}}$.

Since $|t\downarrow_R|$ is of size at most exponential in $|t|$, the decision procedure is in 2-NExpTime (one exponential less than the general case).

9 CONCLUSION

We designed a decision procedure for a fragment of the Bana-Comon indistinguishability logic. This allows to automatically verify that a protocol satisfies some security property. Our result can be reinterpreted, in the cryptographic game transformation setting, as a cut elimination procedure

[9] Doing this in polynomial time is not completely trivial. We R-normalize t (but not u, as this would cause an exponential blow-up), and then check that t and u are equal branch by branch. We omit the details.

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 decidability result was proven for CCA_2 only. While the complete ordered strategies presented in Section 6 apply to any cryptographic assumptions, some of the properties used to prove decidability are specific to the IND-CCA_2 cryptographic assumption (in particular the characterization of Section 7). Consequently, extending our decidability result to other cryptographic assumptions (e.g. EUF-CMA) is not straightforward, and requires further research.

ACKNOWLEDGMENTS

We thank Hubert Comon for his help and useful comments.

Most of this work was conducted while the author was at the LSV, ENS Paris-Saclay, France. This work was also partially conducted while the author was at the Max Planck Institute for Security and Privacy, Germany.

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

REFERENCES

[1] Martín Abadi and Phillip Rogaway. 2002. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). *J. Cryptology* 15, 2 (2002), 103–127.

[2] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella Béguelin, and Paul Zimmermann. 2015. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In *ACM Conference on Computer and Communications Security*. ACM, 5–17.

[3] Michael Backes, Ankit Malik, and Dominique Unruh. 2012. Computational soundness without protocol restrictions. In *ACM Conference on Computer and Communications Security*. ACM, 699–711.

[4] Michael Backes, Esfandiar Mohammadi, and Tim Ruffing. 2014. Computational Soundness Results for ProVerif - Bridging the Gap from Trace Properties to Uniformity. In *POST (Lecture Notes in Computer Science, Vol. 8414)*. Springer, 42–62.

[5] Gergei Bana, Rohit Chadha, and Ajay Kumar Eeralla. 2018. Formal Analysis of Vote Privacy Using Computationally Complete Symbolic Attacker. In *ESORICS (2) (LNCS, Vol. 11099)*. Springer, 350–372.

[6] Gergei Bana, Rohit Chadha, Ajay Kumar Eeralla, and Mitsuhiro Okada. 2020. Verification Methods for the Computationally Complete Symbolic Attacker Based on Indistinguishability. *ACM Trans. Comput. Log.* 21, 1 (2020), 2:1–2:44.

[7] G. Bana and H. Comon-Lundh. 2012. Towards Unconditional Soundness: Computationally Complete Symbolic Attacker. In *Principles of Security and Trust, 2012 (LNCS, Vol. 7215)*. Springer, 189–208.

[8] G. Bana and H. Comon-Lundh. 2014. A Computationally Complete Symbolic Attacker for Equivalence Properties. In *2014 ACM Conference on Computer and Communications Security, CCS '14*. ACM, 609–620.

[9] Gilles Barthe, Juan Manuel Crespo, Benjamin Grégoire, César Kunz, Yassine Lakhnech, Benedikt Schmidt, and Santiago Zanella Béguelin. 2013. Fully automated analysis of padding-based encryption in the computational model. In *ACM Conference on Computer and Communications Security*. ACM, 1247–1260.

[10] Gilles Barthe, Marion Daubignard, Bruce M. Kapron, Yassine Lakhnech, and Vincent Laporte. 2010. 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 (LNCS, Vol. 6355)*, Edmund M. Clarke and Andrei Voronkov (Eds.). Springer, 46–63.

[11] G. Barthe, B. Grégoire, S. Heraud, and S. Zanella Béguelin. 2011. Computer-Aided Security Proofs for the Working Cryptographer. In *Advances in Cryptology - CRYPTO, 2011 (LNCS, Vol. 6841)*. Springer, 71–90.

[12] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. 2000. Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In *EUROCRYPT (LNCS, Vol. 1807)*. Springer, 259–274.

[13] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. 1998. Relations Among Notions of Security for Public-Key Encryption Schemes. In *CRYPTO (LNCS, Vol. 1462)*. Springer, 26–45.

[14] Mihir Bellare and Phillip Rogaway. 2006. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *EUROCRYPT (LNCS, Vol. 4004)*. Springer, 409–426.
[15] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Alfredo Pironti, and Pierre-Yves Strub. 2014. Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 98–113.
[16] Bruno Blanchet. [n.d.]. *ProVerif: Cryptographic protocols verifier in the formal model.* available at http://proseccco.gforge..inria.fr/personal/bblanchet/proverif/.
[17] Bruno Blanchet. 2008. A Computationally Sound Mechanized Prover for Security Protocols. *IEEE Trans. Dependable Sec. Comput.* 5, 4 (2008), 193–207.
[18] Ran Canetti. 2001. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS*. IEEE Computer Society, 136–145.
[19] Chin-Liang Chang and Richard C. T. Lee. 1973. *Symbolic logic and mechanical theorem proving*. Academic Press.
[20] V. Cheval, H. Comon-Lundh, and S. Delaune. 2017. A procedure for deciding symbolic equivalence between sets of constraint systems. *Inf. Comput.* 255 (2017), 94–125.
[21] Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. 2018. DEEPSEC: Deciding Equivalence Properties in Security Protocols Theory and Practice. In *2018 IEEE Symposium on Security and Privacy, SP 2018*. IEEE, 529–546.
[22] Rémy Chrétien, Véronique Cortier, and Stéphanie Delaune. 2015. Decidability of Trace Equivalence for Protocols with Nonces. In *CSF*. IEEE Computer Society, 170–184.
[23] H. Comon and A. Koutsos. 2017. Formal Computational Unlinkability Proofs of RFID Protocols. In *30th Computer Security Foundations Symposium, 2017*. IEEE Computer Society, 100–114.
[24] Hubert Comon-Lundh, Véronique Cortier, and Guillaume Scerri. 2013. Tractable Inference Systems: An Extension with a Deducibility Predicate. In *CADE (LNCS, Vol. 7898)*. Springer, 91–108.
[25] Hubert Comon-Lundh, Véronique Cortier, and Eugen Zalinescu. 2010. Deciding security properties for cryptographic protocols. application to key cycles. *ACM Trans. Comput. Log.* 11, 2 (2010), 9:1–9:42.
[26] Nachum Dershowitz and Jean-Pierre Jouannaud. 1990. Rewrite Systems. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*. Elsevier and MIT Press, 243–320.
[27] Emanuele D'Osualdo, Luke Ong, and Alwen Tiu. 2017. Deciding Secrecy of Security Protocols for an Unbounded Number of Sessions: The Case of Depth-Bounded Processes. In *CSF*. IEEE Computer Society, 464–480.
[28] Alain Finkel and Philippe Schnoebelen. 2001. Well-structured transition systems everywhere! *Theor. Comput. Sci.* 256, 1-2 (2001), 63–92.
[29] Gérard P. Huet. 1980. Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems: Abstract Properties and Applications to Term Rewriting Systems. *J. ACM* 27, 4 (1980), 797–821.
[30] Charanjit S. Jutla and Arnab Roy. 2012. Decision Procedures for Simulatability. In *ESORICS (LNCS, Vol. 7459)*. Springer, 573–590.
[31] Adrien Koutsos. 2019. The 5G-AKA Authentication Protocol Privacy. In *IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019*. IEEE, 464–479. https://doi.org/10.1109/EuroSP.2019.00041
[32] Gavin Lowe. 1995. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Inf. Process. Lett.* 56, 3 (1995), 131–133.
[33] S. Meier, B. Schmidt, C. Cremers, and D. Basin. 2013. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *25th International Conference on Computer Aided Verification, CAV'13*. Springer-Verlag, 696–701.
[34] Guillaume Scerri and Ryan Stanley-Oakes. 2016. 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, 281–295. https://doi.org/10.1109/CSF.2016.27
[35] Victor Shoup. 2004. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive* 2004 (2004), 332. https://eprint.iacr.org/2004/332.

Outline of the Electronic Appendix. We prove local confluence of our term rewriting system in Appendix A. In Appendix B, we define the cryptographic axioms CCA_2, and prove some property of these axioms. In Appendix C, we prove, through a cut elimination procedure, that we can use an eager reduction strategy for some rules of R. We then define a normal form for derivations, and prove that we can assume w.l.o.g. that derivations are in normal form in Appendix D. We prove key properties of terms appearing in derivation in normal form in Appendix E, and in Appendix F we characterize subterms that corresponds to detours in proof. We use this characterization in Appendix G to show a first main proof cut elimination lemma. We prove a second main proof cut

elimination lemma in Appendix H, and show that the resulting derivations contain only subterms of bounded size.

A LOCAL CONFLUENCE OF R

In this section, we prove that for every user-chosen ordering $\succ_u$, the term rewriting system $\rightarrow_{R^{\succ u}}$ is locally confluent on ground terms. 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 $\mathrm{eq}(\_,\_)$. The critical pairs involving $\mathrm{dec}(\_,\_)$ are similar to the critical pairs involving $\pi_1(\_)$, and the critical pairs for $\mathbf{0}(\_)$ are trivial.

$$\begin{aligned} \text{if } b \text{ then } u \text{ else } v \;&\leftarrow^2\; \text{if } b \text{ then } \pi_1(\langle u\,,\,w\rangle) \text{ else } \pi_1(\langle v\,,\,w\rangle) \;\leftarrow \\ &\qquad \underline{\pi_1(\langle \text{if } b \text{ then } u \text{ else } v\,,\,w\rangle)} \;\rightarrow\; \text{if } b \text{ then } u \text{ else } v \end{aligned}$$

$$\begin{aligned} w \leftarrow \text{if } b \text{ then } w \text{ else } w \;&\leftarrow^2\; \text{if } b \text{ then } \pi_1(\langle w\,,\,u\rangle) \text{ else } \pi_2(\langle w\,,\,v\rangle) \;\leftarrow \\ &\qquad \underline{\pi_1(\langle w\,,\,\text{if } b \text{ then } u \text{ else } v\rangle)} \;\rightarrow\; w \end{aligned}$$

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

- **Critical Pairs R_2/R_2:** we assume that $b \succ_u^{\mathrm{lpo}} c$. The other possible orderings are handled in the same fashion.

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

- **Critical Pairs R_2/R_3:**

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

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

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

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

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

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

- **Critical Pairs** R_3/R_3**:**

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

$$\begin{array}{rll}
u \leftarrow \text{if true then } u \text{ else } v & \leftarrow & \underline{\text{if true then } (\text{if true then } u \text{ else } v) \text{ else } w} \\
& \rightarrow & \text{if true then } u \text{ else } w \rightarrow u
\end{array}$$

$$\begin{array}{rll}
\text{if } b \text{ then } u \text{ else } v & \leftarrow & \underline{\text{if } b \text{ then } (\text{if } b \text{ then } u \text{ else } v) \text{ else } (\text{if } b \text{ then } u \text{ else } v)} \\
& \rightarrow & \text{if } b \text{ then } u \text{ else } (\text{if } b \text{ then } u \text{ else } v) \rightarrow \text{if } b \text{ then } u \text{ else } v
\end{array}$$

- **Critical Pairs** R_3/R_4**:**

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

- **Critical Pairs** R_4/R_4**:** we assume that $a \succ_u^{\text{lpo}} b \succ_u^{\text{lpo}} c$. The other possible orderings are handled in the same fashion.

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

B THE CCA_2 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 B.2.

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 $O_{\mathsf{LR}}(\mathsf{pk}_i, b)$ that takes two messages m_0, m_1 as input and returns $\{m_b\}^{\mathsf{n}_r}_{\mathsf{pk}_i}$, 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 $O_{\mathsf{dec}}(\mathsf{sk}_i)$, but cannot call $O_{\mathsf{dec}}(\mathsf{sk}_i)$ on a cipher-text returned by $O_{\mathsf{LR}}(\mathsf{pk}_i, b)$ (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 [12]). Therefore, we allow multiple key pairs to appear in the CCA_2 axioms, and multiple encryptions over different terms using the same public key (each encryption corresponds to one call to a left-right oracle).

Decryption Guards. If we want the following to hold in any computational model

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

then we need to make sure that s is different from all $\{u_i\}^{\mathsf{n}_i}_{\mathsf{pk}}$ and that s' is different from all $\{v_i\}^{\mathsf{n}_i}_{\mathsf{pk}}$. 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 25. We define the function else^* by induction:

$$\mathsf{else}^*(\emptyset, x) \equiv x$$
$$\mathsf{else}^*\left((\mathsf{eq}(a, b)) :: \Gamma, x\right) \equiv \mathsf{if}\ \mathsf{eq}(a, b)\ \mathsf{then}\ \mathbf{0}(x)\ \mathsf{else}\ \mathsf{else}^*(\Gamma, x)$$

Example 17. Let $u \equiv t[\{v_1\}^{\mathsf{n}^1_r}_{\mathsf{pk}}, \{v_2\}^{\mathsf{n}^2_r}_{\mathsf{pk}}]$. Then:

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

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

Definition of CCA_2. 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 CCA_2 axiom. In order to do this we define by induction a binary relation $R^{\mathcal{K}}_{\text{CCA}^a_2}$ on CCA_2 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 26. Let $\mathcal{K}$ be a set of private keys. $(\phi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}})$ is a CCA_2 execution if:

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

$$(\phi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}}) R^{\mathcal{K}}_{\mathrm{CCA}_2^a} (\psi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}})$$

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

Definition 27. Let $\mathcal{K}$ be a finite set of private keys. We define the binary relation $R^{\mathcal{K}}_{\mathrm{CCA}_2^a}$ by induction:

(1) **No Call to the Oracles:** if $\mathcal{K} \sqsubseteq_d \phi$ then $(\phi, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset) R^{\mathcal{K}}_{\mathrm{CCA}_2^a} (\phi, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$ 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:

$$\big((\phi, \{u\}^{\mathsf{n}_r}_{\mathsf{pk}}), \mathcal{X}_{\mathsf{enc}} \cup \{x\}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}} \cup \{x \mapsto \mathsf{n}_r\}, \theta_{\mathsf{enc}} \cup \{x \mapsto \{u\}^{\mathsf{n}_r}_{\mathsf{pk}}\}, \lambda_{\mathsf{dec}}\big)$$
$$R^{\mathcal{K}}_{\mathrm{CCA}_2^a} \big((\psi, \{v\}^{\mathsf{n}'_r}_{\mathsf{pk}}), \mathcal{X}_{\mathsf{enc}} \cup \{x\}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}} \cup \{x \mapsto \mathsf{n}'_r\}, \theta'_{\mathsf{enc}} \cup \{x \mapsto \{v\}^{\mathsf{n}'_r}_{\mathsf{pk}}\}, \lambda'_{\mathsf{dec}}\big)$$

if there exist $t, t' \in \mathcal{T}(\mathcal{F}_{\setminus \mathbf{0}}, \mathcal{N}, \mathcal{X}_{\mathsf{enc}})$ such that:

- $(\phi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}}) R^{\mathcal{K}}_{\mathrm{CCA}_2^a} (\psi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}})$
- $u \equiv t\lambda_{\mathsf{dec}}$, $v \equiv t'\lambda'_{\mathsf{dec}}$
- $\mathsf{nodec}(\mathcal{K}; t, t')$, 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:

$$\big((\phi, \mathsf{else}^*(l, \mathsf{dec}(u, \mathsf{sk}))), \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}} \cup \{z\}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}} \cup \{z \mapsto \mathsf{else}^*(l, \mathsf{dec}(u, \mathsf{sk}))\}\big)$$
$$R^{\mathcal{K}}_{\mathrm{CCA}_2^a} \big((\psi, \mathsf{else}^*(l', \mathsf{dec}(v, \mathsf{sk}))), \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}} \cup \{z\}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}} \cup \{z \mapsto \mathsf{else}^*(l', \mathsf{dec}(v, \mathsf{sk}))\}\big)$$

if there exists $t \in \mathcal{T}(\mathcal{F}_{\setminus \mathsf{if}, \mathbf{0}}, \mathcal{N}, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}})$ such that:

- $(\phi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}}) R^{\mathcal{K}}_{\mathrm{CCA}_2^a} (\psi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}})$
- $u \equiv t\theta_{\mathsf{enc}}\lambda_{\mathsf{dec}}$ 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 $\{\_\}^{\mathsf{n}_x}_{\mathsf{pk}} \in \mathsf{codom}(\theta_{\mathsf{enc}})$ by $\{0\}^{\mathsf{n}_x}_{\mathsf{pk}}$:
$$x\sigma_{\mathsf{rand}} \in u\big\{\{0\}^{\mathsf{n}_x}_{\mathsf{pk}} / \{\_\}^{\mathsf{n}_x}_{\mathsf{pk}} \mid \{\_\}^{\mathsf{n}_x}_{\mathsf{pk}} \in \mathsf{codom}(\theta_{\mathsf{enc}})\big\} \downarrow_R$$
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 $(y'_1, \ldots, y'_m) = \mathsf{sort}(\mathcal{Y}_u\theta'_{\mathsf{enc}})$[10].
- $\mathsf{nodec}(\mathcal{K}; t)$ and $\mathsf{hidden\text{-}rand}(\mathcal{X}_{\mathsf{enc}}\sigma_{\mathsf{rand}} \cup \mathcal{X}_{\mathsf{enc}}\sigma'_{\mathsf{rand}}; \phi, u, \psi, v)$

[10]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 $\mathcal{Y}_u = \mathcal{Y}_v$.

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

Remark 5. 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:

$$\begin{aligned}
&\mathsf{if} \quad \mathsf{eq}(g(\alpha_2), \alpha_0) \ \mathsf{then}\ \mathbf{0}(\mathsf{dec}(g(\alpha_2), \mathsf{sk}))\\
&\mathsf{else\ if}\ \mathsf{eq}(g(\alpha_2), \alpha_1) \ \mathsf{then}\ \mathbf{0}(\mathsf{dec}(g(\alpha_2), \mathsf{sk}))\\
&\mathsf{else\ if}\ \mathsf{eq}(g(\alpha_2), \alpha_2) \ \mathsf{then}\ \mathbf{0}(\mathsf{dec}(g(\alpha_2), \mathsf{sk}))\\
&\mathsf{else} \qquad\qquad\qquad\qquad \mathsf{dec}(g(\alpha_2), \mathsf{sk})
\end{aligned}$$

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:

$$\begin{aligned}
&\mathsf{if}\ \mathsf{eq}(g(\alpha_2), \alpha_2) \ \mathsf{then}\ \mathbf{0}(\mathsf{dec}(g(\alpha_2), \mathsf{sk}))\\
&\qquad\qquad \mathsf{else}\ \mathsf{dec}(g(\alpha_2), \mathsf{sk})
\end{aligned}$$

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. ◇

We can now define the recursive set of axioms CCA^a_2 and show their validity. We also state and prove a key property of these axioms.

Definition 28. CCA^a_2 is the set of atomic axioms $\phi \sim \psi\mu$, where μ is a renaming of names in $\mathcal{N}$ and there exist two CCA_2 executions $\mathcal{Y}, \mathcal{Y}'$ such that:

$$\mathcal{Y} = (\phi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}}) \qquad \mathcal{Y}' = (\psi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}}) \qquad \mathcal{Y}\ R^{\mathcal{K}}_{\text{CCA}^a_2}\ \mathcal{Y}'$$

In that case, we say that $(\mathcal{Y}, \mathcal{Y}')$ is a valid CCA^a_2 application, and $\phi \sim \psi\mu$ is a valid CCA^a_2 instance.

Proposition 10. *All formulas in CCA^a_2 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 $\mathcal{M}_{\mathsf{c}}$ 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 CCA^a_2 such that $[\![\phi]\!] \not\approx_{\mathcal{M}_{\mathsf{c}}} [\![\psi]\!]$ 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 CCA_2 we know that there exist two CCA_2 executions such that:

$$(\phi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}}) R^{\mathcal{K}}_{\text{CCA}^a_2} (\psi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}})$$

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 $R^{\mathcal{K}}_{\mathrm{CCA}_2^a}$ 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 $\mathcal{M}_{\mathsf{c}}$ domain.

(1) $(\phi, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset) R^{\mathcal{K}}_{\mathrm{CCA}_2^a} (\phi, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$: 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**:

$$\big((\phi, \{u\}^{\mathsf{n}_r}_{\mathsf{pk}}), \mathcal{X}_{\mathsf{enc}} \cup \{x\}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}} \cup \{x \mapsto \mathsf{n}_r\}, \theta_{\mathsf{enc}} \cup \{x \mapsto \{u\}^{\mathsf{n}_r}_{\mathsf{pk}}\}, \lambda_{\mathsf{dec}}\big)$$
$$R^{\mathcal{K}}_{\mathrm{CCA}_2^a}\big((\psi, \{v\}^{\mathsf{n}'_r}_{\mathsf{pk}}), \mathcal{X}_{\mathsf{enc}} \cup \{x\}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}} \cup \{x \mapsto \mathsf{n}'_r\}, \theta'_{\mathsf{enc}} \cup \{x \mapsto \{v\}^{\mathsf{n}'_r}_{\mathsf{pk}}\}, \lambda'_{\mathsf{dec}}\big)$$

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:

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

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

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

$\mathcal{B}$ 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, $O_{\mathsf{LR}}(\mathsf{pk}, b)(\alpha, \beta)$ is sampled from $O_{\mathsf{LR}}(\mathsf{pk}, b)([\![u]\!], [\![v]\!])$:
- 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**:

$$((\phi, \mathsf{else}^*(l, \mathsf{dec}(u, \mathsf{sk}))), \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}} \cup \{z\}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}} \cup \{z \mapsto \mathsf{else}^*(l, \mathsf{dec}(u, \mathsf{sk}))\})$$
$$R^{\mathcal{K}}_{\mathrm{CCA}_2^a}\left((\psi, \mathsf{else}^*(l', \mathsf{dec}(v, \mathsf{sk}))), \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}} \cup \{z\}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}} \cup \{z \mapsto \mathsf{else}^*(l', \mathsf{dec}(v, \mathsf{sk}))\}\right)$$

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 $O_{\mathsf{LR}}(\mathsf{pk}, b)$ have already been stored. Let $x_1\theta_{\mathsf{enc}}, \ldots, x_p\theta_{\mathsf{enc}}$ be the corresponding keys in the store. Hence if γ is equal to any of the values stored at keys $x_1\theta_{\mathsf{enc}}, \ldots, x_p\theta_{\mathsf{enc}}$ then $\mathcal{B}$ return $[\![\mathbf{0}]\!](\gamma)$, otherwise $\mathcal{B}$ can call the decryption oracle $O_{\mathsf{dec}}(\mathsf{sk})$ on γ.
As we observed in Remark 5, if the challenger internal bit is 0, checking whether γ is different from the values sampled from $[\![x_1\theta_{\mathsf{enc}}]\!], \ldots, [\![x_p\theta_{\mathsf{enc}}]\!]$ amounts to checking whether γ is different from the values sampled from $[\![y_1]\!], \ldots, [\![y_m]\!]$, 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:

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

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

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

Hence, if the internal bit is 1 then checking whether γ is different from the values sampled from $[\![x_1\theta'_{\mathsf{enc}}]\!], \ldots, [\![x_p\theta'_{\mathsf{enc}}]\!]$ amounts to checking whether γ is different from the values sampled from $[\![y'_1]\!], \ldots, [\![y'_m]\!]$, 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. □

B.1 Closure Under Restr

To close our logic under Restr, we need the atomic axioms to be closed. Therefore, we let CCA$_2$ be the closure of CCA$_2^a$ under Restr.

Definition 29. CCA$_2$ is the set of formula $\phi \sim \psi$ such that we have the derivation:

$$\dfrac{\dfrac{}{\phi' \sim \psi'}\ \mathrm{CCA}_2^a}{\phi \sim \psi}\ \mathsf{Restr}$$

The main contribution of this sub-section, given below, states that any instance $\vec{u} \sim \vec{v}$ of CCA$_2$ can be automatically extended into an instance $\vec{u}\,' \sim \vec{v}\,'$ of CCA$_2^a$ of, at most, polynomial size.

Proposition 11. *For every instance $\vec{u} \sim \vec{v}$ of CCA$_2$, 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 CCA$_2^a$ (modulo Perm) and $|\vec{u}_1| + |\vec{v}_1|$ is of polynomial size in $|\vec{u}| + |\vec{v}|$. We let 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 CCA$_2$ into an instance of CCA$_2^a$. Let $(u_i)_{i\in I} \sim (v_i)_{i\in I}$ be an instance of CCA$_2^a$. Let $I' \subseteq I$, we want to extend $(u_i)_{i\in I'} \sim (v_i)_{i\in I'}$ into an instance of CCA$_2^a$. 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 CCA$_2^a$ we have:

$$(\phi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}}) R^{\mathcal{K}}_{\mathrm{CCA}_2^a} (\psi, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}})$$

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 $R^{\mathcal{K}}_{\mathrm{CCA}_2^a}$ (hence we have $x\lambda_{\mathsf{dec}} \equiv \mathsf{else}^*(l, \mathsf{dec}(t_{i_x}\theta_{\mathsf{enc}}\lambda_{\mathsf{dec}}), \mathsf{sk})$).

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 CCA$_2^a$ 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 $i \in I^{lr}$, $u_i \in \mathrm{st}((u_i)_{i\in I'})$ *and* $v_i \in \mathrm{st}((v_i)_{i\in I'})$. 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 $u_i \in \mathrm{st}((u_i)_{i\in I'})$). 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 CCA$_2^a$ instance $(\tilde{u}_i)_{i\in J} \sim (\tilde{v}_i)_{i\in J}$.

Inductive Definition of the Left and Right Appearance Sets. We define by induction on $i \in I'$ the sets $I_i^l, I_i^r \subseteq I$. Intuitively, I_i^l is the set of indices of I needed so that u_i is well-defined (same for I_i^r and v_i). Let $i \in I'$, we do a case disjunction on the rule applied to u_i, v_i in $R^{\mathcal{K}}_{\mathrm{CCA}_2^a}$:

- **No Call to the Oracles:** In that case we take $I_i^l = I_i^r = \{i\}$.
- **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:

$$I_i^l = \{i\} \cup \bigcup_{x\in\mathcal{X}_{\mathsf{dec}}\cap\mathrm{st}(t)} I^l_{i_x} \qquad\qquad I_i^r = \{i\} \cup \bigcup_{x\in\mathcal{X}_{\mathsf{dec}}\cap\mathrm{st}(t')} I^r_{i_x}$$

- **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 $\mathcal{X}_{\mathsf{enc}} \cap \mathrm{st}(t_i)$ and decryption in $\mathcal{X}_{\mathsf{dec}} \cap \mathrm{st}(t_i)$ to be defined, on the left and on the right. Hence we let:

$$I_i^l = \{i\} \cup \bigcup_{x\in(\mathcal{X}_{\mathsf{dec}}\cup\mathcal{X}_{\mathsf{enc}})\cap\mathrm{st}(t_i)} I^l_{i_x} \qquad\qquad I_i^r = \{i\} \cup \bigcup_{x\in(\mathcal{X}_{\mathsf{dec}}\cup\mathcal{X}_{\mathsf{enc}})\cap\mathrm{st}(t_i)} I^r_{i_x}$$

We let:

$$I^{lr} = \bigcup_{i\in I'} I_i^l \cap \bigcup_{i\in I'} I_i^r \qquad I^l = \bigcup_{i\in I'} I_i^l \cap \overline{\bigcup_{i\in I'} I_i^r} \qquad I^r = \overline{\bigcup_{i\in I'} I_i^l} \cap \bigcup_{i\in I'} I_i^r$$

These three sets are disjoint and form a partition of $\bigcup_{i\in I'} I_i^l \cup I_i^r$. Remark that for every $i \in I_j^l$, 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.

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 $l \equiv (\mathsf{eq}(u, y_1), \ldots, \mathsf{eq}(u, y_m))$ where $(y_1, \ldots, y_m) = \mathsf{sort}(\mathcal{Y}_u\theta_{\mathsf{enc}})$. 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 $\tilde{\lambda}_{\mathsf{dec}} = \{x \mapsto \tilde{u}_{i_x} \mid x \in \mathcal{X}_{\mathsf{dec}}\}$.
  - $\tilde{l} \equiv (\mathsf{eq}(\tilde{u}, \tilde{y}_1), \ldots, \mathsf{eq}(\tilde{u}, \tilde{y}_m))$ where $(\tilde{y}_1, \ldots, \tilde{y}_m) = \mathsf{sort}(\mathcal{Y}_u\tilde{\theta}_{\mathsf{enc}})$.

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

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 CCA_2^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 $|I_i^l| \leq |u_i|$ and $|I_i^r| \leq |v_i|$. We deduce that:

$$|J| = \Big|\bigcup_{i\in I'} I_i^r \cup I_i^l\Big| \leq \sum_{i\in I'} |I_i^r| + |I_i^l| \leq \sum_{i\in I'} |u_i| + |v_i|$$

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 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.2 Length in the CCA_2 Axioms

If we want the formula $\{t\}^r_{\mathsf{pk}} \sim \{t'\}^{r'}_{\mathsf{pk}'}$ to be a valid application of the CCA_2 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 CCA_2 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 CCA_2 instances are valid in any computational model $\mathcal{M}_{\mathsf{c}}$ where

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

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

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

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

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

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

Fig. 11. Definition of the Length partial function.

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:

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

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 11. Then, we let $\mathcal{D}_{\mathsf{L}}$ be the (recursive) set of atomic axioms:

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

Proposition 12. *The function Length is well defined, and the set of axioms $\mathcal{D}_L$ satisfies the soundness and 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 $[\![\mathsf{Length}]\!]_{\mathcal{M}_{\mathsf{c}}}$ is just an under-approximation of $[\![\mathsf{length}]\!]_{\mathcal{M}_{\mathsf{c}}}$ in every computational model $\mathcal{M}_{\mathsf{c}}$ 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 Length(if b then u else v). □

Remark 6. 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 $\mathsf{Length}(A) = \mathsf{Length}(B) = \mathsf{Length}(C) = l_{\mathsf{agent}}$. ◇

C SHAPE OF THE TERMS

In this section, we give the definitions of *$\mathcal{S}$-encryption oracle calls*, *$\mathcal{S}$-decryption oracle calls*, *$\mathcal{S}$-normalized basic terms* and *$\mathcal{S}$-normalized simple terms*, which we omitted in Section 7.

C.1 Definitions

Definition 30. A CCA_2 trace $\mathcal{S}$ is a tuple $(\mathcal{K}, \mathcal{R}, \mathcal{E}, \mathcal{D})$ where:

- $\mathcal{K} \subseteq \{\mathsf{sk(n)} \mid \mathsf{n} \in \mathcal{N}\}$ is a set of secret keys.
- $\mathcal{R} \subseteq \mathcal{N}$ is a set of encryption randomness.
- $\mathcal{E} \subseteq \{\{m\}^{\mathsf{n_e}}_{\mathsf{pk(n)}} \mid \mathsf{n_e} \in \mathcal{R} \wedge \mathsf{sk(n)} \in \mathcal{K}\}$ is a set of encryptions.
- $\mathcal{D} \subseteq \{\mathsf{dec}(m, \mathsf{sk(n)}) \mid \mathsf{sk(n)} \in \mathcal{K}\}$ is a set of decryptions.

We can associate to every CCA_2 instance a left and a right CCA_2 trace.

Definition 31. Given a CCA_2 instance $\phi \sim \psi$ and its corresponding CCA_2^a application:

$$(\_, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma_{\mathsf{rand}}, \theta_{\mathsf{enc}}, \lambda_{\mathsf{dec}}) R^{\mathcal{K}}_{\text{CCA}_2^a} (\_, \mathcal{X}_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}, \sigma'_{\mathsf{rand}}, \theta'_{\mathsf{enc}}, \lambda'_{\mathsf{dec}})$$

we define the left CCA_2 trace $\mathcal{S} = \mathsf{l\text{-}trace}(\phi \sim \psi)$ by:

$$\mathcal{S} = (\mathcal{K}, \mathcal{X}_{\mathsf{enc}}\sigma_{\mathsf{rand}}, \mathcal{X}_{\mathsf{enc}}\theta_{\mathsf{enc}}, \mathcal{X}_{\mathsf{dec}}\lambda_{\mathsf{dec}})$$

We define similarly its right CCA_2 trace $\mathcal{S}' = \mathsf{r\text{-}trace}(\phi \sim \psi)$.

Let $\phi \sim \psi$ be a CCA_2 instance and $\mathcal{S} = \mathsf{l\text{-}trace}(\phi \sim \psi)$ be its left CCA_2 trace. We use $\mathcal{S}$ to define the normal form of the terms appearing, on the left, in branch using the CCA_2 instance $\phi \sim \psi$. This is done through four mutually inductive definitions:

- *$\mathcal{S}$-encryption oracle calls* are well-formed encryptions.
- *$\mathcal{S}$-decryption oracle calls* are well-formed decryptions.
- *$\mathcal{S}$-normalized basic terms* are terms built using function symbols in $\mathcal{F}_{\setminus \mathsf{if}, \mathbf{0}}$ and well-formed encryptions and decryptions.
- *$\mathcal{S}$-normalized simple terms* are combinations of normalized basic terms using if_then_else_.

Later, we prove that all intermediate terms in 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 define *$\mathcal{S}$-simple terms* to be terms having a particular form, and *$\mathcal{S}$-normalized simple terms* to be *$\mathcal{S}$-simple terms* satisfying some further properties.

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

Definition 32. A valid public/private key pair is a pair of terms $(\mathsf{pk(n)}, \mathsf{sk(n)})$ where n is a name.

An *$\mathcal{S}$-encryption oracle call* is a valid encryption in $\mathcal{E}$ of the form $\{u\}^{\mathsf{n_e}}_{\mathsf{pk}}$, where $\mathsf{n_e}$ is a valid encryption randomness in $\mathcal{R}$, pk is a valid public/private key pair appearing in $\mathcal{K}$ and the encrypted plain-text u is, inductively, a *$\mathcal{S}$-normalized simple term.*

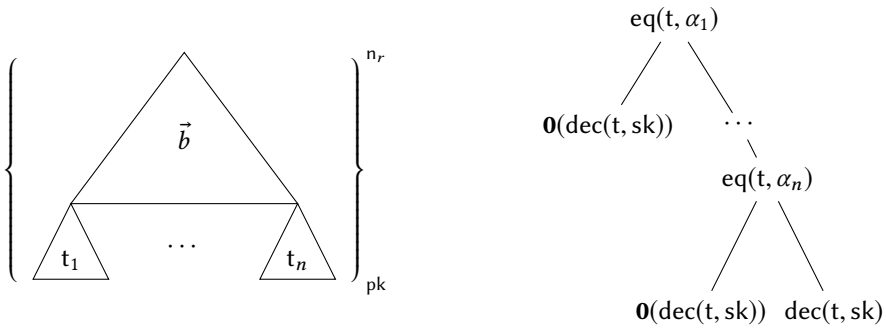
Definition 33. A *$\mathcal{S}$-encryption oracle call* is a term of the form $\{u\}^{\mathsf{n_e}}_{\mathsf{pk}}$ where:

- $\{u\}^{\mathsf{n_e}}_{\mathsf{pk}} \in \mathcal{E}$, $\mathsf{n_e} \in \mathcal{R}$, $(\mathsf{pk}, \mathsf{sk})$ 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 34. A *$\mathcal{S}$-decryption oracle call* is a term of the form $C\left[\vec{g} \diamond (s_i)_{i \leq p}\right]$ in $\mathcal{D}$ where:

- $(\mathsf{pk}, \mathsf{sk})$ is valid public/private key pair and $\mathsf{sk} \in \mathcal{K}$.



Encryption Oracle Call **Decryption Oracle Call**

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

Fig. 12. Shapes of Encryption and Decryption Oracle Calls

- There exists a context u if-free and in R-normal form, and a term t such that:

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

- 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 $(\mathrm{dec}_k)_k$ to denote a vector of decryption oracle calls.

Figure 12 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 $\mathcal{F}_{\setminus \mathrm{if}, \mathbf{0}}$ 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 CCA_2 randomness side-conditions.
- We do not decrypt terms using secret keys in $\mathcal{K}$.

Definition 35. A *$\mathcal{S}$-basic term* is a term of the form $U[\vec{w}, (\alpha_j)_j, (\mathrm{dec}_k)_k]$ where:

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

A *$\mathcal{S}$-basic condition* is a *$\mathcal{S}$-basic term* of sort bool.

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

Definition 36. A *$\mathcal{S}$-normalized basic term* is a *$\mathcal{S}$-basic term* of the form $U[\vec{w}, (\alpha_j)_j, (\mathrm{dec}_k)_k]$ where:

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

A *$\mathcal{S}$-normalized basic condition* is a *$\mathcal{S}$-normalized basic term* of sort bool.

Finally, a *$\mathcal{S}$-simple term* is a term build using only *$\mathcal{S}$-basic term* and the 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 37. 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 conditions* (resp. *$\mathcal{S}$-normalized basic conditions*).
- $\vec{u}$ are *$\mathcal{S}$-basic terms* (resp. *$\mathcal{S}$-normalized basic terms*).

Remark 7. 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 $\mathsf{nodec}(\mathcal{K}, \cdot)$. ◇

Orderings. The inductive definition of $\mathcal{S}$-normalized basic terms naturally gives us a well-founded 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 38. $<^{\mathcal{S}}_{\mathsf{ind}}$ is the reflexive and transitive closure of the relation $<^{\mathcal{S}}$ defined as:

- For all $\mathcal{S}$-encryption oracle call $t \equiv \{u\}^r_{\mathsf{pk}}$, $u <^{\mathcal{S}} t$.
- 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, $\alpha_j <^{\mathcal{S}} t$ and for all k, $\mathsf{dec}_k <^{\mathcal{S}} t$.
- For all $\mathcal{S}$-normalized basic term $t \equiv U[\vec{w}, (\alpha_j)_j, (\mathsf{dec}_k)_k]$, 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 $\le^{\mathcal{S}}_{\mathsf{bt}}$ be union of the restriction of $<^{\mathcal{S}}_{\mathsf{ind}}$ 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 39. Let $<'^{\mathcal{S}}_{\mathsf{ind}}$ be the reflexive and transitive closure of the order $<'^{\mathcal{S}}$, 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, $\alpha_j <'^{\mathcal{S}} t$; for all k, $\mathsf{dec}_k <'^{\mathcal{S}} t$; and for all $b \in \vec{g}$, $b <'^{\mathcal{S}} t$.

We finally define $\le^{\mathcal{S}}_{\mathsf{bt}}$ by requiring that for every terms u, v:
$$u \le^{\mathcal{S}}_{\mathsf{bt}} v \quad \text{iff} \quad u <'^{\mathcal{S}}_{\mathsf{ind}} v \ \text{ and } \ u \text{ is a } \mathcal{S}\text{-normalized basic term}$$

C.2 Eager Reduction for $\mathcal{A}_{\mathsf{FA_s}}$

We now prove that if we have a proof $P \vdash_{\mathcal{A}_{\mathsf{FA_s}}} \beta \sim \beta'$ where β and β' are *basic terms*, then we can rewrite β and β' into *normalized basic terms* γ, γ' such that there exists P' no larger than P with $P' \vdash_{\mathcal{A}_{\mathsf{FA_s}}} \gamma \sim \gamma'$.

To prove this, we may have to extract several sub-proofs of P, and then recombine them into a single proof P'. While the rule $\mathsf{FA_s}$ and Dup can be easily re-combined, this is not the case for CCA_2. Therefore, given a finite family of CCA_2 instances $(\vec{u}_i \sim \vec{v}_i)_{i \in I}$, we give a sufficient condition guaranteeing that they can be recombined into a single proof $(\vec{u}_i)_{i \in I} \sim (\vec{v}_i)_{i \in I}$.

Definition 40. For every P in $\mathcal{A}_{\mathsf{FA_s}}$, we let $\mathsf{instance}(P)$ be the unique CCA_2 instance used in P.

Example 18. If P is the proof:
$$\dfrac{\dfrac{}{\vec{w}, (\alpha_i)_{i \in I}, (\mathsf{dec}_j)_{j \in J} \sim \vec{w}, (\alpha'_i)_{i \in I}, (\mathsf{dec}'_j)_{j \in J}}\ \mathsf{CCA}_2}{\vdots}$$
$$\dfrac{\vdots}{C[\vec{w}, (\alpha_i)_{i \in I}, (\mathsf{dec}_j)_{j \in J}] \sim C[\vec{w}, (\alpha'_i)_{i \in I}, (\mathsf{dec}'_j)_{j \in J}]}\ \mathsf{FA_s}^* \cdot \mathsf{Dup}^*$$

then instance(P) is the CCA$_2$ instance $\vec{w}, (\alpha_i)_{i\in I}, (\mathsf{dec}_j)_{j\in J} \sim \vec{w}, (\alpha'_i)_{i\in I}, (\mathsf{dec}'_j)_{j\in J}$. ◇

We say that a CCA$_2$ instance ϕ is a sub-instance of another CCA$_2$ instance ψ if the set of encryptions and decryptions of ϕ are included into, respectively, the set of encryptions and decryptions of ψ. Moreover, we require that the symmetric part of ϕ contains only sub-terms of the symmetric part of ψ.

Definition 41. A CCA$_2$ instance:

$$\vec{w}_0, (\alpha_i)_{i\in I_0}, (\mathsf{dec}_j)_{j\in J_0} \sim \vec{w}_0, (\alpha'_i)_{i\in I_0}, (\mathsf{dec}'_j)_{j\in J_0}$$

is a sub-instance of a CCA$_2$ instance:

$$\vec{w}, (\alpha_i)_{i\in I}, (\mathsf{dec}_j)_{j\in J} \sim \vec{w}, (\alpha'_i)_{i\in I}, (\mathsf{dec}'_j)_{j\in J}$$

if and only if $\mathrm{st}(\vec{w}_0) \subseteq \mathrm{st}(\vec{w})$, $I_0 \subseteq I$ and $J_0 \subseteq J$.

The following proposition allows to re-combine several proofs $P_1, \dots, P_n$, as long as there exists a CCA$_2$ instance $\vec{u} \sim \vec{v}$ such that for every i, instance(P_i) is a sub-instance of $\vec{u} \sim \vec{v}$.

Proposition 13. *Let $(\beta_n)_{n\in N}$ and $(\beta'_n)_{n\in N}$ be such that for every $n \in N$, there exists a proof $P_n \vdash_{\mathcal{A}_{FA_s}} \beta_n \sim \beta'_n$. If there exists a CCA$_2$ instance $\vec{u} \sim \vec{v}$ such that for every n, instance(P_n) is a sub-instance of $\vec{u} \sim \vec{v}$, then there exists P such that:*

- $P \vdash_{\mathcal{A}_{FA_s}} (\beta_n)_{n\in N} \sim (\beta'_n)_{n\in N}$
- *instance(P) is a sub-instance of $\vec{u} \sim \vec{v}$.*
- *P contain the same number of FA$_s$ rules than the derivations $P_1, \dots, P_N$ altogether.*

Proof. Axioms FA$_s$ and Dup verify a frame property. More precisely:

$$\text{if} \quad \frac{\vec{u}' \sim \vec{v}'}{\vec{u} \sim \vec{v}}\ Ax \qquad \text{then for every } \vec{w}_\mathrm{l}, \vec{w}_\mathrm{r} \text{ of the same length} \qquad \frac{\vec{w}_\mathrm{l}, \vec{u}' \sim \vec{w}_\mathrm{r}, \vec{v}'}{\vec{w}_\mathrm{l}, \vec{u} \sim \vec{w}_\mathrm{r}, \vec{v}}\ Ax$$

Therefore we can easily combine all proofs $(P_n)_{n\in N}$. For every $n \in N$, we let instance$(P_n) \equiv \vec{u}_n \sim \vec{u}'_n$. Moreover, we let $(\vec{v}_n)_{n\in N} \sim (\vec{v}'_n)_{n\in N}$ be the formula obtained from $(\vec{u}_n)_{n\in N} \sim (\vec{u}'_n)_{n\in N}$ by removing all duplicates, and where for every n, $\vec{v}_n \subseteq \vec{u}_n$ and $\vec{v}'_n \subseteq \vec{u}'_n$. Then we have the derivation:

$$\cfrac{\cfrac{(\vec{v}_n)_{n\in N} \sim (\vec{v}'_n)_{n\in N}}{(\vec{u}_n)_{n\in N} \sim (\vec{u}'_n)_{n\in N}}\ \mathsf{Dup}^*}{(\beta_n)_{n\in N} \sim (\beta'_n)_{n\in N}}$$

Now, we want to conclude by applying the CCA$_2$ axiom. The problem is that CCA$_2$ does not verify the frame property. But using the fact that for every n, $\vec{u}_n \sim \vec{u}'_n$ is a sub-instance of $\vec{u} \sim \vec{v}$, and that $(\vec{v}_n)_{n\in N} \sim (\vec{v}'_n)_{n\in N}$ does not contain duplicates, we can check that $(\vec{v}_n)_{n\in N} \sim (\vec{v}'_n)_{n\in N}$ is a sub-instance of $\vec{u} \sim \vec{v}$. Hence we have a valid derivation in $\mathcal{A}_{\mathsf{FA}_s}$. □

We now have the tools to formally state and prove Lemma 9.

Lemma 14. *Let $P \vdash_{\mathcal{A}_{FA_s}} \beta \sim \beta'$ and $\mathcal{S}, \mathcal{S}'$ be the, respectively, left and right CCA$_2$ trace corresponding to instance(P). If β and β' are, respectively, $\mathcal{S}$-*basic term *and $\mathcal{S}'$-*basic term *then there exist $\gamma =_R \beta$ and $\gamma' =_R \beta'$ such that:*

- *γ and γ' are, respectively, $\mathcal{S}$-*normalized basic term *and $\mathcal{S}'$-*normalized basic term.
- *There exists P' such that $P' \vdash_{\mathcal{A}_{FA_s}} \gamma \sim \gamma'$, instance$(P')$ is a sub-instance of instance(P) and P' contains less FA$_s$ rules than P.*

Proof. Let $\mathcal{S} = (\mathcal{K}, \mathcal{R}, \mathcal{E}, \mathcal{D})$. We prove the lemma by induction on the number of FA_s rules in P. If P has no FA_s application, then we have three cases:

- β and β' are identical, up to α-renaming. In that case, we can check that $\gamma \equiv \beta \downarrow_R$ and $\gamma' \equiv \beta' \downarrow_R$ satisfy the wanted properties.
- β and β' are, resp., a $\mathcal{S}$-encryption oracle call and a $\mathcal{S}'$-encryption oracle call. Since an $\mathcal{S}$-encryption oracle call is also a $\mathcal{S}$-normalized basic term, we conclude by taking $\gamma \equiv \beta$ and $\gamma' \equiv \beta'$.
- β and β' are, resp., a $\mathcal{S}$-decryption oracle call and a $\mathcal{S}'$-decryption oracle call. Similarly, a $\mathcal{S}$-decryption oracle call is also a $\mathcal{S}$-normalized basic term. We conclude by taking $\gamma \equiv \beta$ and $\gamma' \equiv \beta'$.

For the inductive case, β and β' must start with the same function symbol. Hence:

$$\beta \equiv f(\beta_1, \ldots, \beta_n) \qquad\qquad \beta' \equiv f(\beta'_1, \ldots, \beta'_n)$$

First, we check that $\beta_1, \ldots, \beta_n$ are $\mathcal{S}$-basic terms. Indeed, the only way that some β_i could not be a $\mathcal{S}$-basic term was if β was an $\mathcal{S}$-encryption oracle call or a $\mathcal{S}$-decryption oracle call. Then, f must be $\{\_\}_\_$ or $\mathsf{dec}(\_,\_)$:

- in the former case, $\beta \equiv \{\_\}_\_^{\mathsf{n_e}}$ where $\mathsf{n_e} \in \mathcal{R}$ and one of the β_i is equal to $\mathsf{n_e}$. Since β is a $\mathcal{S}$-basic term, we know that $\overline{\mathsf{fresh}}(\mathcal{R}; \mathsf{n_e})$. Contradiction.
- in the latter case, $\beta \equiv \mathsf{dec}(\_, \mathsf{sk}(\mathsf{n}))$ where $\mathsf{sk}(\mathsf{n}) \in \mathcal{K}$. Since β is a $\mathcal{S}$-basic term, we know that $\mathsf{nodec}(\mathcal{K}, \mathsf{sk}(\mathsf{n}))$. Contradiction.

Hence $\beta_1, \ldots, \beta_n$ are $\mathcal{S}$-basic terms. Similarly $\beta'_1, \ldots, \beta'_n$ are $\mathcal{S}'$-basic terms.

Using Lemma 1, we know that for every i, we can extract from P a proof of $Q_i \vdash_{\mathcal{A}_{\mathsf{FA_s}}} \beta_i \sim \beta'_i$. One can check that the procedure described in Lemma 1 is such that P has as many FA applications than all the $(Q_i)_i$ altogether. By induction hypothesis, let:

$$P_1 \vdash_{\mathcal{A}_{\mathsf{FA_s}}} \gamma_1 \sim \gamma'_1, \quad \ldots \quad , P_n \vdash_{\mathcal{A}_{\mathsf{FA_s}}} \gamma_n \sim \gamma'_n$$

be such that for every i, $\gamma_i =_R \beta_i, \gamma'_i =_R \beta'$, γ_i is a $\mathcal{S}$-normalized basic term and γ'_i is a $\mathcal{S}'$-normalized basic term, $\mathsf{instance}(P_i)$ is a sub-instance of $\mathsf{instance}(P)$ and P_i has less $\mathsf{FA_s}$ applications than Q_i. By Proposition 13, there exists a proof P' of:

$$P' \vdash_{\mathcal{A}_{\mathsf{FA_s}}} (\gamma_n)_{n \in N} \sim (\gamma'_n)_{n \in N}$$

such that $\mathsf{instance}(P')$ is a sub-instance of $\mathsf{instance}(P)$ and P' has as many $\mathsf{FA_s}$ applications than the $(P_i)_i$ altogether. Since P_i has less $\mathsf{FA_s}$ applications than Q_i, and since P has as many $\mathsf{FA_s}$ applications than all the $(Q_i)_i$ altogether, P' has less $\mathsf{FA_s}$ applications than P.

$f(\beta_1, \ldots, \beta_n)$ and $f(\beta'_1, \ldots, \beta'_n)$ can only have R_1 redexes at the top-level. If they have no R_1 redexes, then $f(\beta_1, \ldots, \beta_n)$ and $f(\beta'_1, \ldots, \beta'_n)$ are, respectively, $\mathcal{S}$-normalized basic term and $\mathcal{S}'$-normalized basic term. We conclude by applying FA_f:

$$\dfrac{\begin{array}{c}\vdots\ (P') \\ \gamma_1, \ldots, \gamma_n \sim \gamma'_1, \ldots, \gamma'_n\end{array}}{f(\gamma_1, \ldots, \gamma_n) \sim f(\gamma'_1, \ldots, \gamma'_n)}\ \mathsf{FA}_f$$

Therefore, assume $f(\beta_1, \ldots, \beta_n)$ or $f(\beta'_1, \ldots, \beta'_n)$ have a R_1 redex. We have several cases:

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

$$\dfrac{\langle \gamma_1\,,\, \gamma_2\rangle \sim \left\langle \gamma'_1\,,\, \gamma'_2\right\rangle}{\pi_1(\langle \gamma_1\,,\, \gamma_2\rangle) \sim \pi_1(\left\langle \gamma'_1\,,\, \gamma'_2\right\rangle)}\ \mathsf{FA}_{\pi_1}$$

  We look at the next rule in P':

– If it is CCA_2, then $\langle \gamma_1 , \gamma_2 \rangle$ and $\langle \gamma_1' , \gamma_2' \rangle$ are the same terms, up to α-renaming. We conclude by taking $\gamma \equiv \gamma_1$ and $\gamma' \equiv \gamma_1'$.
– Or it is a function application:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\ (Q)\\ \gamma_1, \gamma_2 \sim \gamma_1', \gamma_2'\end{array}}{\langle \gamma_1 , \gamma_2 \rangle \sim \langle \gamma_1' , \gamma_2' \rangle}\ \mathsf{FA}_{\langle , \rangle}}{\pi_1(\langle \gamma_1 , \gamma_2 \rangle) \sim \pi_1(\langle \gamma_1' , \gamma_2' \rangle)}\ \mathsf{FA}_{\pi_1}$$

Using Lemma 1, we extract from Q a proof $Q' \vdash_{\mathcal{A}_{\mathsf{FA}_s}} \gamma_1 \sim \gamma_1'$ no larger than Q. We conclude by taking $\gamma \equiv \gamma_1$ and $\gamma' \equiv \gamma_1'$:

$$\dfrac{\begin{array}{c}\vdots\ (Q')\\ \gamma_1 \sim \gamma_1'\end{array}}{\pi_1(\langle \gamma_1 , \gamma_2 \rangle) \sim \pi_1(\langle \gamma_1' , \gamma_2' \rangle)}\ R \tag{8}$$

- Only one side can be reduced by $\pi_i(\langle x_1 , x_2 \rangle) \to x_i$. Therefore the next rule applied in (P') must be CCA_2 (since the head function symbols differ). But in a CCA_2 application, we cannot have $\langle \_ , \_ \rangle \sim f'(\_)$ with $f' \neq \langle , \rangle$. Contradiction.
- Both sides can be reduced by $\mathsf{dec}(\{x\}^r_{\mathsf{pk(n)}}, \mathsf{sk(n)}) \to x$. Hence $n = 2$, $\gamma_1, \gamma_2 \equiv \{u\}^r_{\mathsf{pk(n)}}, \mathsf{sk(n)}$, $\gamma_1', \gamma_2' \equiv \{u'\}^{r'}_{\mathsf{pk(n')}}, \mathsf{sk(n')}$ and P' is of the form:

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

We look at the next rule applied on $\{u\}^r_{\mathsf{pk(n)}}, \_ \sim \{u'\}^{r'}_{\mathsf{pk(n')}}, \_$. If it is a function application then we have a shortcut using Lemma 1, as we did for (8). If it is CCA_2, we have two cases:
– $\{u\}^r_{\mathsf{pk}(n)}$ and $\{u'\}^{r'}_{\mathsf{pk}(n')}$ are the same terms, up to α-renaming. We conclude by taking $\gamma \equiv u$ and $\gamma' \equiv u'$.
– $\{u\}^r_{\mathsf{pk}(n)}$ and $\{u'\}^{r'}_{\mathsf{pk}(n')}$ are, respectively, a $\mathcal{S}$-encryption oracle call and a $\mathcal{S}'$-encryption oracle call. Then $\mathsf{sk(n)} \in \mathcal{K}$. Since $\gamma_2 \equiv \mathsf{sk(n)}$ and γ_2 is a $\mathcal{S}$-normalized basic term, we know that $\mathsf{nodec}(\mathcal{K}, \mathsf{sk(n)})$. Contradiction.
- Only one side can be reduced by $\mathsf{dec}(\{x\}^r_{\mathsf{pk(n)}}, \mathsf{sk(n)}) \to x$. Then (P') is necessarily of the form:

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

We look at the next rule applied to $\{t\}^r_{\mathsf{pk(n)}}$ and $\{t'\}^{r'}_{p'}$:
– If it is CCA_2, then $p' \equiv \mathsf{pk(n')}$. Therefore the right side can be reduced by $\mathsf{dec}(\{x\}^r_{\mathsf{pk(n')}}, \mathsf{sk(n')}) \to x$. Contradiction.
– 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 $p' \equiv \mathsf{pk(n')}$. Therefore the right side can be reduced by $\mathsf{dec}(\{x\}^r_{\mathsf{pk(n')}}, \mathsf{sk(n')}) \to x$. Contradiction.
- Both side can be reduced by $\mathsf{eq}(x, x) \to \mathsf{true}$. In this case the proof cut elimination is trivial.
- Only one side can be reduced by $\mathsf{eq}(x, x) \to \mathsf{true}$. Therefore we have a proof of the form:

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

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

D PROOF FORM

D.1 Early Proof Form

We showed in Lemma 8 that:

$$\mathfrak{F}\left((2\mathsf{Box} + R_{\square})^* \cdot \mathsf{CS}_{\square}^* \cdot \{\overline{\mathsf{BFA}}(b,b')\}^* \cdot \mathsf{UnF} \cdot {\mathsf{FA}_{\mathsf{s}}}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA}_2\right) \qquad (\mathcal{A}_{\succ})$$

is complete for $\mathfrak{F}((\mathsf{CS}+\mathsf{FA}+R+\mathsf{Dup}+\mathsf{CCA}_2)^*)$. Let us consider a proof P following this ordering. The only branching rule in $\mathcal{A}_{\succ}$ is the $\mathsf{CS}_{\square}$ rule, which has two premises. Hence after having completed all the $\mathsf{CS}_{\square}$ 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 CCA_2 axiom. To do so, we index each branch of the proof tree P by some $l \in L$ where L is a finite set of labels.

Definition 42. 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 annotating the branches in P, and for all $l \in \mathsf{label}(P)$, we let $\mathsf{instance}(P,l)$ be the CCA_2 instance used in branch l.

When applying the $\mathsf{CS}_{\square}$ rule on two boxed conditions $\boxed{b_1 \mid b_2}_b$ and $\boxed{b'_1 \mid b'_2}_{b'}$, we know that the sub-proofs of $b_1 \sim b'_1$ and $b_2 \sim b'_2$ lie in the fragment $\mathcal{A}_{\mathsf{CS}_{\square}}$. This gives us useful information on the shape of the terms. To use this, we define the $\mathsf{extract}_{\mathsf{l}}$ and $\mathsf{extract}_{\mathsf{r}}$ functions which allow to retrieve the left and right sub-proofs of, respectively, $b_1 \sim b'_1$ and $b_2 \sim b'_2$.

Definition 43. Given a proof $P \vdash \vec{u} \sim \vec{v}$ and a position h in the proof P such that:

$$P_{|h} = \dfrac{\vec{w}, b_1, (u_i)_i \sim \vec{w}', b'_1, (u'_i)_i \qquad \vec{w}, b_2, (v_i)_i \sim \vec{w}', b'_2, (v'_i)_i}{\vec{w}, \left(\text{if } \boxed{b_1 \mid b_2}_b \text{ then } u_i \text{ else } v_i\right)_i \sim \vec{w}', \left(\text{if } \boxed{b'_1 \mid b'_2}_{b'} \text{ then } u'_i \text{ else } v'_i\right)_i}\ \mathsf{CS}_{\square}$$

We let $\mathsf{extract}_{\mathsf{l}}(h,P)$ be proof of $b_1 \sim b'_1$ extracted from $P_{|h}$, and $\mathsf{extract}_{\mathsf{r}}(h,P)$ be proof of $b_2 \sim b'_2$ extracted from $P_{|h}$, using the Restr elimination procedure described in the proof of Lemma 7.

Using this, we define what are proofs in *early proof form*.

Definition 44. For all terms t, t' and proofs P such that $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_{\square}}} t \sim t'$, we say that P proof in *early proof form* if t and t' are of the following form:

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

where H is a set of positions in P such that:

- for all $h \in H$, the rule applied at position h in P is a $\mathsf{CS}_{\square}$ rule on the conditions:
$$\left(\boxed{b^{h_{\mathsf{l}}} \mid b^{h_{\mathsf{r}}}}_{b^h}, \boxed{b'^{h_{\mathsf{l}}} \mid b'^{h_{\mathsf{r}}}}_{b'^h}\right)$$
- Let $P^{h_{\mathsf{l}}} = \mathsf{extract}_{\mathsf{l}}(h,P)$ and $P^{h_{\mathsf{r}}} = \mathsf{extract}_{\mathsf{r}}(h,P)$, then:
$$P^{h_{\mathsf{l}}} \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_{\square}}} b^{h_{\mathsf{l}}} \sim b'^{h_{\mathsf{l}}} \qquad \text{and} \qquad P^{h_{\mathsf{r}}} \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_{\square}}} b^{h_{\mathsf{r}}} \sim b'^{h_{\mathsf{r}}}$$
and these two proofs are in *early proof form*.
- $\mathsf{label}(P^{h_{\mathsf{l}}}) \subseteq \mathsf{label}(P)$, and for all $l \in \mathsf{label}(P^{h_{\mathsf{l}}})$, $\mathsf{instance}(P^{h_{\mathsf{l}}}, l)$ is a sub-instance of $\mathsf{instance}(P,l)$ (same for $\mathsf{label}(P^{h_{\mathsf{r}}})$).
- For all $l \in \mathsf{label}(P)$, the proof of $u_l \sim u'_l$ extracted from P is in the fragment $\mathcal{A}_{\overline{\mathsf{BFA}}}$.

Moreover, we let $\mathsf{cs\text{-}pos}(P) \equiv H$.

Proposition 14. *For all terms t, t' and proofs P such that $P \vdash_{\mathcal{A}_{CS_\square}} t \sim t'$, there exists a labelling P' of P such that $P' \vdash^b_{\mathcal{A}_{CS_\square}} t \sim t'$ 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 $h \in \text{cs-pos}(P)$ and $\mathsf{x} \in \{\mathsf{l}, \mathsf{r}\}$, $\text{extract}_\mathsf{x}(h, P)$ is of size strictly smaller that P. We only need to do some α-renaming to have the labelling of the sub-proofs coincide.

Finally we can check that the resulting proof Q is such that for all $h \in \text{cs-pos}(Q), \mathsf{x} \in \{\mathsf{l}, \mathsf{r}\}$, for all $l \in \text{label}(\text{extract}_\mathsf{x}(h, P))$, the CCA_2 instance $\text{instance}(\text{extract}_\mathsf{x}(h, P), l)$ is a sub-instance of $\text{instance}(P, l)$. This follows from the fact that $\text{extract}_\mathsf{x}(h, P)$ is obtained through the Restr elimination procedure from P. □

We define below the set $\text{index}(P)$ of *all* positions of P where a $\text{CS}_\square$ rule is applied. This includes the set of positions $\text{cs-pos}(P)$, as well as the $\text{CS}_\square$ applications in sub-proofs of conditions $b \sim b'$. This set is naturally ordered using the prefix ordering on positions.

Definition 45. Let $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\text{CS}_\square}} t \sim t'$ in early proof form.

- We let $\text{index}(P)$ be the set of indices where $\text{CS}_\square$ rules occur in the proof P:
$$\text{index}(P) = \text{cs-pos}(P) \cup \bigcup_{h \in \text{cs-pos}(P)} \text{index}\left(\text{extract}_\mathsf{l}(h, P)\right) \cup \text{index}\left(\text{extract}_\mathsf{r}(h, P)\right)$$
- For all $h, h' \in \text{index}(t, P)$, we let $<$ be the ancestor relation on positions, defined by $h < h'$ if and only if h is a strict prefix of h'.
- For all $\mathsf{h} = h_\mathsf{x}$, where $h \in \text{index}(P)$ and $\mathsf{x} \in \{\mathsf{l}, \mathsf{r}\}$, we let $\text{cs-pos}_P(\mathsf{h}) = \text{cs-pos}(\text{extract}_\mathsf{x}(h, P))$. When there is no ambiguity on the proof P, we write $\text{cs-pos}(\mathsf{h})$ instead of $\text{cs-pos}_P(\mathsf{h})$.

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

Definition 46. Let $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\text{CS}_\square}} t \sim t'$ in early proof form. For all $l \in \text{label}(P)$, we define:
$$\text{h-branch}_P(l) = \{h_\mathsf{x} \mid h \in \text{index}(P) \wedge \mathsf{x} \in \{\mathsf{l}, \mathsf{r}\} \wedge l \in \text{label}(\text{extract}_\mathsf{x}(h, P))\} \cup \{\epsilon\}$$

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

We omit the proof P when there is no ambiguity, writing $\text{h-branch}(l)$ instead of $\text{h-branch}_P(l)$.

D.2 Shape of the Terms

For all proofs in $\mathcal{A}_\succ$, all R rewritings are done at the beginning of the proofs in the $(2\text{Box} + R_\square)^*$ part, and, afterwards, all rules (apart from Dup) only "peel off" terms by removing the top-most function symbol. Therefore the terms just after $(2\text{Box} + R_\square)^*$ characterize the shape of the subsequent proof. This observation is illustrated in Figure 13. Recall that for all $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\text{CS}_\square}} t \sim t'$ in early proof form, we have:

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

where for all $l \in \text{label}(P)$, the extraction from P of the sub-proof of $u_l \sim u'_l$ is in the fragment $\mathcal{A}_{\overline{\text{BFA}}}$. Therefore, for every l, u_l and u'_l are of the form:

$$u_l \equiv D_l\left[(\beta_{i,l})_{i \in I_l} \diamond (\gamma_{m,l})_{m \in M_l}\right] \qquad u'_l \equiv D_l\left[(\beta'_{i,l})_{i \in I_l} \diamond (\gamma'_{m,l})_{m \in M_l}\right]$$

where D_l is an if-context and:

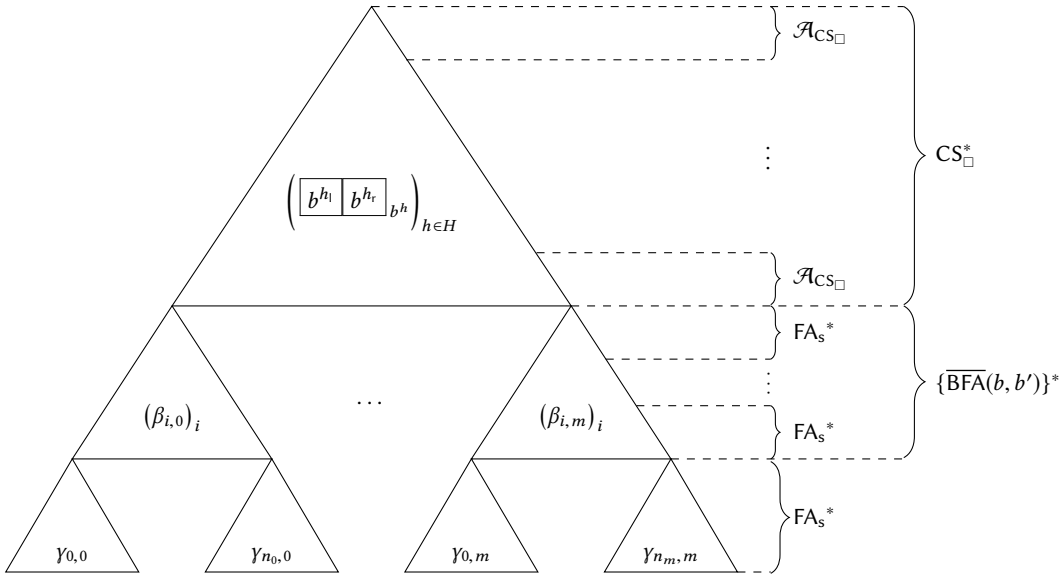


Fig. 13. The shape of the term is determined by the proof.

- $(\beta_{i,l})_{i\in I_l}$ and $(\beta'_{i,l})_{i\in I_l}$ are conditions such that the sub-proofs $(\beta_{i,l} \sim \beta'_{i,l})_{i\in I_l}$ extracted from P are in $\mathcal{A}_{\mathsf{FA_s}}$.
- $(\gamma_{j,l})_{j\in M_l}$ and $(\gamma'_{j,l})_{j\in M_l}$ are terms such that the sub-proofs $(\gamma_{j,l} \sim \gamma'_{j,l})_{j\in M_l}$ extracted from P are in $\mathcal{A}_{\mathsf{FA_s}}$.

Using these notation, we give some definitions:

Definition 47. Let $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\square}} t \sim t'$ in early proof form. For every $l \in \mathsf{label}(P)$, we let:

- $(b,b') \leq^{\epsilon,l}_{\mathsf{cs}\sim\mathsf{cs}} (t \sim t', P)$ if and only if there exists $h_0 \in \mathsf{cs\text{-}pos}(P)$ such that $b \equiv b^{h_0}$ and $b' \equiv b'^{h_0}$.
- $(\beta,\beta') \leq^{\epsilon,l}_{\mathsf{c}\sim\mathsf{c}} (t \sim t', P)$ if and only if there exists $i \in I_l$ such that $\beta \equiv \beta_{i,l}$ and $\beta' \equiv \beta'_{i,l}$
- $(\gamma,\gamma') \leq^{\epsilon,l}_{\mathsf{l}\sim\mathsf{l}} (t \sim t', P)$ if and only if there exists $m \in M_l$ such that $\gamma \equiv \gamma_{m,l}$ and $\gamma' \equiv \gamma'_{m,l}$.

Remark 8. Let $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\square}} t \sim t'$ in early proof form and $L = \mathsf{label}(P)$. Then:

$$t \equiv C\Big[\_ \diamond \big(D_l\big[(\beta)_{\beta \leq^{\epsilon,l}_{\mathsf{c}}(t,P)} \diamond (\gamma)_{\gamma \leq^{\epsilon,l}_{\mathsf{l}}(t,P)}\big]\big)_{l\in L}\Big]$$
$$\text{and} \quad t' \equiv C\Big[\_ \diamond \big(D_l\big[(\beta')_{\beta' \leq^{\epsilon,l}_{\mathsf{c}}(t',P)} \diamond (\gamma')_{\gamma' \leq^{\epsilon,l}_{\mathsf{l}}(t',P)}\big]\big)_{l\in L}\Big] \qquad \diamond$$

These relations allow use to obtain all *pairs* of terms appearing at the root level in P. We naturally define the asymmetric relation $\leq_{\mathsf{x}}$ from $\leq_{\mathsf{x}\sim\mathsf{x}}$:

Definition 48. Let $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\square}} t \sim t'$ in early proof form. For every $l \in \mathsf{label}(P)$ and $\mathsf{x} \in \{\mathsf{c},\mathsf{l},\mathsf{cs}\}$, we let:

$$\forall s.\, s \leq^{\epsilon,l}_{\mathsf{x}} (t,P) \qquad \text{if and only if} \qquad (s,\_) \leq^{\epsilon,l}_{\mathsf{x}\sim\mathsf{x}} (t \sim t', P)$$

Let $h \in \mathsf{index}(P)$ and $\mathsf{x} \in \{\mathsf{l},\mathsf{r}\}$. We lift these relations to h_{x} using the proof $\mathsf{extract}_{\mathsf{x}}(h,P)$.

Definition 49. Let $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\Box}} t \sim t'$ in early proof form. Let $l \in \mathsf{label}(P)$, $h \in \mathsf{index}(P)$, $\mathsf{x} \in \{\mathsf{l}, \mathsf{r}\}$ and b, b' be such that $\mathsf{extract}_{\mathsf{x}}(h, P)$ is a proof of $b \sim b'$. Then:

- For any $\Delta \in \{\mathsf{c}{\sim}\mathsf{c}, \mathsf{l}{\sim}\mathsf{l}, \mathsf{cs}{\sim}\mathsf{cs}\}$:

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

- For any $\Delta \in \{\mathsf{c}, \mathsf{l}, \mathsf{cs}\}$:

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

Remark 9. We extend these notations to proofs P such that $P \vdash^{\mathsf{b}}_{\mathcal{A}_>} t \sim t'$. Let P' be such that:

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

and $P' \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\Box}} t_0 \sim t_0'$, then $(s, s') \leq^{\mathsf{h}, l}_{\Delta} (t \sim t', P)$ if and only if $(s, s') \leq^{\mathsf{h}, l}_{\Delta} (t_0 \sim t_0', P')$ for any $\Delta \in \{\mathsf{c}{\sim}\mathsf{c}, \mathsf{l}{\sim}\mathsf{l}, \mathsf{cs}{\sim}\mathsf{cs}\}$. We have a similar definition for $\Delta \in \{\mathsf{c}, \mathsf{l}, \mathsf{cs}\}$. ◇

D.3 Proof Form and Normalized Proof Form

Definition 50. Let $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\Box}} t \sim t'$ in early proof form and $L = \mathsf{label}(P)$. Let $\mathcal{S}_l$ be the left trace of the CCA_2 instance used in branch l, and $\mathcal{S}'_l$ be the right trace of $\mathsf{instance}(P, l)$:

$$\mathcal{S}^P_l = \mathsf{l\text{-}trace}(\mathsf{instance}(P, l)) \qquad \mathcal{S}'^P_l = \mathsf{r\text{-}trace}(\mathsf{instance}(P, l))$$

We say that P is in *proof form* if and only if, for every $l \in L$:

- for every $h \in \mathsf{cs\text{-}pos}(P)$ and $x \in \{\mathsf{l}, \mathsf{r}\}$, the proof $\mathsf{extract}_{\mathsf{x}}(h, P)$ is in *proof forms*.
- $(\beta, \beta') \leq^{\epsilon, l}_{\mathsf{c}\sim\mathsf{c}} (t \sim t', P)$, β is a $\mathcal{S}$-*basic term* and β' is a $\mathcal{S}'$-*basic term*.
- $(\gamma, \gamma') \leq^{\epsilon, l}_{\mathsf{l}\sim\mathsf{l}} (t \sim t', P)$, γ is a $\mathcal{S}$-*basic term* and γ' is a $\mathcal{S}'$-*basic term*.

We obtain the definition of *normalized proof form* by replacing, in the definition above, *basic term* by *normalized basic term*, and *proof form* by *normalized proof form*.

We write $P \vdash^{\mathsf{npf}} t \sim t'$ whenever P is a proof of $t \sim t'$ in normalized proof form.

Let $P \vdash^{\mathsf{npf}} t \sim t'$, we already defined the set of conditions $\leq^{\mathsf{h}, l}_{\mathsf{c}} (t, P)$ 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 conditions are normalized basic condition. Similarly the set of leaf terms $\leq^{\mathsf{h}, l}_{\mathsf{l}} (t, P)$ 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 $\leq^{\mathsf{h}, l}_{\mathsf{c}} (t, P) \cup \leq^{\mathsf{h}, l}_{\mathsf{l}} (t, P)$.

Definition 51. For every $P \vdash^{\mathsf{npf}} t \sim t'$, for every term s, $s \leq^{\mathsf{h}, l}_{\mathsf{bt}} (t, P)$ if and only if there exists $u (\leq^{\mathsf{h}, l}_{\mathsf{c}} \cup \leq^{\mathsf{h}, l}_{\mathsf{l}})(t, P)$ such that $s \leq^{\mathcal{S}_l}_{\mathsf{bt}} u$.

D.4 Restriction to Proofs in Normalized Proof Form

Definition 52. We let $\overline{\mathrm{CCA}_2}$ be the restriction of CCA_2 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 15. *The following strategy is complete for* $\mathfrak{F}((CS + FA + R + Dup + \mathrm{CCA}_2)^*)$*:*

$$\mathfrak{F}((2Box + R_\Box)^* \cdot CS^*_\Box \cdot \{\overline{BFA}(b, b')\}^* \cdot UnF \cdot FA_s{}^* \cdot Dup^* \cdot \overline{\mathrm{CCA}_2})$$

Proof. By Lemma 8, the following strategy is complete for $\mathfrak{F}(\mathsf{CS} + \mathsf{FA} + R + \mathsf{Dup} + \text{CCA}_2)$:

$$\mathfrak{F}((2\mathsf{Box} + R_{\Box})^* \cdot \mathsf{CS}_{\Box}^* \cdot \{\overline{\mathsf{BFA}}(b, b')\}^* \cdot \mathsf{UnF} \cdot {\mathsf{FA}_s}^* \cdot \mathsf{Dup}^* \cdot \text{CCA}_2) \qquad (\mathcal{A}_{>})$$

For every proof $P \vdash^{\mathsf{b}} t \sim t'$ in this fragment, we let $L^P = \mathsf{label}(P)$ the set of branch indices of P. Moreover, we let $\mathcal{S}_l^P = (\mathcal{K}_l^P, \mathcal{R}_l^P, \mathcal{E}_l^P, \mathcal{D}_l^P)$ be the left trace of the CCA$_2$ instance of branch l, i.e. $\mathcal{S}_l^P = \mathsf{l\text{-}trace}(\mathsf{instance}(P, l))$. Finally, we define the order $<_P^l$ as follows: for all $u, u' \in \mathcal{E}_l^P \cup \mathcal{D}_l^P$, we let $u <_P^l u'$ hold if u is a strict subterm of u'.

We are going to show that for every proof P of $t \sim t'$ in $\mathcal{A}_{>}$, there exists a proof Q of $t \sim t'$ such that for every $l \in \mathsf{label}(Q)$, $\mathcal{E}_l^Q$ and $\mathcal{D}_l^Q$ are sets of, respectively, $\mathcal{S}_l^Q$-encryption oracle calls and $\mathcal{S}_l^Q$-decryption oracle calls, and the right part of Q and P are the same. We prove this by induction on the number of elements of $\bigcup_l \mathcal{E}_l^P \cup \mathcal{D}_l^P$ that are not $\mathcal{S}_l^P$-encryption oracle calls or $\mathcal{S}_l^P$-decryption oracle calls.

Let P be a proof of $t \sim t'$, $l \in L^P$ and let u minimal for $<_P^l$ which is not a $\mathcal{S}_l^P$-encryption oracle call or a $\mathcal{S}_l^P$-decryption oracle call. We have two cases:

- If $u \in \mathcal{E}_l^P$ is an encryption. We know that $u \equiv \{m\}_{\mathsf{pk}}^{\mathsf{n}_r}$ where the corresponding secret key sk is in $\mathcal{K}_l^P$. Let $(\alpha_k)_k$ be $\mathcal{E}_l^P \cap \mathsf{st}(m)$, and $(\mathsf{dec}_n)_n$ be $\mathcal{D}_l^P \cap \mathsf{st}(m)$. Let C be the smallest context such that:
$$m \equiv C[(\alpha_k)_k, (\mathsf{dec}_n)_n]$$
From the definition of CCA$_2$, we know that $C[]$ does not contain the $\mathbf{0}(\_)$ function symbol. 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 $m_0 =_R m$. We are going to show that m_0 is a $\mathcal{S}_l^P$-simple term. Since $C[]$ does not contain the $\mathbf{0}(\_)$ function symbol, we know that the contexts $(B_i[])_i$ and $(U_m[])_m$ do not contain $\mathbf{0}(\_)$. By minimality 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 every k, α_k is of the shape $\alpha_k \equiv \{\_\}_{\mathsf{pk}_k}^{\mathsf{n}_k}$. For every n, we let sk_n be the secret key used in 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, we can only have a redex at one of the encryption. More precisely, there must exist some k such that $\mathsf{dec}(\{[]_k\}_{\mathsf{pk}_k}^{\mathsf{n}_k}, \mathsf{sk}_k)$ is a subterm of $\tilde{m}$. By consequence, sk_k is a subterm of $B_i[]$. But since $\mathsf{sk}_k \in \mathcal{K}_l^P$, we know that $st(B_i)$ does not contain sk_k (sk_k can only appear in $\mathcal{D}_l^P$). Contradiction. Hence $\tilde{m}$ is in R-normal form, which implies that $(B_i[(\alpha_k)_k, (\mathsf{dec}_n)_n])_i$ are $\mathcal{S}_l^P$-normalized basic terms. Similarly we prove that $(U_m[(\alpha_k)_k, (\mathsf{dec}_n)_n])_m$ are $\mathcal{S}_l^P$-normalized basic terms. Hence m_0 is a $\mathcal{S}_l^P$-normalized simple term.
We then rewrite, using R, every occurrence of $\{m\}_{\mathsf{sk}}^{\mathsf{n}_r}$ by $\{m_0\}_{\mathsf{sk}}^{\mathsf{n}_r}$ in branch l of P. We check that this yields a valid proof Q. The only difficulty lies in making sure that the side-conditions of the CCA$_2$ application for the decryptions still holds. Their is one subtlety here: an encryption $\alpha \equiv \{m_\alpha\}_{\mathsf{pk}}^{\mathsf{n}_\alpha}$ must be guarded in some $\mathsf{dec}(u_0, \mathsf{sk})$ iff it appears directly in u_0. This side-condition is preserved as it is stable by any R rewriting (hence in particular the rewriting of $\{m\}_{\mathsf{sk}}^{\mathsf{n}_r}$ into $\{m_0\}_{\mathsf{sk}}^{\mathsf{n}_r}$).
We can check that the resulting proof Q of $t \sim t'$ has 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. 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 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 prove that $\vdash^{\mathsf{npf}}$ is complete.

Lemma 16. *The restriction of the fragment $\mathcal{A}_\succ$ to formulas provable in $\vdash^{npf}$ is complete for:*

$$\mathfrak{F}((CS + FA + R + Dup + CCA_2)^*)$$

Proof. Using Lemma 15, the following strategy is complete for $\mathfrak{F}((\mathsf{CS} + \mathsf{FA} + R + \mathsf{Dup} + \overline{\mathsf{CCA_2}})^*)$:

$$\mathfrak{F}((2\mathsf{Box} + R_\Box)^* \cdot \mathsf{CS}_\Box^* \cdot \{\overline{\mathsf{BFA}}(b, b')\}^* \cdot \mathsf{UnF} \cdot {\mathsf{FA}_\mathsf{s}}^* \cdot \mathsf{Dup}^* \cdot \overline{\mathsf{CCA_2}})$$

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

By Proposition 14, we know that there exists P such that $P \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\Box}} t \sim t'$. Let $h \in \mathsf{index}(P), \mathsf{x} \in \{\mathsf{l}, \mathsf{r}\}$, $\mathsf{h} = h_\mathsf{x}$, and let $b^\mathsf{h}, b'^\mathsf{h}$ be such that $\mathsf{extract}_\mathsf{x}(h, P) \vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\Box}} b^\mathsf{h} \sim b'^\mathsf{h}$. First, we prove that we can ensure that for every $(\beta, \beta') (\leq^{\mathsf{h},l}_{\mathsf{c}\sim\mathsf{c}} \cup \leq^{\mathsf{h},l}_{\mathsf{l}\sim\mathsf{l}})(t \sim t', P)$, the terms β and β' are, respectively, $\mathcal{S}_l^P$-basic term and $\mathcal{S}_l'^P$-basic terms. We know that:

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

where B and B' are if-free and $\vec{w}, (\alpha_j)_j, (\mathsf{dec}_k)_k \sim \vec{w}, (\alpha'_j)_j, (\mathsf{dec}'_k)_k$ is a sub-instance of $\mathsf{instance}(P, l)$.

Since this is a sub-instance, we know that $\mathsf{fresh}(\mathcal{R}_l^P; \vec{w})$ and $\mathsf{nodec}(\mathcal{K}_l^P, \vec{w})$. Moreover, using the fact that $\mathsf{instance}(P, l)$ is a $\overline{\mathsf{CCA_2}}$ instance, we know that $(\alpha_j)_j$ and $(\mathsf{dec}_k)_k$ are, respectively, $\mathcal{S}_l^P$-encryption oracle calls and $\mathcal{S}_l^P$-decryption oracle calls. Therefore if $\vec{w}$ is if-free then β is a $\mathcal{S}_l^P$-basic term.

Assume that $\vec{w}$ is not if-free. 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 $\vdash^{\mathsf{b}}_{\mathcal{A}_{\mathsf{CS}_\Box}}$, 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 ...).

We replaced B by three terms $B_c, B_e[B_0], B_e[B_1]$ containing strictly less if then else applications. Hence, by induction, we ensure that all such contexts B are if-free, by repeating the proof rewriting above. We deduce that there exists a proof Q of $t \sim t'$ where Q is in proof form.

To obtain a *normalized* proof form, we only have to apply the Lemma 14 to all branches l, and to commute the new R rewriting to the bottom of the proof. □

E PROPERTIES OF NORMALIZED BASIC TERMS

E.1 Basic Term Extraction

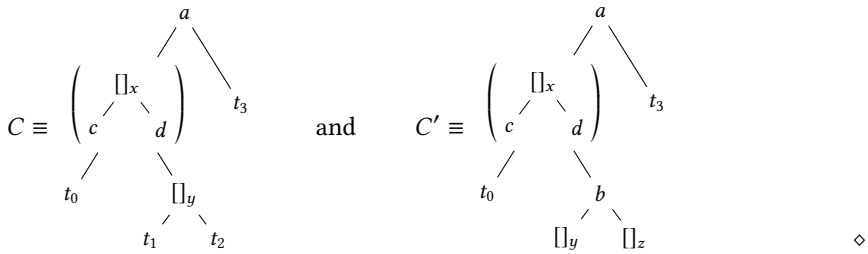
Definition 53. We call a *condition context* a context $C[]_{\vec{x}}$ such that all holes appear in the condition part of an if_then_else_. Formally, for every position p, if $C_{|p}$ is a hole $[]_x$ then $p = p'.0$ and there

exist u and v such that:

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

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

Example 19. We give an example of a condition context C with two holes on the left, and a context C' which is *not* a condition context on the right (since it has holes in leaf positions):



◇

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

Lemma 17. *For all $P \vdash^{npf} t \sim t'$, for all h, l and $\beta, \beta' \leq_{bt}^{h,l} (t, P)$, there exists an almost condition context $\tilde{\beta}'[]$ such that:*

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

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

$$\beta_0 \equiv \quad \text{eq}(\{\text{if } b \text{ then } s \text{ else } t\}^{\mathsf{n}_r}_{\text{pk(n)}}, 0) \qquad\qquad \beta_1 \equiv \text{eq}(\{\text{if } \beta_0^0 \text{ then } u \text{ else } u\}^{\mathsf{n}'_r}_{\text{pk(n)}}, 0)$$

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

where $\beta_0^0, \beta_0^1 \notin \text{cond-st}(u \downarrow_R)$ and $s \neq_R t$. Then $\beta_0^0, \beta_0^1 \notin \text{cond-st}(\beta_1 \downarrow_R)$, because β_0^0 disappear using the rule if x then y else $y \to y$ in R. Hence, Lemma 17 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 CCA_2 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 $\overline{\text{leave-st}}(\cdot)$ and $\overline{\text{cond-st}}(\cdot)$ of, respectively, $\text{leave-st}(\cdot \downarrow_R)$ and $\text{cond-st}(\cdot \downarrow_R)$. Then, we show that Lemma 17 holds for $\overline{\text{leave-st}}(\cdot)$ and $\overline{\text{cond-st}}(\cdot)$.

Definition 54. We define the function $\overline{\text{leave-st}}$ from the set of terms to the set of if-free terms in R-normal form:

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

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

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

$$\overline{\text{cond-st}}(u_0, \ldots, u_n) = \cup_{i \leq n} \overline{\text{cond-st}}(u_i) \qquad \overline{\text{cond-st}}(f(\vec{u})) = \overline{\text{cond-st}}(\vec{u}) \quad (\forall f \in \mathcal{F}_{\setminus \text{if}} \cup \mathcal{N})$$

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

Remark 10. There are multiples over-approximations. For example, assuming that b, u, v, w, s, t are if-free terms in R-normal forms, there in an over-approximation in the if_then_else_ case:

$$\mathsf{leave\text{-}st}\Big(\Big(\ \overset{b}{\underset{v}{u}}\ \ \overset{}{\underset{w}{b}}\ \Big)\downarrow_R\Big) = \{u, w\} \qquad \overline{\mathsf{leave\text{-}st}}\Big(\ \overset{b}{\underset{v}{u}}\ \ \overset{}{\underset{w}{b}}\ \Big) = \{u, v, w\}$$

There in another over-approximation in the f case:

$$\mathsf{leave\text{-}st}\Big(f\Big(\ \overset{b}{\underset{v}{u}}\ ,\ \overset{b}{\underset{t}{s}}\ \Big)\downarrow_R\Big) = \{f(u,s), f(v,t)\}$$

$$\overline{\mathsf{leave\text{-}st}}\Big(f\Big(\ \overset{b}{\underset{v}{u}}\ ,\ \overset{b}{\underset{t}{s}}\ \Big)\Big) = \{f(u,s), f(u,t), f(v,s), f(v,t)\}$$

$\overline{\mathsf{cond\text{-}st}}()$ inherits from $\overline{\mathsf{leave\text{-}st}}()$ over-approximations, and also over-approximates if_then_else_: indeed, while $\mathsf{cond\text{-}st}(t\downarrow_R)$ never contains conditions which are spurious in t, the set $\overline{\mathsf{cond\text{-}st}}(t)$ may. E.g.:

$$\mathsf{cond\text{-}st}\Big(\ \overset{b}{\underset{u}{u}}\ \downarrow_R\Big) = \emptyset \qquad \overline{\mathsf{cond\text{-}st}}\Big(\ \overset{b}{\underset{u}{u}}\ \Big) = \{b\} \qquad \diamond$$

$\overline{\mathsf{leave\text{-}st}}(\cdot)$ is a sound over-approximation of $\mathsf{leave\text{-}st}(\cdot\downarrow_R)$. Moreover, $\overline{\mathsf{leave\text{-}st}}(\cdot)$ and $\mathsf{leave\text{-}st}(\cdot\downarrow_R)$ coincides on terms in R-normal form. The same properties hold for $\overline{\mathsf{leave\text{-}st}}(\cdot)$ and $\mathsf{leave\text{-}st}(\cdot\downarrow_R)$.

Proposition 15. *$\overline{\mathsf{leave\text{-}st}}$ and $\overline{\mathsf{cond\text{-}st}}$ are sound over-approximations:*

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

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

- $u \equiv \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ t)\ \mathsf{else}\ w$ and $v \equiv \mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ w$ then:
$$\begin{aligned}\overline{\mathsf{leave\text{-}st}}(u) &= \overline{\mathsf{leave\text{-}st}}(s) \cup \overline{\mathsf{leave\text{-}st}}(t) \cup \overline{\mathsf{leave\text{-}st}}(w)\\ &\supseteq \overline{\mathsf{leave\text{-}st}}(s) \cup \overline{\mathsf{leave\text{-}st}}(w)\\ &= \overline{\mathsf{leave\text{-}st}}(v)\end{aligned}$$
- $u \equiv \mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ s$ and $v \equiv s$ then:
$$\overline{\mathsf{leave\text{-}st}}(u) = \overline{\mathsf{leave\text{-}st}}(s) = \overline{\mathsf{leave\text{-}st}}(v)$$
- $u \equiv \mathsf{if}\ (\mathsf{if}\ b\ \mathsf{then}\ a\ \mathsf{else}\ c)\ \mathsf{then}\ s\ \mathsf{else}\ t$ and $v \equiv \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ s\ \mathsf{else}\ t)\ \mathsf{else}\ (\mathsf{if}\ c\ \mathsf{then}\ s\ \mathsf{else}\ t)$:
$$\overline{\mathsf{leave\text{-}st}}(u) = \overline{\mathsf{leave\text{-}st}}(s) \cup \overline{\mathsf{leave\text{-}st}}(t) = \overline{\mathsf{leave\text{-}st}}(v)$$
- $u \equiv \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{if}\ a\ \mathsf{then}\ s\ \mathsf{else}\ t)\ \mathsf{else}\ w$ and $v \equiv \mathsf{if}\ a\ \mathsf{then}\ (\mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ w)\ \mathsf{else}\ (\mathsf{if}\ b\ \mathsf{then}\ t\ \mathsf{else}\ w)$:
$$\overline{\mathsf{leave\text{-}st}}(u) = \overline{\mathsf{leave\text{-}st}}(s) \cup \overline{\mathsf{leave\text{-}st}}(t) \cup \overline{\mathsf{leave\text{-}st}}(w) = \overline{\mathsf{leave\text{-}st}}(v)$$

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

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

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

Similarly, we show by induction on $\rightarrow_R^*$ that for all $u \rightarrow_R^* u'$, $\overline{\mathsf{cond\text{-}st}}(u) \supseteq \overline{\mathsf{cond\text{-}st}}(u')$. If $u \equiv u'$ this is immediate, assume that $u \rightarrow_R v \rightarrow_R^* u'$. By induction hypothesis we know that $\overline{\mathsf{leave\text{-}st}}(v) \supseteq \overline{\mathsf{leave\text{-}st}}(u')$. Therefore, we only need to show that $\overline{\mathsf{leave\text{-}st}}(u) \supseteq \overline{\mathsf{leave\text{-}st}}(v)$. We do a case disjunction on the rule applied at $u \rightarrow_R v$ (we omit the redundant or obvious cases):

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

$$\begin{aligned}
\overline{\mathsf{cond\text{-}st}}(u) &= \overline{\mathsf{cond\text{-}st}}(s, t, w) \cup \overline{\mathsf{cond\text{-}st}}(b) \cup \overline{\mathsf{leave\text{-}st}}(b) \\
&\supseteq \overline{\mathsf{cond\text{-}st}}(s, w) \cup \overline{\mathsf{cond\text{-}st}}(b) \cup \overline{\mathsf{leave\text{-}st}}(b) \\
&\supseteq \overline{\mathsf{cond\text{-}st}}(v)
\end{aligned}$$

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

$$\overline{\mathsf{cond\text{-}st}}(u) = \overline{\mathsf{cond\text{-}st}}(b, a, c, s, t) \cup \overline{\mathsf{leave\text{-}st}}(b, a, c) = \overline{\mathsf{cond\text{-}st}}(v)$$

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

$$\overline{\mathsf{cond\text{-}st}}(u) = \overline{\mathsf{cond\text{-}st}}(b, \vec{w}, \vec{s}, \vec{t}) \cup \overline{\mathsf{leave\text{-}st}}(b) = \overline{\mathsf{cond\text{-}st}}(v)$$

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

Corollary 1. *For every term u, $\overline{\mathsf{leave\text{-}st}}(u) \supseteq \mathsf{leave\text{-}st}(u \downarrow_R)$ and $\overline{\mathsf{cond\text{-}st}}(u) \supseteq \mathsf{cond\text{-}st}(u \downarrow_R)$.*

Let us show the following helpful propositions:

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 we have $\mathcal{S}_l$-normalized basic terms $B[\vec{w}, (\alpha^j)_j, (\delta^k)_k]$ and $B[\vec{w}, (\alpha'^j)_j, (\delta'^k)_k]$ such that:

$$\beta \equiv B[\vec{w}, (\alpha^j)_j, (\delta^k)_k] \qquad \beta' \equiv B[\vec{w}, (\alpha'^j)_j, (\delta'^k)_k]$$

$$\forall j, \ \overline{\mathsf{leave\text{-}st}}(\alpha^j) \cap \overline{\mathsf{leave\text{-}st}}(\alpha'^j) \neq \emptyset \qquad \forall k, \ \overline{\mathsf{leave\text{-}st}}(\delta^k) \cap \overline{\mathsf{leave\text{-}st}}(\delta'^k) \neq \emptyset$$

Proof. We have $\mathcal{S}_l$-normalized basic terms $B[\vec{w}, (\alpha^j)_j, (\delta^k)_k]$ and $D[\vec{w}', (\alpha'^j)_j, (\delta'^k)_k]$ such that:

$$\beta \equiv B[\vec{w}, (\alpha^j)_j, (\delta^k)_k] \qquad \beta' \equiv D[\vec{w}', (\alpha'^j)_j, (\delta'^k)_k]$$

Since β, β' are $\mathcal{S}_l$-normalized basic terms, we know that:

$$B[\vec{w}, (\{[]_j\}_{\_})_j, (\mathsf{dec}([]_k, \_))_k] \qquad D[\vec{w}', (\{[]_j\}_{\_})_j, (\mathsf{dec}([]_k, \_))_k]$$

are in R-normal form, that $B[], D[], \vec{w}, \vec{w}'$ are if-free and that $B[], D[]$ do not contain $\mathbf{0}(\_)$. Hence:

$$\overline{\mathsf{leave\text{-}st}}(\beta) = \left\{ B[\vec{w}, (a^j)_j, (d^k)_k] \mid \forall j, a^j \in \overline{\mathsf{leave\text{-}st}}(\alpha^j) \ \wedge \ \forall k, d^k \in \overline{\mathsf{leave\text{-}st}}(\delta^k) \right\}$$

$$\overline{\mathsf{leave\text{-}st}}(\beta') = \left\{ D[\vec{w}', (a'^j)_j, (d'^k)_k] \mid \forall j, a'^j \in \overline{\mathsf{leave\text{-}st}}(\alpha'^j) \ \wedge \ \forall k, d'^k \in \overline{\mathsf{leave\text{-}st}}(\delta'^k) \right\}$$

Similarly to what we did in the proof of Lemma 2, we prove that we can assume that $B[] \equiv D[]$ by induction on the number of hole positions in $B[]$ or $D[]$ such that $(B[])_{|p}$ differs from $(D[])_{|p}$ (modulo hole renaming). Knowing that $B[] \equiv D[]$, it is then straightforward to show that:

$$\vec{w} \equiv \vec{w}' \qquad \forall j, \ \overline{\mathsf{leave\text{-}st}}(\alpha^j) \cap \overline{\mathsf{leave\text{-}st}}(\alpha'^j) \neq \emptyset \qquad \forall k, \ \overline{\mathsf{leave\text{-}st}}(\delta^k) \cap \overline{\mathsf{leave\text{-}st}}(\delta'^k) \neq \emptyset$$

The base case is trivial, let us prove the inductive case. Let $B[\vec{w}, (a^j)_j, (d^k)_k]$ and $D[\vec{w}', (a'^j)_j, (d'^k)_k]$ be such that:

$$\forall j, k.\ a^j \in \overline{\mathsf{leave\text{-}st}}(\alpha^j) \wedge d^k \in \overline{\mathsf{leave\text{-}st}}(\delta^k) \qquad \forall j, k.\ a'^j \in \overline{\mathsf{leave\text{-}st}}(\alpha'^j) \wedge d'^k \in \overline{\mathsf{leave\text{-}st}}(\delta'^k)$$

and:

$$B[\vec{w}, (a^j)_j, (d^k)_k] \equiv D[\vec{w}', (a'^j)_j, (d'^k)_k] \in \overline{\mathsf{leave\text{-}st}}(\beta) \cap \overline{\mathsf{leave\text{-}st}}(\beta')$$

First, observe that if a position p is valid in both $B[]$ and $D[]$, and is not a hole in both contexts, then $B[]$ and $D[]$ coincide on p.

Let p be the position of a hole in $B[]$ such that p is a valid position in $D[]$, but not a hole (if p is not valid in $D[]$, invert $B[]$ and $D[]$). We then have three cases depending on $(B[])_{|p}$:

- B contains a hole $[]_x$ at position p such that $\beta_{|p} \in \vec{w}$. Then let $\tilde{D}$ be the context D 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 $\tilde{D}$ 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 CCA_2 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 D, 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 D, 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 8.

Proposition 17. *For all $\mathcal{S}_l$-normalized basic terms β, β', we have $\beta \equiv \beta'$ whenever:*

$$\overline{\mathsf{leave\text{-}st}}(\beta) \cap \overline{\mathsf{leave\text{-}st}}(\beta') \neq \emptyset$$

Proof. We show this by induction on $|\beta| + |\beta'|$. Using Proposition 16 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:

$$\beta \equiv B[\vec{w}, (\alpha^j)_j, (\delta^k)_k] \qquad \beta' \equiv B[\vec{w}, (\alpha'^j)_j, (\delta'^k)_k]$$

$$\forall j, \ \overline{\mathsf{leave\text{-}st}}(\alpha^j) \cap \overline{\mathsf{leave\text{-}st}}(\alpha'^j) \neq \emptyset \qquad \forall k, \ \overline{\mathsf{leave\text{-}st}}(\delta^k) \cap \overline{\mathsf{leave\text{-}st}}(\delta'^k) \neq \emptyset$$

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 $\alpha^j \equiv \{m\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}$ and $\alpha'^j \equiv \{m'\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}$. 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

$$\delta^k \equiv C\left[\vec{g} \diamond (s_i)_{i \leq p}\right] \qquad\qquad \delta'^k \equiv C'\left[\vec{g}' \diamond (s'_i)_{i \leq p'}\right]$$

where:

- There exists contexts u, u', if-free and in R-normal form, such that:

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

$$\forall g \in \vec{g},\ g \equiv \mathsf{eq}(u[(\alpha_j)_j, (\mathsf{dec}_k)_k], \alpha_g) \text{ where } \alpha_g \in (\alpha_j)_j$$

$$\forall i < p',\ s'_i \equiv \mathbf{0}(\mathsf{dec}(u'[(\alpha'_j)_j, (\mathsf{dec}'_k)_k], \mathsf{sk}')) \qquad s'_p \equiv \mathsf{dec}(u'[(\alpha'_j)_j, (\mathsf{dec}'_k)_k], \mathsf{sk}')$$

$$\forall g \in \vec{g}',\ g \equiv \mathsf{eq}(u'[(\alpha'_j)_j, (\mathsf{dec}'_k)_k], \alpha'_g) \text{ where } \alpha'_g \in (\alpha'_j)_j$$

- $(\alpha_j)_j, (\alpha'_j)_j$ 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 CCA_2 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 n appear directly in s_p (resp. s'_p). 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 $R^{\mathcal{K}}_{\mathrm{CCA}_2^a}$), we know that $\delta^k \equiv \delta'^k$. □

Corollary 2. *For all $P \vdash^{npf} t \sim t'$, for all h, l:*

(i) *for all $\beta, \beta' \leq_c^{h,l} (t, P)$ 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 $\gamma, \gamma' \leq_l^{h,l} (t, P)$ 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 $\beta \leq_c^{h,l} (t, P)$, $\gamma \leq_l^{h,l} (t, P)$ if* $\mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \mathsf{leave\text{-}st}(\gamma \downarrow_R) \neq \emptyset$ *then* $\beta \equiv \gamma$.

We can now show the following lemma, which subsumes Lemma 17:

Lemma 18. *For all $P \vdash^{npf} t \sim t'$, for all h, l and $\beta, \beta' \leq_{bt}^{h,l} (t, P)$, there exists an almost condition context $\tilde{\beta}'[]$ such that:*

$$\beta' \equiv \tilde{\beta}'[\beta] \quad \text{and} \quad \mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \overline{\mathsf{cond\text{-}st}}(\tilde{\beta}'[]) = \emptyset$$

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 condition context $u_c[]$ such that:

$$u \equiv u_c[\beta] \qquad \mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \overline{\mathsf{cond\text{-}st}}(u_c[]) = \emptyset \qquad \overline{\mathsf{leave\text{-}st}}(\vec{u}_c) = \overline{\mathsf{leave\text{-}st}}(\vec{u})$$

Moreover if u is a $\mathcal{S}_l$-normalized basic term then there exists an *almost* condition context $u_d[]$ such that:

$$u \equiv u_d[\beta] \qquad \mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \left(\overline{\mathsf{cond\text{-}st}}(u_d[]) \cup \overline{\mathsf{leave\text{-}st}}(u_d[])\right) = \emptyset$$

- **Normalized Simple Term:** Let $u \equiv C[\vec{b} \diamond \vec{s}]$, where $\vec{b}$ are $\mathcal{S}_l$-normalized basic conditions 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:

$$\overline{\mathsf{leave\text{-}st}}(\vec{s}_c[]) = \overline{\mathsf{leave\text{-}st}}(\vec{s}) \qquad \mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \left(\overline{\mathsf{cond\text{-}st}}(\vec{b}_d[], \vec{s}_c[]) \cup \overline{\mathsf{leave\text{-}st}}(\vec{b}_d[])\right) = \emptyset$$

  Moreover:

$$\overline{\mathsf{cond\text{-}st}}(C[\vec{b}_d[] \diamond \vec{s}_c[]]) = \overline{\mathsf{cond\text{-}st}}(\vec{b}_d[], \vec{s}_c[]) \cup \overline{\mathsf{leave\text{-}st}}(\vec{b}_d[])$$

$$\overline{\mathsf{leave\text{-}st}}(C[\vec{b}_d[] \diamond \vec{s}_c[]]) = \overline{\mathsf{leave\text{-}st}}(\vec{s}_c[]) = \overline{\mathsf{leave\text{-}st}}(\vec{s}) = \overline{\mathsf{leave\text{-}st}}(C[\vec{b} \diamond \vec{s}])$$

  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:

$$\overline{\mathsf{leave\text{-}st}}\big((\alpha^i_c)_i, (\mathsf{dec}^j_c)_i\big) = \overline{\mathsf{leave\text{-}st}}\big((\alpha^i)_i, (\mathsf{dec}^j)_i\big)$$

  and since B and $\vec{w}$ are if-free, one can check that:

$$\overline{\mathsf{leave\text{-}st}}\big(B[\vec{w}, (\alpha^i_c)_i, (\mathsf{dec}^j_c)_j]\big) = \overline{\mathsf{leave\text{-}st}}\big(B[\vec{w}, (\alpha^i)_i, (\mathsf{dec}^j)_j]\big)$$

  It is then immediate to check that $u_c \equiv B[\vec{w}, (\alpha^i_c)_i, (\mathsf{dec}^j_c)_j]$ satisfies the wanted properties. It remains to construct the context $u_d[]$. If $\mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \overline{\mathsf{leave\text{-}st}}(u) = \emptyset$ then $u_d \equiv u_c$ satisfies the wanted properties. Otherwise using Proposition 17 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. □

E.2 Well-Nested Sets

Definition 55. 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 56. We let *well-nested* be the smallest relation between sets $(\mathcal{C}, \mathcal{D})$ of flat terms such that:

(a) $(\mathcal{C}, \mathcal{D})$ is well-nested if for every $C_0[\vec{a}_0 \diamond \vec{b}_0] \in \mathcal{C}$:

$$\forall C[\vec{a} \diamond \vec{b}] \in \mathcal{C} \cup \mathcal{D}, \quad \vec{b}_0 \cap \vec{a} = \emptyset$$

(b) $(\mathcal{C}, \mathcal{D})$ is well-nested if for every $\beta_0 \equiv C_0[\vec{a}_0 \diamond \vec{b}_0] \in \mathcal{C}$:

(i) For all $\beta \equiv C[\vec{a} \diamond \vec{b}] \in \mathcal{C} \cup \mathcal{D}$, there exist two if-contexts C'_β, C''_β such that:

$$\beta =_R \mathsf{if}\ \beta_0\ \mathsf{then}\ C'_\beta[\vec{a}'_\beta \diamond \vec{b}'_\beta]\ \mathsf{else}\ C''_\beta[\vec{a}''_\beta \diamond \vec{b}''_\beta]$$

where $\vec{a}'_\beta, \vec{a}''_\beta \subseteq \vec{a} \backslash \vec{b}_0$ and $\vec{b}'_\beta, \vec{b}''_\beta \subseteq \vec{b}$.

(ii) The following couples of sets are well-nested:

$$\left(\left\{C'_\beta[\vec{a}'_\beta \diamond \vec{b}'_\beta] \mid C[\vec{a} \diamond \vec{b}] \in \mathcal{C}\right\}, \left\{C'_\beta[\vec{a}'_\beta \diamond \vec{b}'_\beta] \mid C[\vec{a} \diamond \vec{b}] \in \mathcal{D}\right\}\right)$$

$$\left(\left\{C''_\beta[\vec{a}''_\beta \diamond \vec{b}''_\beta] \mid C[\vec{a} \diamond \vec{b}] \in \mathcal{C}\right\}, \left\{C''_\beta[\vec{a}''_\beta \diamond \vec{b}''_\beta] \mid C[\vec{a} \diamond \vec{b}] \in \mathcal{D}\right\}\right)$$

Proposition 18. *If $(\mathcal{C}, \mathcal{D})$ verifies the property (a) above, then it satisfies properties (i) and (ii).*

Proof. Trivial by taking $C'_\beta \equiv C''_\beta \equiv C$. □

Definition 57. We let head be the partial function defined on terms such that for all $f \in \mathcal{F}$, for all terms $\vec{t}$, $\mathsf{head}(f(\vec{t})) \equiv f$.

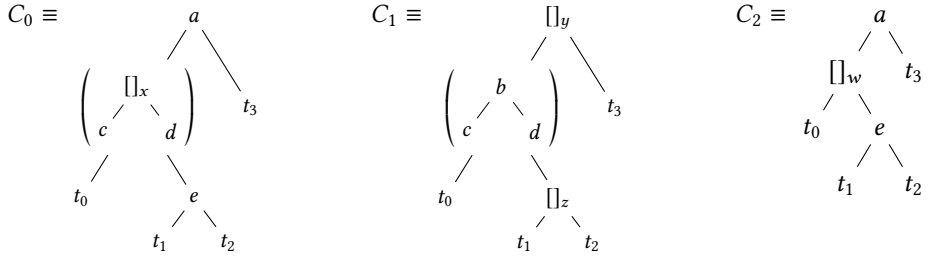
Definition 58. For all condition contexts C_0, C_1, we let $C_0 \sqcup_c C_1$ be the condition context, if it exists, defined as follows: $\mathsf{pos}(C_1 \sqcup_c C_2) = \mathsf{pos}(C_0) \cap \mathsf{pos}(C_1)$ and for all position p in $\mathsf{pos}(C_0 \sqcup_c C_1)$:

$$(C_0 \sqcup_c C_1)_{|p} \equiv \begin{cases} a & \text{if } \mathsf{head}((C_0)_{|p}) \equiv \mathsf{head}((C_1)_{|p}) \equiv a \qquad (a \in \mathcal{F} \cup \mathcal{N}) \\ []_x & \text{if } (C_0)_{|p} \equiv []_x \wedge \big(\mathsf{head}((C_1)_{|p}) \equiv []_x \vee \mathsf{head}((C_1)_{|p}) \equiv a\big) \qquad (a \in \mathcal{F} \cup \mathcal{N}) \\ []_x & \text{if } (C_1)_{|p} \equiv []_x \wedge \big(\mathsf{head}((C_0)_{|p}) \equiv []_x \vee \mathsf{head}((C_0)_{|p}) \equiv a\big) \qquad (a \in \mathcal{F} \cup \mathcal{N}) \end{cases}$$

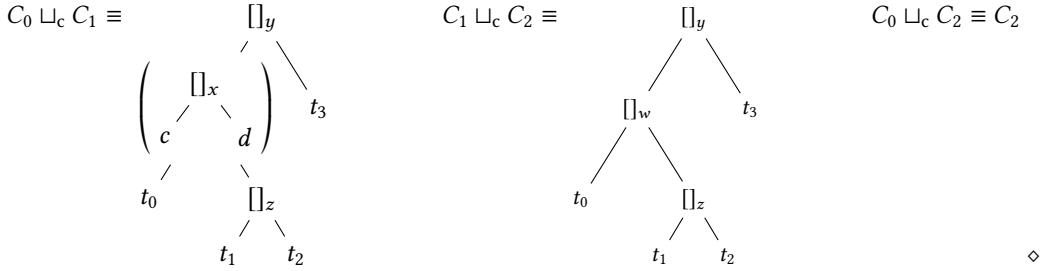
If such a condition context does not exist, then $C_0 \sqcup_c C_1$ is the special element undefined. We also let:

$$\mathsf{undefined} \sqcup_c C_0 \equiv C_0 \sqcup_c \mathsf{undefined} \equiv \mathsf{undefined}$$

Example 20. For all conditions a, b, c, d, e, f and terms $t_0, \ldots, t_3$, if we let:



Then we have:



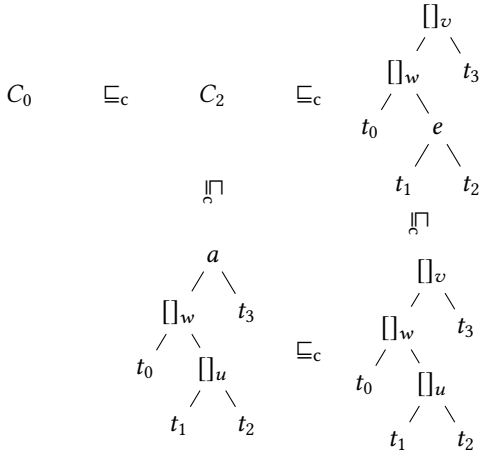
◇

Definition 59. We let $\sqsubseteq_c$ be the relation on condition contexts defined as follows: for all condition contexts C_0, C_1, we let $C_0 \sqsubseteq_c C_1$ hold if $\mathsf{pos}(C_1) \subseteq \mathsf{pos}(C_0)$ and for all position p in $\mathsf{pos}(C_1)$:

$$\text{if } \mathsf{head}((C_1)_{|p}) \equiv \begin{cases} a & \text{then } \mathsf{head}((C_0)_{|p}) \equiv a \qquad \text{where } (a \in \mathcal{F} \cup \mathcal{N}) \\ []_x & \text{then } \mathsf{head}((C_0)_{|p}) \in \mathcal{F} \cup \mathcal{N} \cup \{[]_x\} \end{cases}$$

Moreover we let $C_0 \sqsubseteq_c \mathsf{undefined}$ for all condition context C_0 (and $\mathsf{undefined} \sqsubseteq_c \mathsf{undefined}$).

Example 21. Using the condition contexts defined in Example 20, we have, for example, the following relations:



◇

Proposition 19. *Let $\mathcal{S}_{cc}$ be the set of condition contexts extended with undefined. Then $(\mathcal{S}_{cc}, \sqcup_c, \sqsubseteq_c)$ is a semi-lattice. That is, we have the following properties:*

(i) $\sqcup_c$ is associative, commutative, idempotent.
(ii) $\sqsubseteq_c$ is an pre-order (i.e. reflexive and transitive).
(iii) For all $C_0, C_1 \in \mathcal{S}_{cc}$, we have $C_0 \sqsubseteq_c (C_0 \sqcup_c C_1)$ and $C_1 \sqsubseteq_c (C_0 \sqcup_c C_1)$. Moreover $(C_0 \sqcup_c C_1)$ 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 $(C_0 \sqcup_c C_1)$ is the least upper-bound of C_0 and C_1. Assume that there is C such that:

$$C_0 \sqsubseteq_c C \sqsubseteq_c C_0 \sqcup_c C_1 \qquad\qquad C_1 \sqsubseteq_c C \sqsubseteq_c C_0 \sqcup_c C_1$$

If $C_0 \sqcup_c C_1 \equiv$ undefined then one can check that $C \equiv$ undefined. Otherwise we know that $\mathsf{pos}(C_0 \sqcup_c C_1) = \mathsf{pos}(C_0) \cap \mathsf{pos}(C_1)$, and that:

$$\mathsf{pos}(C_0) \supseteq \mathsf{pos}(C) \supseteq \mathsf{pos}(C_0 \sqcup_c C_1) \qquad\qquad \mathsf{pos}(C_1) \supseteq \mathsf{pos}(C) \supseteq \mathsf{pos}(C_0 \sqcup_c C_1)$$

Hence $\mathsf{pos}(C) = \mathsf{pos}(C_0 \sqcup_c C_1)$. Using the fact that $C \sqsubseteq_c C_0 \sqcup_c C_1$ we know that for all position $p \in \mathsf{pos}(C)$, if $\mathsf{head}((C_0 \sqcup_c C_1)_{|p}) = a$ (with $a \in \mathcal{F} \cup \mathcal{N}$) then $\mathsf{head}(C_{|p}) = a$. If $\mathsf{head}((C_0 \sqcup_c C_1)_{|p}) = []_x$ then either $\mathsf{head}(C_{|p}) = []_x$ or $\mathsf{head}(C_{|p}) = a$ (with $a \in \mathcal{F} \cup \mathcal{N}$). In the former case there is nothing to show, in the the latter case observe that $\mathsf{head}((C_0 \sqcup_c C_1)_{|p}) = []_x$ implies that either $\mathsf{head}((C_0)_{|p}) = []_x$ or $\mathsf{head}((C_1)_{|p}) = []_x$. W.l.o.g. assume $\mathsf{head}((C_0)_{|p}) = []_x$. Then using the fact that $C_0 \sqsubseteq_c C$, we know that $\mathsf{head}((C_0)_{|p}) = []_x$ implies that $\mathsf{head}((C_0)_{|p}) = []_x$. Absurd.

Therefore $\forall p \in \mathsf{pos}(C), \mathsf{head}(C_{|p}) = \mathsf{head}((C_0 \sqcup_c C_1)_{|p})$. Moreover $\mathsf{pos}(C) = \mathsf{pos}(C_0 \sqcup_c C_1)$. Hence $C \equiv C_0 \sqcup_c C_1$. □

Let $C_0, C_1 \in \mathcal{S}_{cc}$ such that $C_0 \sqsubseteq_c C_1$. Moreover, assume that:

$$\forall p, p' \in \mathsf{pos}(C_1), (C_1)_{|p} \equiv (C_1)_{|p'} \equiv []_x \;\Rightarrow\; (C_0)_{|p} \equiv (C_0)_{|p'}$$

Then, we know that C_0 and C_1 coincides on $\mathsf{pos}(C_1)$. Therefore, any $\to_R$ reduction done on C_1 can be mimicked on C_0. We simultaneously reduce C_1 and C_0, which yields the terms C_1' and C_0', where C_1' is in R-normal form. Then the conditions of C_1' which do not have hole variables (i.e. $\mathsf{cond\text{-}st}(C_1' \downarrow_R) \cap \mathcal{T}(\mathcal{F}_{\setminus \mathsf{if}}, \mathcal{N})$) all appear directly as subterm of C_0', hence are in $\overline{\mathsf{cond\text{-}st}}(C_0')$.

Proposition 20. *For every* $C_0, C_1 \in \mathcal{S}_{cc}$, *if* $C_0 \sqsubseteq_c C_1$ *and if:*

$$\forall p, p' \in pos(C_1), (C_1)_{|p} \equiv (C_1)_{|p'} \equiv []_x \implies (C_0)_{|p} \equiv (C_0)_{|p'}$$

then cond-st$(C_1 \downarrow_R) \cap \mathcal{T}(\mathcal{F}_{\setminus if}, \mathcal{N}) \subseteq \overline{\text{cond-st}}(C_0)$.

Proof. Assume that $C_0 \sqsubseteq_c C_1$, with $C_0, C_1 \neq$ undefined (the cases $C_0 =$ undefined and $C_1 =$ undefined are easy to handle, with the convention that cond-st(undefined) $= \emptyset$), and that:

$$\forall p, p' \in \text{pos}(C_1), (C_1)_{|p} \equiv (C_1)_{|p'} \equiv []_x \implies (C_0)_{|p} \equiv (C_0)_{|p'} \tag{9}$$

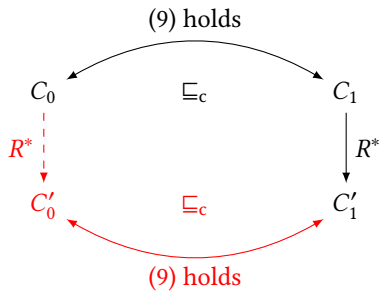
First we show that we can extend this property as follows:

$$\forall p, p' \in \text{pos}(C_1), (C_1)_{|p} \equiv (C_1)_{|p'} \implies (C_0)_{|p} \equiv (C_0)_{|p'} \tag{10}$$

Let $q = p \cdot q_0$ and $q = p' \cdot q_0$ be positions in pos(C_1). Since $(C_1)_{|p} \equiv (C_1)_{|p'}$, we know that head$((C_1)_{|q}) \equiv$ head$((C_1)_{|q'})$.

- If head$((C_1)_{|q}) \equiv a$ (with $a \in \mathcal{F} \cup \mathcal{N}$) then, from the fact that $C_0 \sqsubseteq_c C_1$ we get that head$((C_0)_{|q}) \equiv a$, and that head$((C_0)_{|q'}) \equiv a$.
- If head$((C_1)_{|q}) \equiv []_x$ then using (9) we get that $(C_0)_{|p} \equiv (C_0)_{|p'}$.

. Then, 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 $C_0' \sqsubseteq_c C_1'$, (9) holds for C_0', C_1' and $C_0 \to_R^* C_0'$. Graphically (hypothesis are in black, goals are in red):



Let $\to_{R'}$ be $\to_R$ without the non left-linear rules, which are:

$$\text{if } x \text{ then } y \text{ else } y \to y \qquad\qquad \text{dec}(\{x\}^r_{\text{pk}(y)}, \text{sk}(y)) \to x$$

$$\text{if } w \text{ then (if } w \text{ then } x \text{ else } y) \text{ else } z \to \text{if } w \text{ then } x \text{ else } z$$

$$\text{if } w \text{ then } x \text{ else (if } w \text{ then } y \text{ else } z) \to \text{if } w \text{ then } x \text{ else } z$$

We mimic all reduction $\to_R$ on C_1 by a reduction on C_0, while maintaining $\sqsubseteq_c$ and the invariant of (9). Mimicking rules in $\to_R$ is easy as they are left-linear. To mimic rules in $(\to_R \setminus \to_{R'})$, we use (10).

Therefore let C_1' be in R-normal form such that $C_1 \to_R^* C_1'$. Let C_0' be such that $C_0' \sqsubseteq_c C_1'$,(9) 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:

$$\text{cond-st}(C_1 \downarrow_R) \cap \mathcal{T}(\mathcal{F}_{\setminus \text{if}}, \mathcal{N}) = \text{cond-st}(C_1') \cap \mathcal{T}(\mathcal{F}_{\setminus \text{if}}, \mathcal{N}) = \vec{b}$$

Finally, we observe that $\vec{b} \subseteq \overline{\text{cond-st}}(C_0')$, and that $\overline{\text{cond-st}}(C_0') \subseteq \overline{\text{cond-st}}(C_0)$ by Proposition 15. □

Lemma 19. *For all $P \vdash^{npf} t \sim t'$, for all h, l, the following couple of sets is well-nested:*

$$\Big(\{\beta \downarrow_R |\ \beta \leq_c^{h,l} (t,P)\}, \{\gamma \downarrow_R |\ \gamma \leq_l^{h,l} (t,P)\}\Big)$$

Proof. We do this proof in the case $\mathsf{h} = \epsilon$. The other cases are identical.

Part 1. We consider an arbitrary ordering $(\beta_i)_{1 \leq i \leq i_{\max}}$ of:

$$\{\beta \mid \beta \leq_{\mathsf{c}}^{\mathsf{h},l} (t,P)\}$$

Using Lemma 18, we know that all $i \neq i_0$, there exists a condition context $\tilde{\beta}_i$ such that:

$$\beta_i \equiv \tilde{\beta}_i\left[\beta_{i_0}\right] \quad \text{and} \quad \mathsf{leave\text{-}st}(\beta_{i_0} \downarrow_R) \cap \overline{\mathsf{cond\text{-}st}}(\tilde{\beta}_i) = \emptyset$$

From now on we use $\beta_i^{(i_0)}$ to denote this condition context, and $[]_{i_0}$ the hole variable used in the condition contexts $\{\beta_i^{(i_0)} \mid 1 \leq i \leq i_{\max}\}$. We extend this notation to $i_0 = 0$ by letting $\beta_i^{(0)} \equiv \beta_i$.

Let $1 \leq i \leq i_{\max}$, and let $l_0, \ldots, l_n$ be a sequence of distinct indices in $\{0, \ldots, i_{\max}\}$ such that $l_0 = 0$. Using Proposition 19.((iii)) we know that for every $0 \leq j_0 \leq n$, if $i \neq l_{j_0}$ then:

$$\beta_i^{(l_{j_0})} \sqsubseteq_{\mathsf{c}} \sqcup_{\mathsf{c}\, j \leq n} \beta_i^{(l_j)}$$

Using Proposition 20, we deduce that:

$$\overline{\mathsf{cond\text{-}st}}\big(\beta_i^{(l_{j_0})}\big) \supseteq \mathsf{cond\text{-}st}\big(\sqcup_{\mathsf{c}\, j \leq n} \beta_i^{(l_j)} \downarrow_R\big) \cap \mathcal{T}(\mathcal{F}_{\setminus \mathsf{if}}, \mathcal{N})$$

Which implies that:

$$\mathsf{leave\text{-}st}(\beta_{l_{j_0}} \downarrow_R) \cap \mathsf{cond\text{-}st}\big(\sqcup_{\mathsf{c}\, j \leq n} \beta_i^{(l_j)} \downarrow_R\big) = \emptyset \tag{11}$$

Moreover, if $n = n_0 + 1$ and $i \neq l_{n+1}$, we can check that:

$$\begin{aligned} \sqcup_{\mathsf{c}\, j \leq n_0} \beta_i^{(l_j)} \ &\equiv\ \big(\sqcup_{\mathsf{c}\, j \leq n_0+1} \beta_i^{(l_j)}\big)\{\sqcup_{\mathsf{c}\, j \leq n_0} \beta_{l_{n_0+1}}^{(l_j)} / []_{l_{n_0+1}}\} \\ &=_R \mathsf{if}\ \big(\sqcup_{\mathsf{c}\, j \leq n_0} \beta_{l_{n_0+1}}^{(l_j)}\big)\ \mathsf{then}\ \big(\sqcup_{\mathsf{c}\, j \leq n_0+1} \beta_i^{(l_j)}\big)\{\mathsf{true}/[]_{l_{n_0+1}}\} \\ &\qquad\qquad \mathsf{else}\ \big(\sqcup_{\mathsf{c}\, j \leq n_0+1} \beta_i^{(l_j)}\big)\{\mathsf{false}/[]_{l_{n_0+1}}\} \end{aligned} \tag{12}$$

Part 2. Similarly, let $(\gamma_m)_{1 \leq m \leq m_{\max}}$ be an arbitrary ordering of:

$$\{\gamma \mid \gamma \leq_{\mathsf{l}}^{\mathsf{h},l} (t,P)\}$$

Let $1 \leq i_0 \leq i_{\max}$. For every m, we have $\gamma_m^{(i_0)}$ such that:

$$\gamma_m \equiv \gamma_m^{(i_0)}\left[\beta_{i_0}\right] \quad \text{and} \quad \mathsf{leave\text{-}st}(\beta_{i_0} \downarrow_R) \cap \overline{\mathsf{cond\text{-}st}}\big(\gamma_m^{(i_0)}\big) = \emptyset$$

Moreover, we let $\gamma_m^{(0)} \equiv \gamma_m$. Let $1 \leq m \leq m_{\max}$, and let $l_0, \ldots, l_n$ be a sequence of distinct indices in $\{0, \ldots, i_{\max}\}$ such that $l_0 = 0$. We have equations corresponding to (11) and (12), with $\sqcup_{\mathsf{c}\, j \leq n} \gamma_m^{(l_j)}$ instead of $\sqcup_{\mathsf{c}\, j \leq n} \beta_i^{(l_j)}$.

Part 3. Consider the following family of couples of sets:

$$\begin{aligned} \Big\{\Big(\big(\sqcup_{\mathsf{c}\, j \leq n} \beta_i^{(l_j)}\{e_j/[]_{l_j} \mid j \leq n\} \downarrow_R\big)_{1 \leq i \leq i_{\max}}, \big(\sqcup_{\mathsf{c}\, j \leq n} \gamma_m^{(l_j)}\{e_j/[]_{l_j} \mid j \leq n\} \downarrow_R\big)_{1 \leq m \leq m_{\max}}\Big) \ \Big| \\ l_0, \ldots, l_n \text{ distinct indices in } \{0, \ldots, i_{\max}\} \text{ s.t. } l_0 = 0 \text{ and } (e_j)_{1 \leq j \leq n} \in \{\mathsf{true}, \mathsf{false}\}^n\Big\} \end{aligned} \tag{13}$$

We show by decreasing induction on n, starting from $n = i_{\max} + 1$ down to $n = 0$, that all the elements above are well-nested.

Let $l_0, \ldots, l_n$ be distinct indices in $\{0, \ldots, i_{\max}\}$ such that $l_0 = 0$, and let $(e_j)_{1 \leq j \leq n} \in \{\mathsf{true}, \mathsf{false}\}^n$.

Base case. If $n = n_{\max} + 1$ then from (11) we get that for every $l \neq i$ in $\{1, \ldots, i_{\max}\}$:

$$\mathsf{leave\text{-}st}(\beta_l \downarrow_R) \cap \mathsf{cond\text{-}st}\big(\big(\sqcup_{\mathrm{c}\, j \leq n}\, \beta_i^{(j)}\big)\{e_j/[]_j \mid j \leq n\} \downarrow_R\big) = \emptyset$$

Moreover:

$$\mathsf{leave\text{-}st}\big(\big(\sqcup_{\mathrm{c}\, j \leq n}\, \beta_l^{(j)}\big)\{e_j/[]_j \mid j \leq n\} \downarrow_R\big) \subseteq \mathsf{leave\text{-}st}(\beta_l \downarrow_R)$$

Hence:

$$\mathsf{leave\text{-}st}\big(\big(\sqcup_{\mathrm{c}\, j \leq n}\, \beta_l^{(j)}\big)\{e_j/[]_j \mid j \leq n\} \downarrow_R\big) \cap \mathsf{cond\text{-}st}\big(\big(\sqcup_{\mathrm{c}\, j \leq n}\, \beta_i^{(j)}\big)\{e_j/[]_j \mid j \leq n\} \downarrow_R\big) = \emptyset \quad (14)$$

Similarly, for every $1 \leq m \leq m_{\max}$:

$$\mathsf{leave\text{-}st}\big(\big(\sqcup_{\mathrm{c}\, j \leq n}\, \beta_l^{(j)}\big)\{e_j/[]_j \mid j \leq n\} \downarrow_R\big) \cap \mathsf{cond\text{-}st}\big(\big(\sqcup_{\mathrm{c}\, j \leq n}\, \gamma_m^{(j)}\big)\{e_j/[]_j \mid j \leq n\} \downarrow_R\big) = \emptyset$$

By consequence, the following set is well-nested:

$$\big(\big(\sqcup_{\mathrm{c}\, j \leq n}\, \beta_i^{(j)}\{e_j/[]_j\} \downarrow_R\big)_{1 \leq i \leq i_{\max}}, \big(\sqcup_{\mathrm{c}\, j \leq n}\, \gamma_m^{(j)}\{e_j/[]_j\} \downarrow_R\big)_{1 \leq m \leq m_{\max}}\big)$$

Inductive Case. If $n \leq n_{\max} \neq 0$, then from (12) we get that for every $l \neq i$ in $\{1, \ldots, i_{\max}\}$:

$$\begin{aligned}\big(\sqcup_{\mathrm{c}\, j \leq n}\, \beta_i^{(l_j)}\big)\{e_{l_j}/[]_{l_j} \mid j \leq n\} \;=_R\\ \text{if } \big(\big(\sqcup_{\mathrm{c}\, j \leq n}\, \beta_{l_{n+1}}^{(l_j)}\big)\{e_{l_j}/[]_{l_j} \mid j \leq n\}\big) \text{ then } \big(\sqcup_{\mathrm{c}\, j \leq n+1}\, \beta_i^{(l_j)}\big)\{e_{l_j}/[]_{l_j} \mid j \leq n\}\{\mathsf{true}/[]_{l_{n+1}}\}\\ \text{else } \big(\sqcup_{\mathrm{c}\, j \leq n+1}\, \beta_i^{(l_j)}\big)\{e_{l_j}/[]_{l_j} \mid j \leq n\}\{\mathsf{false}/[]_{l_{n+1}}\}\end{aligned}$$

As we did for (14), we can show that for every $i \neq l_{n+1}$:

$$\mathsf{leave\text{-}st}(\sqcup_{\mathrm{c}\, j \leq n} \beta_{l_{n+1}}^{(l_j)} \downarrow_R) \cap \mathsf{cond\text{-}st}\big(\big(\sqcup_{\mathrm{c}\, j \leq n+1}\, \beta_i^{(l_j)}\big)\{e_{l_j}/[]_{l_j} \mid j \leq n+1\} \downarrow_R\big) = \emptyset$$

Where $e_{l_{n+1}}$ is either true or false. Similarly, for every m:

$$\mathsf{leave\text{-}st}(\sqcup_{\mathrm{c}\, j \leq n} \beta_{l_{n+1}}^{(l_j)} \downarrow_R) \cap \mathsf{cond\text{-}st}\big(\big(\sqcup_{\mathrm{c}\, j \leq n+1}\, \gamma_m^{(l_j)}\big)\{e_{l_j}/[]_{l_j} \mid j \leq n+1\} \downarrow_R\big) = \emptyset$$

Moreover, by induction hypothesis, we know that:

$$\big(\big(\sqcup_{\mathrm{c}\, j \leq n+1}\, \beta_i^{(l_j)}\big)\{e_{l_j}/[]_{l_j} \mid j \leq n+1\} \downarrow_R\big)_i, \big(\big(\sqcup_{\mathrm{c}\, j \leq n+1}\, \gamma_i^{(l_j)}\big)\{e_{l_j}/[]_{l_j} \mid j \leq n+1\} \downarrow_R\big)_i\big)$$

is well-nested for $e_{l_{n+1}} \equiv \mathsf{true}$ and for $e_{l_{n+1}} \equiv \mathsf{false}$. We deduce that the following set is well nested:

$$\big(\big(\sqcup_{\mathrm{c}\, j \leq n}\, \beta_i^{(l_j)}\big)\{e_{l_j}/[]_{l_j} \mid j \leq n\} \downarrow_R\big)_i, \big(\big(\sqcup_{\mathrm{c}\, j \leq n}\, \gamma_i^{(l_j)}\big)\{e_{l_j}/[]_{l_j} \mid j \leq n\} \downarrow_R\big)_i\big)$$

Conclusion. Recall that $\beta_i^{(l_0)} \equiv \beta_i^{(0)} \equiv \beta_i$. Hence:

$$\Big(\big\{\beta \downarrow_R \mid \beta \leq_{\mathrm{c}}^{\epsilon,l} (t, P)\big\}, \big\{\gamma \downarrow_R \mid \gamma \leq_{\mathrm{I}}^{\epsilon,l} (t, P)\big\}\Big)$$

is the couple of sets:

$$\big(\big(\sqcup_{\mathrm{c}\, j \leq 0}\, \beta_i^{(l_j)}\big) \downarrow_R\big)_{1 \leq i \leq i_{\max}}, \big(\big(\sqcup_{\mathrm{c}\, j \leq 0}\, \gamma_m^{(l_j)}\big)\} \downarrow_R\big)_{1 \leq m \leq m_{\max}}\big)$$

which is the family of well-nested sets in (13), and is therefore well-nested. □

F SPURIOUS CONDITIONS AND PERSISTENT LEAVES

An if-free conditions b is *spurious* in a term t if, when we normalize t, the condition b disappears. For example, the condition b is spurious in if b then 0 else 0.

Definition 60. An if-free condition b is said to be *spurious* in a term t if $b \downarrow_R \notin \text{cond-st}(t \downarrow_R)$.

An if-free term u is *persistent* in a term t if, when we normalize t, the term u *does not* disappear. For example, n_0 is persistent in if b then n_0 else if b then n_1 else n_2, but n_2 is not.

Definition 61. An if-free terms u is said to be *persistent* in a term t if $u \downarrow_R \in \text{cond-st}(t \downarrow_R)$.

The notion of *spurious set* is related to the notion of *spurious condition*. A set of position S in a term is a spurious set if we can safely replace in t the terms at positions S by true.

Definition 62. A set of positions S is spurious in a term t if it is non-empty and $t[\mathsf{true}/x \mid x \in S] =_R t[\mathsf{false}/x \mid x \in S] =_R t$. 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 22. Let $a \equiv \text{eq}(A, 0)$ and $b \equiv \text{eq}(B, 0)$ be two conditions. Consider the term t:

$$\begin{array}{l}\text{if } b \text{ then if } a \text{ then if } b \text{ then } T \text{ else } U\\ \qquad\qquad\qquad\qquad\text{else } V\\ \qquad\qquad\text{else if } a \text{ then } T\\ \qquad\qquad\qquad\qquad\text{else if } a \text{ then } V \text{ else } V\end{array}$$

Then the condition b is spurious in t, since b is not a subterm of $t \downarrow_R \equiv \text{if } a \text{ then } T \text{ else } V$. Moreover the condition a is a subterm of $t \downarrow_R$, hence is not spurious. Nonetheless, the set of position $S = \{220\}$ is spurious. Indeed we have:

$$\begin{array}{l}\text{if } b \text{ then if } a \text{ then if } b \text{ then } T \text{ else } U\\ \qquad\qquad\qquad\text{else } V\\ \qquad\qquad\text{else if } a \text{ then } T\\ \qquad\qquad\qquad\text{else if } \boxed{a}_{220} \text{ then } V \text{ else } V\end{array} \quad =_R \quad \begin{array}{l}\text{if } b \text{ then if } a \text{ then if } b \text{ then } T \text{ else } U\\ \qquad\qquad\qquad\qquad\text{else } V\\ \qquad\qquad\qquad\text{else if } a \text{ then } T\\ \qquad\qquad\qquad\qquad\text{else if } \boxed{\mathsf{true}}_{220} \text{ then } V \text{ else } V\end{array}$$

$$=_R \quad \begin{array}{l}\text{if } b \text{ then if } a \text{ then if } b \text{ then } T \text{ else } U\\ \qquad\qquad\qquad\qquad\text{else } V\\ \qquad\qquad\qquad\text{else if } a \text{ then } T\\ \qquad\qquad\qquad\qquad\text{else if } \boxed{\mathsf{false}}_{220} \text{ then } V \text{ else } V\end{array} \quad \diamond$$

First Objective. Let t be a term, and a be a spurious condition in t such that a is a sub-term of t. If this happens in a proof $P \vdash^{\text{npf}} t \sim t'$, we would like to find a proof-cut elimination getting rid of a. A way of building such a cut elimination is to find a set of positions S which is spurious in both t and t', and such that for every $p \in S$ and $t_{|p} \equiv a$. Then, under some conditions on S, we may be able to obtain a proof $P' \vdash^{\text{npf}} t\{\mathsf{true}/S\} \sim t'\{\mathsf{true}/S\}$. If we can repeat this proof cut sufficiently many times, we may eventually remove all occurrences of a in t.

Our first goal is the following: given a term t and a spurious condition a in t, and given a set of positions S such that for every $p \in S$ and $t_{|p} \equiv a$, we give sufficient conditions under which S is a spurious set in t. This is done in Section F.1.

Second Objective. Consider a proof $P \vdash^{\mathsf{npf}} t \sim t'$, where t is of the form:

$$t \equiv C[(\beta_i)_{i\in I} \diamond (\gamma_j)_{j\in J}]$$

where $(\beta_i)_{i\in I}$ and $(\gamma_j)_{j\in J}$ are $\mathcal{S}$-normalized basic terms and $\mathcal{S}$ is the left CCA_2 trace of P. Remember that we showed in Corollary 2.(ii) that for every $j, j' \in J$:

$$\mathsf{leave\text{-}st}(\gamma_j \downarrow_R) \cap \mathsf{leave\text{-}st}(\gamma_{j'} \downarrow_R) \neq \emptyset \quad \text{implies that} \quad \gamma_j \equiv \gamma_{j'}$$

This followed from the fact that given a leaf $u \in \mathsf{leave\text{-}st}(\gamma_j \downarrow_R)$, there exists a unique way of completing u into a $\mathcal{S}$-normalized basic term. Moreover, we will see later that $|\gamma_j|$ is bounded by $|u|$. Assume that we can show that, for every j, $\mathsf{leave\text{-}st}(\gamma_j \downarrow_R)$ contains a persistent term in t, i.e. that $\mathsf{leave\text{-}st}(\gamma_j \downarrow_R) \cap \mathsf{leave\text{-}st}(t \downarrow_R)$ is non-empty. Since $\mathsf{leave\text{-}st}(t \downarrow_R)$ is bounded by the size of the normal form of t, we just bounded the size of the set $\{\gamma_j \mid j \in J\}$.

Therefore, a way of bounding the size of the $\mathcal{S}$-normalized basic terms $(\gamma_j)_{j\in J}$ is to show that they all have a persistent leaf. In other word, we want to prove that we can assume, w.l.o.g., that for every j:

$$\mathsf{leave\text{-}st}(\gamma_j \downarrow_R) \cap \mathsf{leave\text{-}st}(t \downarrow_R) \neq \emptyset$$

This is a key lemma to show decidability. In Section F.2, we give sufficient conditions for this to hold.

F.1 Spurious Conditions to Spurious Sets

We give sufficient conditions under which a set of positions S is spurious in a term t.

Lemma 20. *Let $a, \vec{a}, \vec{b}, \vec{c}$ be if-free conditions in R-normal form. Let s be the context:*

$$\tau[] \equiv B\left[\vec{c} \diamond \left(\vec{w}, \mathit{if}\ C[\vec{b} \diamond \vec{a}, []]\ \mathit{then}\ u\ \mathit{else}\ v\right)\right]$$

Let t be the term $\tau[a]$, and assume that a is spurious in t, and that:

- $a \notin \vec{a} \cup \vec{b} \cup \{\mathit{true}, \mathit{false}\} \cup \mathsf{cond\text{-}st}(u \downarrow_R) \cup \mathsf{cond\text{-}st}(v \downarrow_R)$.
- $a \notin \rho$ *where ρ is the set of conditions appearing on the path from the root to if $C[\vec{b} \diamond \vec{a}, a]$ then u else v.*

Then $t \equiv \tau[a] =_R \tau[\mathit{true}]$.

Proof. We start by observing that:

$$\begin{aligned}\mathsf{if}\ C[\vec{b} \diamond \vec{a}, a]\ \mathsf{then}\ u\ \mathsf{else}\ v\ =_R\ &\mathsf{if}\ a\ \mathsf{then}\ \mathsf{if}\ C[\vec{b} \diamond \vec{a}, \mathsf{true}]\ \mathsf{then}\ u\ \mathsf{else}\ v\\ &\mathsf{else}\ \mathsf{if}\ C[\vec{b} \diamond \vec{a}, \mathsf{false}]\ \mathsf{then}\ u\ \mathsf{else}\ v\end{aligned}$$

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 :

$$\begin{aligned}\mathsf{if}\ C[\vec{b} \diamond \vec{a}, \mathsf{true}]\ \mathsf{then}\ u\ \mathsf{else}\ v =_R C_l[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v]\\ \mathsf{if}\ C[\vec{b} \diamond \vec{a}, \mathsf{false}]\ \mathsf{then}\ u\ \mathsf{else}\ v =_R C_r[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v]\end{aligned}$$

Since $a \notin \mathsf{cond\text{-}st}(u \downarrow_R), \mathsf{cond\text{-}st}(v \downarrow_R)$ we know that $a \notin \vec{b}_u, \vec{b}_v$. Moreover since $a \notin \vec{a} \cup \vec{b}$ we know that $a \notin \vec{b}_u, \vec{b}_v, \vec{b}, \vec{a}$. Therefore:

$$a \notin \mathsf{cond\text{-}st}(C_l[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v]) \qquad a \notin \mathsf{cond\text{-}st}(C_r[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v])$$

We get rid in C_l and C_r of all the conditions appearing in ρ. We let $\vec{a}^{\mathsf{l}}$ and $\vec{a}^{\mathsf{r}}$ be such that:

$$\vec{a}^{\mathsf{l}} \subseteq \vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \backslash \rho \qquad \vec{a}^{\mathsf{r}} \subseteq \vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \backslash \rho$$

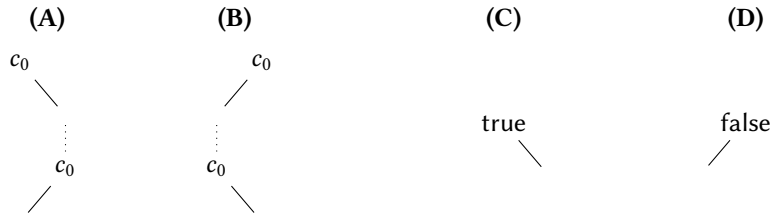
and C'_l, C'_r such that:

$$B\left[\vec{c} \diamond \left(\vec{w}, C_l[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v]\right)\right] =_R B\left[\vec{c} \diamond \left(\vec{w}, C'_l[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v]\right)\right] \tag{15}$$

$$B\left[\vec{c} \diamond \left(\vec{w}, C_r[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v]\right)\right] =_R B\left[\vec{c} \diamond (\vec{w}, C'_r[\vec{a}^{\mathsf{r}} \diamond \vec{t}_u, \vec{t}_v])\right] \tag{16}$$

We know that $a \notin \vec{a}^{\mathsf{l}}$ and $a \notin \vec{a}^{\mathsf{r}}$.

Case 1. If there exists $c_0 \in \vec{c}$ such that the path ρ from the root of t to if $C[\vec{b} \diamond \vec{a}]$ then u 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 we can do any rewriting we want. For example, in case **(A)**, we use the fact that:

$$\text{if } x \text{ then } y \text{ else } (\text{if } x \text{ then } \mathbf{v} \text{ else } z) \to_R^* \text{if } x \text{ then } y \text{ else } (\text{if } x \text{ then } \mathbf{v}' \text{ else } z) \quad (\text{ for all term } \mathbf{v}')$$

to rewrite (if $C[\vec{b} \diamond \vec{a}, a]$ then u else v) into the term (if a then if $C[\vec{b} \diamond \vec{a}, \text{true}]$ then u else v else).

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' and $\vec{\omega}$ such that $s[] =_R B'\left[\vec{c} \diamond (\vec{\omega}, [])\right]$ and the path ρ' from the root of s 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 conditions, 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 $\rho' = c_1, \ldots, c_n$, we take B' minimal, i.e. only a branch $c_1, \ldots, c_n$. We give an example of such an if-context in Figure 14.

Wet let $\vec{\omega} = w_1, \ldots, w_n$, and we have:

$$s =_R B'\left[c_1, \ldots, c_n \diamond w_1, \ldots, w_n, []\right]$$

We let $\succ_u$ be a total ordering on if-free condition in R-normal form such that the $n+1$ maximum elements are $c_1 \succ_u \cdots \succ c_n \succ_u a$. For every y, we let $W_i[\vec{d}_i \diamond \vec{e}_i]$ be the $R_{\succ_u}$-normal form of w_i. Then:

$$s =_R B'\left[c_1, \ldots, c_n \diamond \left(W_i[\vec{d}_i \diamond \vec{e}_i]\right)_{i \le n}, []\right]$$

We get rid of any occurrence of $c_1, \ldots, c_n$ in $(\vec{d}_i)_i$. For every i, we let $W'_i[\vec{d}'_i \diamond \vec{e}'_i]$ be terms in R-normal form such that $\vec{d}'_i \cap \{c_j \mid j \le i\} = \emptyset$ and:

$$s =_R B'\left[c_1, \ldots, c_n \diamond \left(W'_i[\vec{d}'_i \diamond \vec{e}'_i]\right)_{i \le n}, []\right]$$

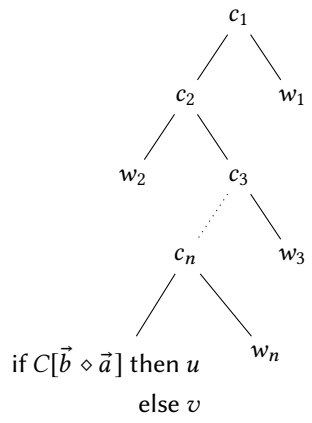


Fig. 14. Example of if-context B'

Using (15) and (16) we get:

$$t =_R B'\left[c_1, \ldots, c_n \diamond \left(W_i'[\vec{d}_i' \diamond \vec{e}_i']\right)_{i \leq n}, \begin{array}{r}\text{if } a \text{ then } C_l'[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v] \\ \text{else } C_r'[\vec{a}^{\mathsf{r}} \diamond \vec{t}_u, \vec{t}_v]\end{array}\right]$$

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:

$$\begin{aligned} t \downarrow_{R_{\succ u}} &\equiv \left(B'\left[c_1, \ldots, c_n \diamond \left(W_i'[\vec{d}_i' \diamond \vec{e}_i']\right)_{i \leq n}, \begin{array}{r}\text{if } a \text{ then } C_l'[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v] \\ \text{else } C_r'[\vec{a}^{\mathsf{r}} \diamond \vec{t}_u, \vec{t}_v]\end{array}\right]\right) \downarrow_{R_{\succ u}} \\ &\equiv B''\left[c_{i_1}, \ldots, c_{i_k} \diamond \left(W_{ij}'[\vec{d}_{i_j}' \diamond \vec{e}_{i_j}']\right)_{j \leq k}, \left(\begin{array}{r}\text{if } a \text{ then } C_l'[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v] \\ \text{else } C_r'[\vec{a}^{\mathsf{r}} \diamond \vec{t}_u, \vec{t}_v]\end{array}\right) \downarrow_{R_{\succ u}}\right] \end{aligned}$$

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 condition in the R-normal form of this term (because of the order $\succ_u$ we chose), and since $a \neq \text{true}$ $a \neq \text{false}$ 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 $x \longrightarrow x$. 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:

$$\begin{aligned} t &=_R B'\left[c_1, \ldots, c_n \diamond \left(W_i'[\vec{d}_i' \diamond \vec{e}_i']\right)_{i \leq n}, C_l'[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v]\right] \\ &=_R s\left[C_l'[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v]\right] \\ &=_R B\left[\vec{c} \diamond \left(\vec{w}, C_l'[\vec{a}^{\mathsf{l}} \diamond \vec{t}_u, \vec{t}_v]\right)\right] \end{aligned}$$

Hence using (15) we get:

$$t =_R B\left[\vec{c} \diamond \left(\vec{w}, C_l[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v]\right)\right] =_R B\left[\vec{c} \diamond \left(\vec{w}, \text{if } C[\vec{b} \diamond \vec{a}, \text{true}] \text{ then } u \text{ else } v\right)\right] \qquad \square$$

F.2 Persistent Terms

Let a be a condition and $s[]$ be a context. The following proposition give sufficient conditions under which the persistent terms of $s[a]$ are exactly the persistent terms of $s[\mathsf{true}]$ and $s[\mathsf{false}]$.

Proposition 21. *Let $a, (\vec{a}_i, \vec{b}_i)_i, (\vec{c}_j, \vec{t}_j)_j$ be if-free terms in R-normal form such that for every i, $a \notin \vec{a}_i \cup \vec{b}_i \cup \vec{c}_j$, and let $s[]$ be a context such that:*

$$s[] \equiv B\left[\left(C_i[\vec{a}_i, [] \diamond \vec{b}_i, []]\right)_i \diamond \left(D_j[\vec{c}_j, [] \diamond \vec{t}_j]\right)_j\right]$$

Then $\mathsf{leave\text{-}st}(s[a]\downarrow_R) = \mathsf{leave\text{-}st}(s[\mathit{true}]\downarrow_R) \cup \mathsf{leave\text{-}st}(s[\mathit{false}]\downarrow_R)$.

Proof. We know that $s[a] =_R$ if a then $s[\mathsf{true}]$ else $s[\mathsf{false}]$. Let $\succ_u$ be a total order on if-free conditions in R-normal form such that a is minimal. It is straightforward to check that:

$$\begin{aligned} s[a]\downarrow_{R^{\succ u}} &\equiv (\text{if } a \text{ then } s[\mathsf{true}] \text{ else } s[\mathsf{false}])\downarrow_{R^{\succ u}} \\ &\equiv \begin{cases} (s[\mathsf{true}])\downarrow_{R^{\succ u}} & \text{if } s[\mathsf{true}] =_{R^{\succ u}} s[\mathsf{false}] \\ \text{if } a \text{ then } (s[\mathsf{true}])\downarrow_{R^{\succ u}} \text{ else } (s[\mathsf{false}])\downarrow_{R^{\succ u}} & \text{otherwise} \end{cases} \end{aligned}$$

Therefore:

$$\mathsf{leave\text{-}st}(s[a]\downarrow_{R^{\succ u}}) = \mathsf{leave\text{-}st}(s[\mathsf{true}]\downarrow_{R^{\succ u}}) \cup \mathsf{leave\text{-}st}(s[\mathsf{false}]\downarrow_{R^{\succ u}})$$

The wanted result follows from Proposition 5. □

We show the following technical proposition, that we use later in this section. Given a condition a and two terms t_l and t_r, we give sufficient conditions under which a persistent term in t_l or t_r is a persistent term in if a then t_l else t_r.

Proposition 22 (Persistent Term Lifting). *Consider the terms:*

$$C[\vec{a} \diamond \vec{b}] \qquad t_l \equiv B^l\left[\left(C_i^l[\vec{a}_i^l \diamond \vec{b}_i^l]\right)_i \diamond \left(D_j^l[\vec{c}_j^l \diamond \vec{t}_j^l]\right)_j\right] \qquad t_r \equiv B^r\left[\left(C_i^r[\vec{a}_i^r \diamond \vec{b}_i^r]\right)_i \diamond \left(D_j^r[\vec{c}_j^r \diamond \vec{t}_j^r]\right)_j\right]$$

where:

- *For every $x \in \{l, r\}$, i and j, the terms $\vec{a}_i^x, \vec{b}_i^x, \vec{c}_j^x, \vec{t}_j^x$ are if-free and in R-normal form.*
- *$\vec{a}, \vec{b}$ are if-free, in R-normal form and $(\vec{a} \cup \vec{b}) \cap \{\mathit{true}, \mathit{false}\} = \emptyset$.*
- *$\vec{b} \cap (\bigcup_{x\in\{l,r\},i} \vec{a}_i^x, \vec{b}_i^x) = \emptyset$ and $\vec{b} \cap (\bigcup_{x\in\{l,r\},j} \vec{c}_i^x) = \emptyset$.*
- *$\vec{a} \cap \vec{b} = \emptyset$.*

Then:

$$\mathsf{leave\text{-}st}(t_l\downarrow_R) \cup \mathsf{leave\text{-}st}(t_r\downarrow_R) \subseteq \mathsf{leave\text{-}st}\left((\mathit{if}\ C[\vec{a} \diamond \vec{b}]\ \mathit{then}\ t_l\ \mathit{else}\ t_r)\downarrow_R\right)$$

Proof. We prove this by induction on $|\vec{a}|$.

Base Case. If $|\vec{a}| = 0$ then $C[\vec{a} \diamond \vec{b}] \equiv b$, where b is if-free. Let $\succ_u$ be any total order on if-free conditions 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 t_l and t_r. By Proposition 5, we know that:

$$\mathsf{leave\text{-}st}(t_l\downarrow_R) = \mathsf{leave\text{-}st}(t_l\downarrow_{R_{\succ_u}}) = \mathsf{leave\text{-}st}\left(\left(D_l[\vec{a}_l \diamond \vec{t}_l]\right)\downarrow_{R_{\succ_u}}\right) \tag{17}$$

Using the fact that $(\vec{a}_i^l, \vec{b}_i^l)_i$ and $\vec{c}_j^l, \vec{t}_j^l)_j$ are if-free and in R-normal form, it is simple to show by induction on the length of the reduction that $\vec{a}_l \subseteq (\vec{a}_i^l, \vec{b}_i^l)_i, (\vec{c}_j^l)_j$. Since $b \notin (\bigcup_{x\in\{l,r\},i} \vec{a}_i^x, \vec{b}_i^x)$ and $b \notin (\bigcup_{x\in\{l,r\},j}, \vec{c}_j^x)$, this shows that $b \notin \vec{a}_l$. Similarly $\vec{a}_r \subseteq (\vec{a}_i^r, \vec{b}_i^r)_i, (\vec{c}_j^r)_j$ and $b \notin \vec{a}_r$.

$$(\text{if } b \text{ then } t_l \text{ else } t_r)\downarrow_{R^{\succ u}} \equiv \left(\text{if } b \text{ then } D_l[\vec{a}_l \diamond \vec{t}_l] \text{ else } D_r[\vec{a}_r \diamond \vec{t}_r]\right)\downarrow_{R^{\succ u}}$$

Since b is and if-free condition 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, since $b \notin \{\mathsf{true}, \mathsf{false}\}$ and since $b \notin \vec{a}_l \cup \vec{a}_r$, there is only one rule that may be applicable: if b then x else $x \longrightarrow x$. Therefore:

$$(\text{if } b \text{ then } t_l \text{ else } t_r)\downarrow_{R^{\succ_u}} \equiv \begin{cases} t_l \downarrow_{R^{\succ_u}} & \text{if } t_l =_{R^{\succ_u}} t_r \\ \text{if } b \text{ then } t_l \downarrow_{R^{\succ_u}} \text{ else } t_r \downarrow_{R^{\succ_u}} & \text{otherwise} \end{cases}$$

Which shows the wanted result.

Inductive Case. Assume that the result holds for m, and consider $\vec{a}, a$ of length $m+1$. First:

$$\text{if } C[\vec{a}, a \diamond \vec{b}] \text{ then } t_l \text{ else } t_r =_R \text{if } a \text{ then if } C[\vec{a}, \mathsf{true} \diamond \vec{b}] \text{ then } t_l \text{ else } t_r \\ \text{else if } C[\vec{a}, \mathsf{false} \diamond \vec{b}] \text{ then } t_l \text{ else } t_r$$

Let $s_l[]$ be a context such that $s_l[a] \equiv t_l$ and $a \notin \text{cond-st}(s_l[]\downarrow_R)$. Similarly, let $s_r[]$ be such that $s_r[a] \equiv t_r$ and $a \notin \text{cond-st}(s_r[]\downarrow_R)$. We are going to rewrite the then branch to replace any occurrence of a by true. Similarly, we rewrite the else branch to replace any occurrence of a by false.

Moreover, we get rid of true and false in $C[\vec{a}, \mathsf{true} \diamond \vec{b}]$ and $C[\vec{a}, \mathsf{false} \diamond \vec{b}]$. Let $C'[\vec{a}' \diamond \vec{b}']$ and $C''[\vec{a}'' \diamond \vec{b}'']$ be 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 $\vec{a}' \cup \vec{a}'' \subseteq \vec{a} \backslash \{a\}$ and $\vec{b}' \cup \vec{b}'' \subseteq \vec{b} \backslash \{a\}$. Then:

$$\text{if } C[\vec{a}, \mathsf{true} \diamond \vec{b}] \text{ then } t_l \text{ else } t_r =_R \boxed{\text{if } C'[\vec{a}' \diamond \vec{b}'] \text{ then } s_l[\mathsf{true}] \text{ else } s_r[\mathsf{true}]}$$

$$\text{if } C[\vec{a}, \mathsf{false} \diamond \vec{b}] \text{ then } t_l \text{ else } t_r =_R \boxed{\text{if } C''[\vec{a}'' \diamond \vec{b}''] \text{ then } s_l[\mathsf{false}] \text{ else } s_r[\mathsf{false}]}$$

We start by checking that the induction hypothesis on the red framed term. The first condition is trivial, we check the other:

- Since $\vec{a}' \subseteq \vec{a}, \vec{b}' \subseteq \vec{b}$ and $(\vec{a} \cup \vec{b}) \cap \{\mathsf{true}, \mathsf{false}\} = \emptyset$, we know that $(\vec{a}' \cup \vec{b}') \cap \{\mathsf{true}, \mathsf{false}\} = \emptyset$.
- The term $s_l[a]$ is obtained from t_l by replacing every occurrence of a by true. Hence, since $\mathsf{true} \notin \vec{b}, \vec{b}' \subseteq \vec{b}$ and:

  $$\vec{b} \cap (\bigcup_{x \in \{l,r\}, i} \vec{a}_i^x, \vec{b}_i^x) = \emptyset \qquad \vec{b} \cap (\bigcup_{x \in \{l,r\}, j} \vec{c}_i^x) = \emptyset$$

  We know that the third condition holds.
- Since $\vec{a}' \subseteq \vec{a}, \vec{b}' \subseteq \vec{b}$ and $\vec{a} \cap \vec{b} = \emptyset$, we know that $\vec{a}' \cap \vec{b}' = \emptyset$.

By applying the induction hypothesis, we deduce that:

$$\text{leave-st}(s_l[\mathsf{true}]\downarrow_R) \cup \text{leave-st}(s_r[\mathsf{true}]\downarrow_R) \\ \subseteq \boxed{\text{leave-st}(\text{if } C'[\vec{a}' \diamond \vec{b}'] \text{ then } s_l[\mathsf{true}] \text{ else } s_r[\mathsf{true}]\downarrow_R)}$$

Similarly, by applying the induction hypothesis on the rewriting of the blue framed term, we get:

$$\text{leave-st}(s_l[\mathsf{false}]\downarrow_R) \cup \text{leave-st}(s_r[\mathsf{false}]\downarrow_R) \\ \subseteq \boxed{\text{leave-st}(\text{if } C''[\vec{a}'' \diamond \vec{b}''] \text{ then } s_l[\mathsf{false}] \text{ else } s_r[\mathsf{false}]\downarrow_R)}$$

Finally, we apply again the induction hypothesis (with $m = 0$) to the term u below:

$$u \equiv \text{if } a \text{ then } \boxed{\text{leave-st}(\text{if } C'[\vec{a}' \diamond \vec{b}'] \text{ then } s_l[\text{true}] \text{ else } s_r[\text{true}] \downarrow_R)}$$
$$\text{else } \boxed{\text{leave-st}(\text{if } C''[\vec{a}'' \diamond \vec{b}''] \text{ then } s_l[\text{false}] \text{ else } s_r[\text{false}] \downarrow_R)}$$

We get that:

$$\text{leave-st}(u \downarrow_R) \supseteq \text{leave-st}\left(\boxed{\text{leave-st}(\text{if } C'[\vec{a}' \diamond \vec{b}'] \text{ then } s_l[\text{true}] \text{ else } s_r[\text{true}] \downarrow_R)} \downarrow_R\right)$$
$$\cup \text{leave-st}\left(\boxed{\text{leave-st}(\text{if } C''[\vec{a}'' \diamond \vec{b}''] \text{ then } s_l[\text{false}] \text{ else } s_r[\text{false}] \downarrow_R)} \downarrow_R\right)$$

By applying Proposition 21 twice, we know that:

$$\text{leave-st}(t_l \downarrow_R) \cup \text{leave-st}(t_r \downarrow_R) =$$
$$\text{leave-st}(s_l[\text{true}] \downarrow_R) \cup \text{leave-st}(s_r[\text{true}] \downarrow_R) \cup \text{leave-st}(s_l[\text{false}] \downarrow_R) \cup \text{leave-st}(s_r[\text{false}] \downarrow_R)$$

Hence we deduce that:

$$\text{leave-st}(t_l \downarrow_R) \cup \text{leave-st}(t_r \downarrow_R) \subseteq \text{leave-st}(u \downarrow_R) = \text{leave-st}(\text{if } C[\vec{a}, a \diamond \vec{b}] \text{ then } t_l \text{ else } t_r \downarrow_R) \quad \square$$

We are now ready to prove the main lemma of this section, which, under some conditions, shows that all leaf term γ of a term t has a persistent leaf.

Lemma 21. *Let s be a term of the form:*

$$s \equiv A\left[\vec{d} \diamond \left(B_l\left[\left(\beta_{i,l}\right)_i \diamond \left(\gamma_{j,l}\right)_j\right]\right)_l\right]$$

such that:

(i) *$\vec{d}$ are if-free and in R-normal form, and for every i, l,* $\text{cond-st}(\beta_{i,l} \downarrow_R) \cap \text{leave-st}(\beta_{j,l} \downarrow_R) = \emptyset$.

(ii) $\left(\vec{d} \cup \bigcup_{i,l} \text{leave-st}(\beta_{i,l} \downarrow_R)\right) \cap \{true, false\} = \emptyset$.

(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:*

$$\text{leave-st}(\zeta \downarrow_R) \cap \text{leave-st}(\zeta' \downarrow_R) = \emptyset$$

(iv) *For every l, i, j,* $\text{leave-st}(\beta_{i,l} \downarrow_R) \cap \text{leave-st}(\beta_{j,l} \downarrow_R) \neq \emptyset$ *implies that $\beta_{i,l} \equiv \beta_{j,l}$.*

(v) *For every l, the following couple of sets is well-nested:*

$$\left(\left\{\beta_{i,l} \downarrow_R |\ i\right\}, \left\{\gamma_{j,l} \downarrow_R |\ j\right\}\right)$$

then for every l, j, $\gamma_{j,l}$ contains a persistent term in s, i.e. $\text{leave-st}(\gamma_{j,l} \downarrow_R) \cap \text{leave-st}(s \downarrow_R) \neq \emptyset$.

Proof. We start by showing that the property holds when $\vec{d} = \emptyset$ and $A \equiv []$. We deal with the general case afterward.

Part 1. For all i, j, we let $C_i[]$, $D_j[]$ be if-contexts and $\vec{a}_i, \vec{b}_i, \vec{c}_j, \vec{t}_j$ be if-free terms in R-normal form such that:

$$\vec{a}_i \equiv \text{cond-st}(\beta_i \downarrow_R) \qquad \vec{b}_i \equiv \text{leave-st}(\beta_i \downarrow_R) \qquad \vec{c}_i \equiv \text{cond-st}(\gamma_j \downarrow_R) \qquad \vec{t}_i \equiv \text{leave-st}(\gamma_j \downarrow_R)$$

$$\beta_i \downarrow_R \equiv\ C_i[\vec{a}_i \diamond \vec{b}_i] \qquad\qquad \gamma_j \downarrow_R \equiv\ D_j[\vec{c}_j \diamond \vec{t}_j]$$

We know that:

$$s =_R B\left[\left(C_i[\vec{a}_i \diamond \vec{b}_i]\right)_i \diamond \left(D_j[\vec{c}_j \diamond \vec{t}_j]\right)_j\right]$$

satisfying conditions (i) to (v). We prove the proposition by structural induction on $B[]$.

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:

$$\text{leave-st}(s \downarrow_R) = \text{leave-st}(D[\vec{c} \diamond \vec{t}] \downarrow_R) \subseteq \vec{t}$$

Part 1: Inductive Case. Consider:

$$s \equiv \text{if } C_0[\vec{a}_0 \diamond \vec{b}_0] \text{ then } B^l \left[\left(C_i[\vec{a}_i \diamond \vec{b}_i] \right)_{i \in I^l} \diamond \left(D_j[\vec{c}_j \diamond \vec{t}_j] \right)_{j \in J^l} \right]$$
$$\text{else } B^r \left[\left(C_i[\vec{a}_i \diamond \vec{b}_i] \right)_{i \in I^r} \diamond \left(D_j[\vec{c}_j \diamond \vec{t}_j] \right)_{j \in J^r} \right]$$

Using the well-nested hypothesis, for every $j \in I^l \cup I^r$, there exist two if-context C'_j, C''_j such that:

$$C_j[\vec{a}_j \diamond \vec{b}_j] =_R \text{if } C_0[\vec{a}_0 \diamond \vec{b}_0] \text{ then } C'_j[\vec{a}'_j \diamond \vec{b}'_j] \text{ else } C''_j[\vec{a}''_j \diamond \vec{b}''_j]$$

where $\vec{a}'_j, \vec{a}''_j \subseteq \vec{a}_j \backslash \vec{b}_0$ and $\vec{b}'_j, \vec{b}''_j \subseteq \vec{b}_j$. Similarly, for every $j \in J^l \cup J^r$, there exist D'_j, D''_j such that:

$$D_j[\vec{c}_j \diamond \vec{t}_j] =_R \text{if } C_0[\vec{a}_0 \diamond \vec{b}_0] \text{ then } D'_j[\vec{c}'_j \diamond \vec{t}'_j] \text{ else } C''_j[\vec{c}''_j \diamond \vec{t}''_j]$$

where $\vec{c}'_j, \vec{c}''_j \subseteq \vec{a}_j \backslash \vec{b}_0$ and $\vec{t}'_j, \vec{t}''_j \subseteq \vec{t}_j$. Then:

$$s \equiv \text{if } C_0[\vec{a}_0 \diamond \vec{b}_0] \text{ then } \boxed{B^l \left[\left(C'_i[\vec{a}'_i \diamond \vec{b}'_i] \right)_{i \in I^l} \diamond \left(D'_j[\vec{c}'_j \diamond \vec{t}'_j] \right)_{j \in J^l} \right]} \; s_{\text{true}}$$
$$\text{else } \boxed{B^r \left[\left(C''_i[\vec{a}''_i \diamond \vec{b}''_i] \right)_{i \in I^r} \diamond \left(D''_j[\vec{c}''_j \diamond \vec{t}''_j] \right)_{j \in J^r} \right]} \; s_{\text{false}}$$

We want to show that for all $j \in J^l \cup J^r$, $\exists t \in \vec{t}_j . \, t \in \text{leave-st}(s \downarrow_R)$. Let $j \in J^l$ (the proof for $j \in J^r$ is similar). We are going to apply the induction hypothesis to s_{true} (for $j \in J^r$, we apply the induction hypothesis to s_{false}). Lets check that the premises hold for s_{true}:

- (i) and (ii) trivially hold.
- For (iii), we use the fact that we know that the property holds in s for every positions $\epsilon < p < p'$ in if [] then B^l else B^r, and the fact that for every $i \in I^l \cup I^r$, $\vec{b}'^i \subseteq \vec{b}^i$.
- Checking that (iv) holds is straightforward. Assume that there exists $i, j \in I^l$ 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:

$$C'_i[\vec{a}'_i \diamond \vec{b}'_i] \equiv C'_j[\vec{a}'_j \diamond \vec{b}'_j] \qquad C''_i[\vec{a}''_i \diamond \vec{b}''_i] \equiv C''_j[\vec{a}''_j \diamond \vec{b}''_j]$$

- 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, for every $(C, \mathcal{D}), (C', \mathcal{D}')$, if $(C, \mathcal{D})$ is well-nested and $C' \subseteq C \wedge \mathcal{D}' \subseteq \mathcal{D}$ then $(C', \mathcal{D}')$ is well-nested, we know that the following couple of sets is well-nested:

$$\left(\left\{ C'_i[\vec{a}'_i \diamond \vec{b}'_i] \mid i \in I^l \cup \{0\} \right\}, \left\{ D'_j[\vec{c}'_j \diamond \vec{t}'_j] \mid j \in J^l \right\}_j \right)$$

We can apply the induction hypothesis to s_{true}, which shows that for all $j \in J^l$, there exists $t \in \vec{t}'_j$ such that $t \in \text{leave-st}(s_{\text{true}} \downarrow_R)$. To conclude, we have to lift this to $\text{leave-st}(s \downarrow_R)$.

Let $S = I^l \cup I^r \cup \{0\} \cup J^l \cup J^r$, and $S' = S\backslash\{0\}$. We apply Proposition 22 to show that $t \in \text{leave-st}(s\downarrow_R)$. The only difficulty lies in showing that:

$$\vec{b}_0 \cap \left(\bigcup_{i\in S'} \vec{a}'_i, \vec{a}''_i, \vec{b}'_i, \vec{b}''_i, \vec{c}'_i, \vec{c}''_i\right) = \emptyset$$

We know that $b_0 \cap \left(\bigcup_{i\in S'} \vec{a}'_i, \vec{a}''_i, \vec{c}'_i, \vec{c}''_i\right) = \emptyset$ (since $\vec{a}'_i \subseteq \vec{a}_i\backslash\vec{b}_0, \dots$), so it only remains to show that:

$$\vec{b}_0 \cap \bigcup_{i\in S'} \vec{b}'_i, \vec{b}''_i = \emptyset \tag{18}$$

For every $i \in S'$, we know that $\vec{b}'_i \subseteq \vec{b}_i$ and $\vec{b}''_i \subseteq \vec{b}_i$. Hence, if $\vec{b}_0 \cap \vec{b}'_i \neq \emptyset$ or $\vec{b}_0 \cap \vec{b}''_i \neq \emptyset$ then $\vec{b}_i \cap \vec{b}_0 \neq \emptyset$. Since $C_0[]$ is at the root of s, we know using (iii) that $\vec{b}_i \cap \vec{b}_0 = \emptyset$. Hence (18) holds.

Part 2. For the general case, we just observe that we can take:

$$B[] \equiv A[([]_d)_{d\in\vec{d}} \diamond (B_l[])_l]$$

We only need to check that the property (i)-(v) are verified for $B[]$. Properties (i)-(iv) are straightforward. For (v), we only observe that, since $\vec{d}$ are if-free and in R-normal form, if $(C, \mathcal{D})$ is well-nested then $(C \cup \vec{d}, \mathcal{D})$ is well-nested. □

G PROOF CUT ELIMINATION

Consider a proof $P \vdash^{\text{npf}} t \sim t'$. Lemma 21 shows that, under some conditions, any normalized basic term $\gamma \leq_{\text{l}}^{\epsilon,l} (t, P)$ has a persistent leaf in t, i.e. $\text{leave-st}(\gamma\downarrow_R) \cap \text{leave-st}(t\downarrow_R) \neq \emptyset$. To apply this lemma, we need to have a proof P satisfying the hypothesis of Lemma 21. We give simplified version of these conditions below:

(i) for every $\beta, \beta' \leq_{\text{c}}^{\epsilon,l} (t, P)$, we have $\text{cond-st}(\beta\downarrow_R) \cap \text{leave-st}(\beta\downarrow_R) = \emptyset$.
(ii) $\left(\bigcup_{\beta\leq_{\text{c}}^{\epsilon,l}(t,P)} \text{leave-st}(\beta\downarrow_R)\right) \cap \{\text{true}, \text{false}\} = \emptyset$.
(iii) For every $\beta, \beta' \leq_{\text{c}}^{\epsilon,l} (t, P)$ and positions $p < p'$ in t such that $t_{|p} \equiv \beta$ and $t_{|p'} \equiv \beta'$, we have:

$$\text{leave-st}(\beta\downarrow_R) \cap \text{leave-st}(\beta'\downarrow_R) = \emptyset$$

(iv) For every $\beta, \beta' \leq_{\text{c}}^{\epsilon,l} (t, P)$, if $\text{leave-st}(\beta\downarrow_R) \cap \text{leave-st}(\beta'\downarrow_R) \neq \emptyset$ then $\beta \equiv \beta$.
(v) The following couple of sets is well-nested:

$$\left(\left\{\beta\downarrow_R |\ \beta \leq_{\text{c}}^{\epsilon,l} (t, P)\right\}, \left\{\gamma\downarrow_R |\ \gamma \leq_{\text{l}}^{\epsilon,l} (t, P)\right\}\right)$$

For each property above, we give the proposition or lemma proving that it holds, or we announce in which section we will prove it.

(i) In other word, this means that every normalized basic terms has disjoint conditions and leaves. We will prove this in Section G.2.
(ii) For this to hold, we need to prove that, w.l.o.g., we can assume that true and false do not appear in the leaves of normalized basic terms. We will show this in Section G.1.
(iii) This requires two non-trivial proof cut, which we explain in Section G.3. It relies on Lemma 2.
(iv) This is a consequence of Proposition 17, which we already proved.
(v) We showed that these sets are well-nested in Lemma 19.

The rest of this section is organized as follows: in Section G.1 we deal with (ii), by showing that we can assume that true and false do not appear in proof in normal proof form; in Section G.2 we prove that conditions and leaves of basic terms are disjoints, which we need for (i); in Section G.3, we give examples of proof cut elimination used to obtain (iii); finally, in Section G.4, we use Lemma 21 to prove that we can assume, w.l.o.g., that every leaf term appearing t has a persistent leaf in t.

G.1 Removing True and False From Basic Terms

In this section, we prove that we can assume, w.l.o.g., that true and false do not appear in the leaves of normalized basic terms.

Key Observation. Let s be an if-free in R-normal form, s can be rewritten into a complex term u:

$$u \equiv C\left[\left(D_i[\vec{a}_i \diamond \vec{b}_i]\right)_i \diamond \vec{t}\right]$$

that is not if-free. Basically, the following proposition shows that as long as the term u does not contain true and false conditions, 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:

$$\mathsf{if\ true\ then}\ v\ \mathsf{else}\ w \to w \qquad\qquad \mathsf{if\ false\ then}\ v\ \mathsf{else}\ w \to w$$

Proposition 23. *For all if-free term s in R-normal form, if $s =_R C\left[\left(D_i[\vec{a}_i \diamond \vec{b}_i]\right)_i \diamond \vec{t}\ \right]$ where:*

- *$\vec{t} \cup \bigcup_i(\vec{a}_i \cup \vec{b}_i)$ are if-free and in R-normal form.*
- *For every i such that $D_i[\vec{a}_i \diamond \vec{b}_i]$ is a term appearing on the left-most (resp. right-most) branch of C, we have that $\mathsf{false} \notin \vec{a}_i \cup \vec{b}_i$ (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 $\to_R$, 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 would like to prove that for every b, there exists no derivation of $b \sim \mathsf{true}$ or $b \sim \mathsf{false}$. Such derivations would be problematic since true and false are conditions of constant size, but b could be of any size (and we are trying to bound all conditions appearing in a proof). Also, the else branch of a true condition can contain anything and is, a priori, not bounded by the proof conclusion. Using Proposition 23 we proved above, we show that there exists no proof of $b \sim \mathsf{true}$ or false, as long as b is if-free and the proof is in the fragment $\mathcal{A}_{\succ}$.

Proposition (9). *Let b an if-free condition 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 condition in R-normal form which is not true and false, and let P be such that $P \vdash^{\mathsf{npf}} b \sim \mathsf{false}$. We choose the condition b such that its proof 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 $b_0 \leq^{\mathsf{h}}_{\mathsf{cs}} (\mathsf{false}, P)$, $b_0 \neq_R \mathsf{false}$. Let $H = \mathsf{cs\text{-}pos}(P)$. We now focus on the left-most branch of the proof.

First Part. 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 $(\beta, \beta') \leq^{\epsilon,l}_{\mathsf{c}\sim\mathsf{c}} (b \sim \mathsf{false}, P)$. 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 $\mathsf{leave\text{-}st}(\beta \downarrow_R) = \mathsf{leave\text{-}st}(\mathsf{false} \downarrow_R) = \{\mathsf{false}\}$. Since β is a normalized basic condition (for the CCA_2 trace $\mathcal{S}$ of its branch in P), and since false is a normalized basic condition, using Proposition 17 we have $\beta \equiv \mathsf{false}$.

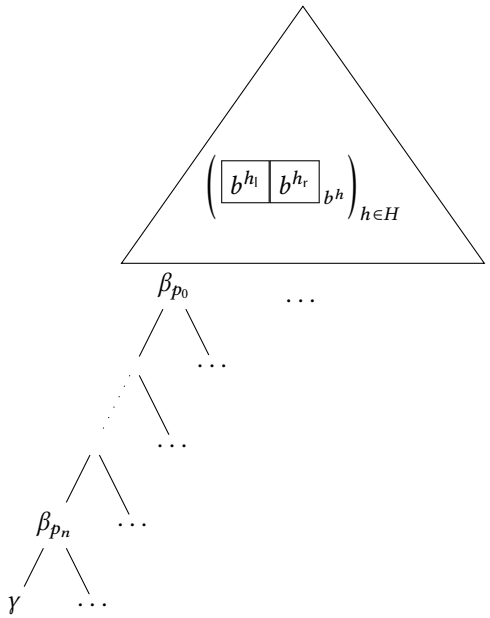


Fig. 15. Shape of the Term in the Proof of Proposition 9

There exists a derivation of $\beta \sim \beta'$ in $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA}_2)$. Since $\beta \equiv$ false, no rules in $\mathsf{FA_s}$ are applied. Therefore the derivation is only an application of CCA_2, which is not possible. Similarly we do not have $\beta \neq_R$ false and $\beta' =_R$ false.

Part 2. Using Proposition 17 we know that $\beta \neq_R$ false implies that for all $u \in$ leave-st$(\beta \downarrow_R)$, $u \not\equiv$ false. Moreover, for any term w, $w \downarrow_R$ does not contain false in its conditions (or we could apply if false then x else $y \longrightarrow y$). Hence for every $a \in$ cond-st$(\beta \downarrow_R)$, $a \not\equiv$ false.

We let $(\gamma, \gamma') \leq_{\mathsf{l}}^{\epsilon,l} (b \sim \text{false}, P)$ be the left-most elements (as shown in Figure 15). For all $a \in$ cond-st$(\gamma \downarrow_R)$, $a \not\equiv$ false. Hence if we let $u_0 \in$ leave-st$(\gamma \downarrow_R)$ be the left-most leave element of $\gamma \downarrow_R$, then by Proposition 23 we know that $u_0 \equiv b$. Recall that $b \neq_R$ false.

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 17 we get that $\gamma' \equiv$ false. Since there exists a derivation of $\gamma \sim \gamma'$ in $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA}_2)$, no rule in $\mathsf{FA_s}$ is applied. Therefore the derivation is only an application of CCA_2. Contradiction. □

Thanks to this proposition, we can ensure that any proof P of $t \sim t'$ does not contain a $\mathsf{CS}_\square$ or $\overline{\mathsf{BFA}}$ application on true or false: if we have a $\mathsf{CS}_\square$ or $\overline{\mathsf{BFA}}$ application on (true, true) or (false, false) then there is an easy proof cut elimination, and the previous proposition ensures that there are no $\mathsf{CS}_\square$ or $\overline{\mathsf{BFA}}$ applications on (true, b), (b, true), (false, b) or (b, false) (with $b \neq_R$ false, true).

Proposition 24. *For all $P \vdash^{npf} t \sim t'$, there exists P' such that $P' \vdash^{npf} t \sim t'$ and for all $l \in$ label$(P'), h \in$ index$(P'), x \in \{l, r\}$ we have:*

$$\forall \beta \in \left((\leq_c^{h_x, l} \cup \leq_{cs}^{h_x})(t, P')\right) \cup \left((\leq_c^{h_x, l} \cup \leq_{cs}^{h_x})(t', P')\right), \quad \{\mathit{false}, \mathit{true}\} \cap \text{leave-st}(\beta \downarrow_R) = \emptyset$$

Proof. Through simple proof cut eliminations, We can construct a proof P' from P such that:

$$\{(\mathsf{true},\mathsf{true}),(\mathsf{false},\mathsf{false})\} \cap (\leq^{h_{\mathsf{x}},l}_{\mathsf{c}\sim\mathsf{c}} (t \sim t', P) \cup \leq^{h_{\mathsf{x}}}_{\mathsf{cs}\sim\mathsf{cs}} (t \sim t', P)) = \emptyset$$

Then using Proposition 9 we know that for all:

$$(\beta,\beta') \in (\leq^{h_{\mathsf{x}},l}_{\mathsf{c}\sim\mathsf{c}} (t \sim t', P) \cup \leq^{h_{\mathsf{x}}}_{\mathsf{cs}\sim\mathsf{cs}} (t \sim t', P))$$

the conditions 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 using Proposition 17 we know that for every $u \in \mathsf{leave\text{-}st}(\beta\downarrow_R)$, $u \not\equiv \mathsf{false}$ (idem with true). □

G.2 Basic Terms have Disjoints Conditions and Leaves

We now prove that every normalized basic terms has disjoint conditions and leaves.Let β be a normalized basic terms. First. we show that every condition term b in $\mathsf{cond\text{-}st}(\beta\downarrow_R)$ is the leaf of another normalized basic term β', which is a strict subterm of β. Therefore, if $\mathsf{cond\text{-}st}(\beta\downarrow_R) \cap \mathsf{leave\text{-}st}(\beta\downarrow_R) \neq \emptyset$ then there exists β' such that $\mathsf{leave\text{-}st}(\beta\downarrow_R) \cap \mathsf{leave\text{-}st}(\beta'\downarrow_R) \neq \emptyset$. Using Proposition 17, we deduce that $\beta \equiv \beta'$, which contradicts the fact that β' is a strict sub-term of β.

First, we define the set of normalized basic conditions appearing in a term t.

Definition 63. For all term t, we let $<^{\mathcal{S}}_{\mathsf{bc}} t$ be the set of $\mathcal{S}$-normalized basic condition appearing in t, defined inductively by:

- If t is a $\mathcal{S}$-normalized simple term $C[\vec{b} \diamond \vec{u}]$, then:
$$<^{\mathcal{S}}_{\mathsf{bc}} t \;=\; \vec{b} \;\cup\; \left(<^{\mathcal{S}}_{\mathsf{bc}} \vec{b}\right) \;\cup\; \left(<^{\mathcal{S}}_{\mathsf{bc}} \vec{u}\right)$$
- If t is a $\mathcal{S}$-normalized basic term $B[\vec{w},(\alpha_i)_i,(\mathsf{dec}_j)_j]$, then:
$$<^{\mathcal{S}}_{\mathsf{bc}} t \;=\; \bigcup_i <^{\mathcal{S}}_{\mathsf{bc}} \alpha_i \;\cup\; \bigcup_j <^{\mathcal{S}}_{\mathsf{bc}} \mathsf{dec}_j$$
- For every $\mathcal{S}$-encryption oracle call $t \equiv \{u\}^r_{\mathsf{pk}}$, then:
$$<^{\mathcal{S}}_{\mathsf{bc}} t \;=\; <^{\mathcal{S}}_{\mathsf{bc}} u$$
- For every $\mathcal{S}$-decryption oracle call $C[\vec{b} \diamond \vec{u}]$, let s, sk be such that s is if-free and the terms in $\vec{u}$ are of the form $\mathbf{0}(\mathsf{dec}(s[(\alpha_i),(\mathsf{dec}_j)_j],\mathsf{sk}))$ or $\mathsf{dec}(s[(\alpha_i),(\mathsf{dec}_j)_j],\mathsf{sk})$. Then:
$$<^{\mathcal{S}}_{\mathsf{bc}} t \;=\; \vec{b} \;\cup\; \left(<^{\mathcal{S}}_{\mathsf{bc}} \vec{b}\right) \;\cup\; \bigcup_i <^{\mathcal{S}}_{\mathsf{bc}} \alpha_i \;\cup\; \bigcup_j <^{\mathcal{S}}_{\mathsf{bc}} \mathsf{dec}_j$$

We show that the over-approximated set of conditions $\overline{\mathsf{cond\text{-}st}}(\beta)$ is exactly the over-approximated set of leaves of the normalized basic conditions that are subterm of β.

Proposition 25. *For every 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:*

$$\overline{\mathsf{cond\text{-}st}}(\beta) = \bigcup_{u <^{\mathcal{S}}_{bc} \beta} \overline{\mathsf{leave\text{-}st}}(u)$$

Proof. We prove this by induction on the order $<^{\mathcal{S}}_{\mathsf{ind}}$, which, we recall, is the order stemming from $\mathcal{S}$-normalized basic terms, $\mathcal{S}$-normalized simple terms, $\mathcal{S}$-decryption oracle calls or $\mathcal{S}$-encryption oracle calls mutually inductive definitions.

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 s 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 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.

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 $<^{\mathcal{S}}_{\mathsf{bc}}$:

$$\begin{aligned}
\overline{\mathsf{cond\text{-}st}}(\beta) &= \bigcup_i \overline{\mathsf{cond\text{-}st}}(\alpha_i) \quad \cup \quad \bigcup_j \overline{\mathsf{cond\text{-}st}}(\mathsf{dec}_i) && \text{(By definition of } \overline{\mathsf{cond\text{-}st}}(\cdot)) \\
&= \bigcup_i \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \alpha_i} \overline{\mathsf{leave\text{-}st}}(u) \quad \cup \quad \bigcup_j \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \mathsf{dec}_j} \overline{\mathsf{leave\text{-}st}}(u) && \text{(By induction hypothesis)} \\
&= \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \beta} \overline{\mathsf{leave\text{-}st}}(u) && \text{(By definition of } <^{\mathcal{S}}_{\mathsf{bc}})
\end{aligned}$$

- $\mathcal{S}$-decryption oracle call: t is of the form $C[\vec{g} \diamond \vec{u}\,]$, 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 s is if-free. Then:

$$\begin{aligned}
\overline{\mathsf{cond\text{-}st}}(\beta) &= \bigcup_i \overline{\mathsf{cond\text{-}st}}(\alpha_i) \quad \cup \bigcup_j \overline{\mathsf{cond\text{-}st}}(\mathsf{dec}_i) \quad \cup \overline{\mathsf{cond\text{-}st}}(\vec{g}) \quad \cup \overline{\mathsf{leave\text{-}st}}(\vec{g}) \\
&\qquad\qquad \text{(By definition of } \overline{\mathsf{cond\text{-}st}}(\cdot)) \\
&= \bigcup_i \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \alpha_i} \overline{\mathsf{leave\text{-}st}}(u) \cup \bigcup_j \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \mathsf{dec}_j} \overline{\mathsf{leave\text{-}st}}(u) \cup \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \vec{g}} \overline{\mathsf{leave\text{-}st}}(u) \cup \overline{\mathsf{leave\text{-}st}}(\vec{g}) \\
&\text{(By induction hypothesis: remark that guards in } \vec{g} \text{ are } \mathcal{S}\text{-normalized basic terms s.t. } \vec{g} \leq^{\mathcal{S}}_{\mathsf{bt}} \beta) \\
&= \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \beta} \overline{\mathsf{leave\text{-}st}}(u) && \text{(By definition of } <^{\mathcal{S}}_{\mathsf{bc}})
\end{aligned}$$

- $\mathcal{S}$-encryption oracle call: t is of the form $\{s\}^r_{\mathsf{pk}}$, then:

$$\begin{aligned}
\overline{\mathsf{cond\text{-}st}}(\beta) &= \overline{\mathsf{cond\text{-}st}}(s) && \text{(By definition of } \overline{\mathsf{cond\text{-}st}}(\cdot)) \\
&= \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} s} \overline{\mathsf{leave\text{-}st}}(u) && \text{(By induction hypothesis)} \\
&= \bigcup_{u <^{\mathcal{S}}_{\mathsf{bc}} \beta} \overline{\mathsf{leave\text{-}st}}(u) && \text{(By definition of } <^{\mathcal{S}}_{\mathsf{bc}})
\end{aligned}$$

- $\mathcal{S}$-normalized simple term: t is of the form $C[\vec{b} \diamond \vec{v}\,]$. Then:

$$\overline{\mathsf{cond\text{-}st}}(\beta) = \overline{\mathsf{cond\text{-}st}}(\vec{b}) \quad \cup \overline{\mathsf{cond\text{-}st}}(\vec{v}) \quad \cup \overline{\mathsf{leave\text{-}st}}(\vec{b}) \quad \text{(By definition of } \overline{\mathsf{cond\text{-}st}}(\cdot))$$

$$= \bigcup_{u <^{\mathcal{S}}_{\mathrm{bc}} \vec{b}} \overline{\text{leave-st}}(u) \cup \bigcup_{u <^{\mathcal{S}}_{\mathrm{bc}} \vec{v}} \overline{\text{leave-st}}(u) \cup \overline{\text{leave-st}}(\vec{b}) \qquad \text{(By induction hypothesis)}$$

$$= \bigcup_{u <^{\mathcal{S}}_{\mathrm{bc}} \beta} \overline{\text{leave-st}}(u) \qquad \text{(By definition of } <^{\mathcal{S}}_{\mathrm{bc}}) \quad \square$$

We can now prove that every normalized basic terms has disjoint conditions and leaves, using Proposition 17 and the result above.

Proposition 26. *Let $P \vdash^{npf} t \sim t'$. Then for all h, l for all $\beta \leq^{h,l}_{bt} (t, P)$:*

$$\overline{\text{cond-st}}(\beta) \cap \overline{\text{leave-st}}(\beta) = \emptyset$$

Proof. Let h, l and $\beta \leq^{\mathsf{h},l}_{\mathrm{bt}} (t, P)$ be such that $\overline{\text{cond-st}}(\beta) \cap \overline{\text{leave-st}}(\beta) \neq \emptyset$. By Proposition 25 this means that there exists a $\mathcal{S}^P_l$-normalized basic term $u <^{\mathcal{S}_l}_{\mathrm{bc}} \beta$ such that $\overline{\text{leave-st}}(u) \cap \overline{\text{leave-st}}(\beta) \neq \emptyset$.

By Proposition 17, $u \equiv \beta$. But $u <^{\mathcal{S}_l}_{\mathrm{bc}} \beta$ implies that u is a strict subterm of β. Absurd. $\square$

G.3 Proof Cuts on Branches

For the hypothesis (iii) of Lemma 21 to hold, we need to make sure that the same condition never appear twice in the same branch[11]. Therefore, we need to show that if some normalized basic term β appears twice in the same branch, then there is a proof cut. We have three possibilities:

- The two occurrences of β are involved in $\overline{\mathsf{BFA}}$ applications.
- The two occurrences of β are involved in $\mathsf{CS}_\square$ applications.
- One occurrence of β is with an $\overline{\mathsf{BFA}}$ application, the other with a $\mathsf{CS}_\square$ applications.

The first two cases have already dealt with in Section 8. We deal with the cross case later.

Before continuing, we give the proof of Lemma 11, which we omitted in the body. We recall the lemma statement below:

Lemma (11). *For all a, a', b, c such that their R-normal form is if-free and $a =_R a'$, if there exists a proof P such that $P \vdash^{npf} a, a' \sim b, c$, 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 $P \vdash^{\mathsf{npf}} a, a' \sim b, c$. Using Proposition 24, we can assume that for every $h \in \mathsf{index}(P), l, \mathsf{x}$:

$$\forall \beta \in \left((\leq^{h_{\mathsf{x}},l}_{\mathsf{c}} \cup \leq^{h_{\mathsf{x}},l}_{\mathsf{cs}})(t, P')\right) \cup \left((\leq^{h_{\mathsf{x}},l}_{\mathsf{c}} \cup \leq^{h_{\mathsf{x}},l}_{\mathsf{cs}})(t', P')\right), \quad \{\mathsf{false}, \mathsf{true}\} \cap \mathsf{leave\text{-}st}(\beta \downarrow_R) = \emptyset$$

Let $(\gamma, \gamma') \leq^{\epsilon,l}_{\mathsf{l}} (t \sim t', P)$ be the left-most elements of t and t'. By Proposition 23 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 $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA}_2)$, we know that there exist $\mathcal{S}^P_l$-normalized basic terms γ_1, γ_2 and $\mathcal{S}'^P_l$-normalized basic terms γ'_1, γ'_2 such that $\gamma =_R \langle \gamma_1\,,\, \gamma_2\rangle$, $\gamma' =_R \left\langle \gamma'_1\,,\, \gamma'_2\right\rangle$, and $\gamma_1, \gamma_2 \sim \gamma'_1, \gamma'_2$ is provable in $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA}_2)$.

Moreover $a \in \mathsf{leave\text{-}st}(\gamma_1 \downarrow_R)$ and $a \in \mathsf{leave\text{-}st}(\gamma_2 \downarrow_R)$, hence $\mathsf{leave\text{-}st}(\gamma_1 \downarrow_R) \cap \mathsf{leave\text{-}st}(\gamma_2 \downarrow_R) \neq \emptyset$. By Proposition 17 we deduce that $\gamma_1 \equiv \gamma_2$.

Therefore there exists a proof of $\gamma_1, \gamma_1 \sim \gamma'_1, \gamma'_2$ in $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \mathsf{CCA}_2)$. By Lemma 2, $\gamma'_1 \equiv \gamma'_2$. We conclude by observing that since $\langle b\,,\, c\rangle$ is the let-most element of $\gamma' \downarrow_R$, b and c are the left-most element of, respectively, γ'_1 and γ'_2. Therefore $b \equiv c$. $\square$

[11] Indeed, we recall that Proposition 17 shows that if $\mathsf{leave\text{-}st}(\beta \downarrow_R) \cap \mathsf{leave\text{-}st}(\beta' \downarrow_R) \neq \emptyset$ then $\beta \equiv \beta'$.

G.4 Main Lemma

Definition 64. 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 conditions and $d_0, \dots, d_n$ (the directions) are in $\{\mathsf{then}, \mathsf{else}\}$.

Two directed paths ${}^{\delta}\vec{\rho}$ and ${}^{\delta}\vec{\rho}\,'$ 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, resp., ${}^{\delta}\vec{\rho}$ and ${}^{\delta}\vec{\rho}\,'$, are equal.

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

Example 23. Let s be the term of Example 6, which we recall below:

$$\begin{array}{l}\mathsf{if}\ b_1\ \mathsf{then}\ \mathsf{if}\ b_2\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2\\ \qquad\qquad \mathsf{else}\ t_3\end{array}$$

Then ${}^{\delta}\vec{\rho} = (b_1, \mathsf{then}), (b_2, \mathsf{else})$ is the directed path corresponding to the branch starting at the root of s and ending at the term t_2. Moreover, $\vec{\rho} = b_1, b_2$. ◇

Definition 65. Let $P \vdash^{\mathsf{npf}} t \sim t'$, we know that t is of the form:

$$t \equiv C\left[\left(\boxed{b^{h_{\mathsf{l}}}\,\middle|\,b^{h_{\mathsf{r}}}}_{\,b^h}\right)_{h\in H} \diamond \left(D_l\left[(\beta)_{\beta \le_{\mathsf{c}}^{\epsilon,l}(t,P)} \diamond (\gamma)_{\gamma \le_{\mathsf{l}}^{\epsilon,l}(t,P)}\right]\right)_{l\in L}\right]$$

For all l, we let:

- ${}^{\delta}\mathsf{cs\text{-}path}^{\epsilon,l}(t, P)$ be the directed path of condition 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 conditions 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:

$$\begin{array}{lll} & {}^{\delta}\mathsf{cs\text{-}path}^{h_{\mathsf{x}},l}(t, P) & = {}^{\delta}\mathsf{cs\text{-}path}^{\epsilon,l}(b, \mathsf{extract}_{\mathsf{x}}(h, P))\\ \text{and} & {}^{\delta}\mathsf{cs\text{-}path}_{\sim}^{h_{\mathsf{x}},l}(t \sim t', P) & = {}^{\delta}\mathsf{cs\text{-}path}_{\sim}^{\epsilon,l}(b \sim b', \mathsf{extract}_{\mathsf{x}}(h, P))\end{array}$$

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

We let the depth of a position h in P to be the number of nested applications of the $\mathsf{CS}_{\square}$ rule to h.

Definition 66. sym]if-depth$_P$ Let $P \vdash^{\mathsf{npf}} t \sim t'$. For every $h \in \mathsf{index}(P)$, we let $\mathsf{if\text{-}depth}_P(h)$ be the depth of h in P, defined by:

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

Lemma 22. *Let $P \vdash^{npf} t \sim t'$. There exists P' such that $P' \vdash^{npf} t \sim t'$ and for all $h \in index(P')$ with $h \neq \epsilon$, for all $x \in \{l, r\}$, if we let $h = h_x$ and $P^h = extract_x(h, P')$ be the proof of $b^h \sim b'^h$ then for all $l \in label(P^h)$:*

(a) The proof P^h does not use the $\{\overline{BFA}(b, b')\}$ rules.
(b) $cs\text{-}path^{h,l}(t, P)$ (resp. $cs\text{-}path^{h,l}(t', P)$) does not contain two occurrences of the same condition.
(c) For all $\gamma \le_l^{h,l} (t, P')$, $(b^h \downarrow_R) \in \mathsf{leave\text{-}st}(\gamma \downarrow_R)$ and for all $\gamma' \le_l^{h,l} (t', P')$, $(b'^h \downarrow_R) \in \mathsf{leave\text{-}st}(\gamma' \downarrow_R)$.
(d) For all $\beta \le_c^{\epsilon,l} (t, P')$, $\mathsf{leave\text{-}st}(\beta \downarrow_R) \cap cs\text{-}path^{\epsilon,l}(t, P) = \emptyset$ (same for t').
(e) For all $\gamma \le_l^{\epsilon,l} (t, P')$, $\mathsf{leave\text{-}st}(t \downarrow_R) \cap \mathsf{leave\text{-}st}(\gamma \downarrow_R) \neq \emptyset$ (same for t').

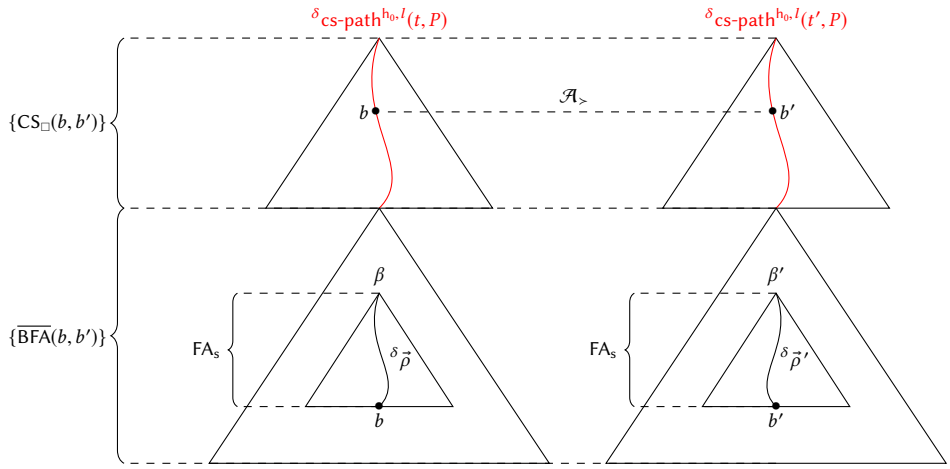


Fig. 16. Corresponding occurrences of b and b' in the proof of Lemma 22

Proof. Using Proposition 24, 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:

$$\forall \beta \in \left((\leq_{\mathsf{c}}^{h_{\mathsf{x}},l} \cup \leq_{\mathsf{cs}}^{h_{\mathsf{x}},l})(t,P)\right) \cup \left((\leq_{\mathsf{c}}^{h_{\mathsf{x}},l} \cup \leq_{\mathsf{cs}}^{h_{\mathsf{x}},l})(t',P)\right), \quad \{\mathsf{false},\mathsf{true}\} \cap \mathsf{leave\text{-}st}(\beta\downarrow_R) = \emptyset \tag{19}$$

First, we rewrite the proof P so that all CS applications are of the form:

$$\frac{b,(u_i)_i \sim b',(u'_i)_i \quad b,(v_i)_i \sim b',(v'_i)_i}{(\mathsf{if}\ b\ \mathsf{then}\ u_i\ \mathsf{else}\ v_i)_i \sim (\mathsf{if}\ b'\ \mathsf{then}\ u'_i\ \mathsf{else}\ v'_i)_i}\ \mathsf{CS} \tag{20}$$

We prove by induction on n, starting with the inner-most CS conditions, that there exists P such that $P \vdash^{\mathsf{npf}} t \sim t'$, (19) 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 condition.
(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 if-depth in the proof of $t \sim t'$:

$$n_{\max} = \max_{h \in \mathsf{index}(P)} \mathsf{if\text{-}depth}_P(h)$$

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 (20).

Inductive Case: Assume that the property holds for $n+1$ and let us show that it holds for n.

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 $\mathsf{x}_0 \in \{\mathsf{l}, \mathsf{r}\}$ and $\mathsf{h}_0 = h_{0\mathsf{x}_0}$. 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 $(\beta, \beta') \leq_{\mathsf{c}\sim\mathsf{c}}^{\mathsf{h}_0, l} (t \sim t', P)$ s.t.:

- $b' \in \mathsf{leave\text{-}st}(\beta' \downarrow_R)$.
- There exists a directed path ${}^{\delta}\vec{\rho}$ (resp. ${}^{\delta}\vec{\rho}'$) of the conditions occurring from the root of $\beta \downarrow_R$ (resp. $\beta' \downarrow_R$) to a leave b (resp. b') 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 Figure 16.

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 $h \in \mathsf{cs\text{-}pos}(\mathsf{extract}_{\mathsf{x}_0}(h_0, P))$ 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:

$$a \equiv C\left[(b^g)_{g \in H} \diamond (\gamma_{l_a})_{l_a}\right] \qquad a' \equiv C\left[(b'^g)_{g \in H} \diamond (\gamma'_{l_a})_{l_a}\right]$$

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 17 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 $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \overline{\mathsf{CCA}_2})$. 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.

We now prove the second bullet point. By induction hypothesis (iv) we know that:

$${}^{\delta}\mathsf{cs\text{-}path}^{\mathsf{h}_0, l}(t, P) \supseteq {}^{\delta}\mathsf{cs\text{-}path}^{h_{\mathsf{x}}, l}(t, P) \qquad {}^{\delta}\mathsf{cs\text{-}path}^{\mathsf{h}_0, l}(t', P) \supseteq {}^{\delta}\mathsf{cs\text{-}path}^{h_{\mathsf{x}}, l}(t', P)$$

By definition of $\vec{\rho}$, $\mathsf{cond\text{-}st}(\gamma_l \downarrow_R) = \mathsf{cond\text{-}st}(\beta \downarrow_R) \supseteq \vec{\rho}$. We can do better, and obtained an inclusion in the *directed* condition path. First, we know that:

- $a \equiv C\left[(b^g)_{g \in H} \diamond \left(\gamma_{l_a}\right)_{l_a}\right]$, $a =_R b$ and b is if-free and in R-normal form.
- Invariant (ii) holds, hence ${}^{\delta}\mathsf{cs\text{-}path}^{h_{\mathsf{x}}, l}(t, P)$ does not contain two occurrences of the same condition.
- ${}^{\delta}\mathsf{cs\text{-}path}^{h_{\mathsf{x}}, l}(t, P)$ does not contain true and false.

The existence of a decomposition as described above is invariant by (chunks of) $\rightarrow_{R^{\succ u}}$ reductions, for a well-chosen ordering $\succ_u$. At the end of the reduction, we have b. By looking at the reduction backward, we see that b is a leaf of $\gamma_l \downarrow_{R^{\succ u}}$, such that the directed path ${}^{\delta}\vec{\rho}$ from the root of $\gamma_l \downarrow_{R^{\succ u}}$ to b is included in the path from the root of a to γ_l.

We deduce that ${}^{\delta}\vec{\rho} \subseteq {}^{\delta}\mathsf{cs\text{-}path}^{h_{\mathsf{x}}, l}(t, P)$. By consequence, ${}^{\delta}\vec{\rho} \subseteq {}^{\delta}\mathsf{cs\text{-}path}^{\mathsf{h}_0, l}(t, P)$. Similarly we show that ${}^{\delta}\vec{\rho}' \subseteq {}^{\delta}\mathsf{cs\text{-}path}^{\mathsf{h}_0, l}(t', P)$.

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:

$${}^{\delta}\vec{\rho} \subseteq {}^{\delta}\mathsf{cs\text{-}path}^{\mathsf{h}_0, l}(t, P) \qquad {}^{\delta}\vec{\rho}' \subseteq {}^{\delta}\mathsf{cs\text{-}path}^{\mathsf{h}_0, l}(t', P)$$

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 (19) and the induction invariants.

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 11 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 preserves invariant (iv): this follows from the fact that invariant (iv) holds for P at depth $n+1$ and that the cut takes place at depth n.

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:

$$b^{\mathsf{h}_0} \equiv C\left[\left(\boxed{b^{h_{\mathsf{l}}}}\boxed{b^{h_{\mathsf{r}}}}_{b^h}\right)_{h\in H^{\mathsf{h}_0}} \diamond \left(D_l^{\mathsf{h}_0}\left[(\beta)_{\beta \leq_{\mathsf{c}}^{\mathsf{h}_0,l}(t,P)} \diamond (\gamma)_{\gamma \leq_{\mathsf{l}}^{\mathsf{h}_0,l}(t,P)}\right]\right)_{l\in L^{\mathsf{h}_0}}\right]$$

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 $\gamma \leq_{\mathsf{l}}^{\mathsf{h}_0,l} (t, P)$, 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 Lemma 21 hold for b^{h_0} (then we do the same thing with b'^{h_0} to show that for all $\gamma' \leq_{\mathsf{l}}^{\mathsf{h}_0,l} (t', P)$ we have $b'^{\mathsf{h}_0} \downarrow_R \in \mathsf{leave\text{-}st}(\gamma' \downarrow_R)$):

- (21.i): the only difficulty lies in proving that for all $\beta \leq_{\mathsf{c}}^{\mathsf{h}_0,l} (t, P)$, $\mathsf{cond\text{-}st}(\beta \downarrow_R) \cap \mathsf{leave\text{-}st}(\beta \downarrow_R) = \emptyset$, which is shown in Proposition 26.
- (21.ii): this is a consequence of the fact that (19) holds for P.
- (21.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 $D_l^{\mathsf{h}_0} \times D_l^{\mathsf{h}_0}$ we have a proof cut elimination (which we already described in Section G.3): let $p < p'$ be the positions in b^{h_0} of $\beta_0, \beta_1 \leq_{\mathsf{c}}^{\mathsf{h}_0,l} (t, P)$ on the same branch such that $\mathsf{leave\text{-}st}(\beta_0) \cap \mathsf{leave\text{-}st}(\beta_1) \neq \emptyset$. By Proposition 17 we know that $\beta_0 \equiv \beta_1$. Let β_0', β_1' be the conditions at positions, respectively, p and p' in b'^{h_0}. We know that $(\beta_0, \beta_0'), (\beta_1, \beta_1') \leq_{\mathsf{c}}^{\mathsf{h}_0,l} (t \sim t', P)$. We can extract from P a proof of:
$$\beta_0, \beta_0 \sim \beta_0', \beta_1'$$
in $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \overline{\mathsf{CCA}_2})$, 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 conditions. This yields a new proof Q in proof normal form, such that (19) 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 $\mathsf{cs\text{-}path}^{\mathsf{h}_0,l}(t, P) \times D_l^{\mathsf{h}_0}$.
- (21.iv): this is a consequence of Corollary 2.(i).
- (21.v): this is a consequence of Lemma 19.

Step 4. We conclude by showing that we can get rid of the $\{\overline{\mathsf{BFA}}(b, b')\}$ applications.

Using Corollary 2.(ii) and the proof Q constructed at **Step 3**, we know that for all $\gamma, \gamma' \leq_{\mathsf{l}}^{\mathsf{h}_0,l} (t, Q)$, $\gamma \equiv \gamma'$ (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:

$$D_l\left[\_ \diamond (\gamma)_{\gamma \leq_{\mathsf{l}}^{\mathsf{h}_0,l}(t,Q)}\right] \quad \text{and} \quad D_l\left[\_ \diamond (\gamma')_{\gamma \leq_{\mathsf{l}}^{\mathsf{h}_0,l}(t',Q)}\right]$$

into, respectively, γ_0 and γ_0' (where $\gamma_0 \leq_{\mathsf{l}}^{\mathsf{h}_0,l} (t,Q)$ and $\gamma_0' \leq_{\mathsf{l}}^{\mathsf{h}_0,l} (t',Q)$).

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**). □

H BOUNDING THE BASIC TERMS

H.1 α-Bounded Conditions

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 conditions 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 condition (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 conditions, 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 67. 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 22. 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 22 which use only (t,P)-α-bounded terms. Moreover, we require that no basic conditions appears twice on the same branch.

Definition 68. 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 22 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 condition from the root of t to some leave, $\vec{\rho}$ does not contain any duplicates. The same property must hold for t'.

We now prove the last proof cut elimination lemma.

Lemma (12). *$\vdash_{\alpha}^{npf}$ is complete for $\vdash^{npf}$.*

Proof. Let P be such that $P \vdash^{\mathsf{npf}} t \sim t'$, where P is obtained using Lemma 22. Therefore P satisfies the item (I) of Definition 68. Now, we are going to build from P a proof P' of $t \sim t'$ that satisfies the item (II) and (III) of Definition 68.

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.

Order $<_g$. Let $<_g$ be the transitive closure of the relation $\lessdot_g$ 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:

$$(\zeta, \mathsf{h}) \lessdot_g (\zeta', \mathsf{h}') \text{ iff } \begin{cases} \mathsf{h} = \mathsf{h}' \wedge \zeta, \zeta' \leq_{\mathsf{bt}}^{\mathsf{h},l} (t,P) \wedge\ \zeta \text{ is a guard of a dec. oracle call } d \in \mathsf{st}(\zeta') \\ \mathsf{h} = \mathsf{h}' \wedge \zeta \in \mathsf{cs\text{-}path}^{\mathsf{h},l}(t,P) \wedge \zeta' \leq_{\mathsf{bt}}^{\mathsf{h},l} (t,P) \wedge \zeta \in \mathsf{leave\text{-}st}(\zeta' \downarrow_R) \\ \mathsf{h} > \mathsf{h}' \wedge \zeta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t,P) \wedge \zeta' \in \mathsf{cs\text{-}path}^{\mathsf{h}',l}(t,P) \wedge \zeta' \in \mathsf{leave\text{-}st}(\zeta \downarrow_R) \end{cases}$$

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 $\zeta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t,P)$.

Case $h = \epsilon$. First we are going to describe what to do for $\mathsf{h} = \epsilon$. From Lemma 22.(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 22.(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 2.(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 $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \overline{\mathsf{CCA_2}})$. Therefore:

$$\forall b, b' \in \{\beta'_i \mid i \in I\}, b \equiv b' \equiv \beta' \tag{21}$$

  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:

$$\forall b, b' \in \{\beta_i \mid i \in I'\}, b \equiv b' \equiv \beta \tag{22}$$

  From (21) we get that $I \subseteq I'$ and conversely from (22) we get that $I' \subseteq I$. Therefore we have the equality $I = I'$.
- **(i), Part 2** For every $i \notin I$, using Lemma 17 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:

$$\begin{aligned} & D_l\left[\left((\beta)_{i\in I}, (\tilde{\beta}_i[\beta])_{i\notin I}\right) \diamond (\tilde{\gamma}_m[\beta])_{m\in M}\right] \\ =_R\ & \text{if } C_\beta[\vec{b}_\beta \diamond \vec{u}_\beta] \text{ then } D_l\left[\left((\mathsf{true})_{i\in I}, (\tilde{\beta}_i[\mathsf{true}])_{i\notin I}\right) \diamond (\tilde{\gamma}_m[\mathsf{true}])_{m\in M}\right] \\ & \qquad \text{else } D_l\left[\left((\mathsf{false})_{i\in I}, (\tilde{\beta}_i[\mathsf{false}])_{i\notin I}\right) \diamond (\tilde{\gamma}_m[\mathsf{false}])_{m\in M}\right] \end{aligned}$$

Since $\vec{u}_\beta = \text{leave-st}(\beta \downarrow_R)$, for every $u \in \vec{u}_\beta$, $i \in J$ and $m \in M$, we know that $u \notin \text{cond-st}(\tilde{\beta}_i[] \downarrow_R)$ and $u \notin \text{cond-st}(\tilde{\gamma}_m[] \downarrow_R)$. Let $\vec{\rho}$ be the conditions appearing on the path from the root of t to $D_l[\_]$. Using Lemma 22.(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 Lemma 20 to all u we know that:

$$\begin{aligned} & C\left[\vec{b}_{cs} \diamond \begin{pmatrix} \text{if } C_\beta\left[\vec{b}_\beta \diamond \vec{u}_\beta\right] \text{ then } D_l\left[\left((\mathsf{true})_{i\in I}, (\tilde{\beta}_i[\mathsf{true}])_{i\notin I}\right) \diamond (\tilde{\gamma}_i[\mathsf{true}])_m\right] \\ \qquad\qquad \text{else } D_l\left[\left((\mathsf{false})_{i\in I}, (\tilde{\beta}_i[\mathsf{false}])_{i\notin I}\right) \diamond (\tilde{\gamma}_i[\mathsf{false}])_m\right], \Delta \end{pmatrix}\right] \\ =_R\ & C\left[\vec{b}_{cs} \diamond \begin{pmatrix} \text{if } C_\beta\left[\vec{b}_\beta \diamond (\mathsf{true})_o\right] \text{ then } D_l\left[\left((\mathsf{true})_{i\in I}, (\tilde{\beta}_i[\mathsf{true}])_{i\notin I}\right) \diamond (\tilde{\gamma}_i[\mathsf{true}])_m\right] \\ \qquad\qquad \text{else } D_l\left[\left((\mathsf{false})_{i\in I}, (\tilde{\beta}_i[\mathsf{false}])_{i\notin I}\right) \diamond (\tilde{\gamma}_i[\mathsf{false}])_m\right], \Delta \end{pmatrix}\right] \\ =_R\ & C\left[\vec{b}_{cs} \diamond \left(D_l\left[\left((\mathsf{true})_{i\in I}, (\tilde{\beta}_i[\mathsf{true}])_{i\notin I}\right) \diamond (\tilde{\gamma}_i[\mathsf{true}])_m\right], \Delta\right)\right] \end{aligned} \tag{23}$$

- **(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:

$$\tilde{\beta}'_i[\beta'] \equiv \beta'_i \qquad \text{and} \qquad \text{leave-st}(\beta' \downarrow_R) \cap \text{cond-st}(\tilde{\beta}'_i[] \downarrow_R) = \emptyset$$

And, for every $m \in M$, there exists $\tilde{\gamma}'_m[]$ such that:

$$\tilde{\gamma}'_m[\beta'] \equiv \gamma'_m \qquad \text{and} \qquad \text{leave-st}(\beta' \downarrow_R) \cap \text{cond-st}(\tilde{\gamma}'_m[] \downarrow_R) = \emptyset$$

Then by the same reasoning we have:

$$\begin{aligned} t' \equiv\ & C\left[\vec{b}'_{cs} \diamond \left(D_l\left[\left(\beta'_i\right)_i \diamond \left(\gamma'_m\right)_{m\in M}\right], \Delta'\right)\right] \\ \equiv\ & C\left[\vec{b}'_{cs} \diamond \left(D_l\left[\left((\beta')_{i\in I}, (\tilde{\beta}'_i[\beta'])_{i\notin I}\right) \diamond \left(\tilde{\gamma}'_m[\beta']\right)_{m\in M}\right], \Delta'\right)\right] \\ =_R\ & C\left[\vec{b}'_{cs} \diamond \left(D_l\left[\left((\mathsf{true})_{i\in I}, (\tilde{\beta}'_i[\mathsf{true}])_{i\notin I}\right) \diamond \left(\tilde{\gamma}'_m[\mathsf{true}]\right)_{m\in M}\right], \Delta'\right)\right] \end{aligned} \tag{24}$$

Observe that corresponding sub-terms of (23) and (24) can be matched to corresponding sub-terms of t and t'. It is straightforward to build a proof of the equivalence of (23) and (24) using P, except for the CCA_2 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 11 instead of Lemma 2. The case (iii) is exactly like the case (i) when taking $I = \emptyset$.

Case $h \neq \epsilon$. In that case, thanks to Lemma 22.(a), we know that $\beta \not\leq_{\text{c}}^{\text{h},l} (t, P)$. We have three cases:

(a) either $\beta \leq_{\text{l}}^{\text{h},l} (t, P)$: using Lemma 22.(c), there exists h_0, b^h such that $\mathsf{h} \in \text{cs-pos}(\mathsf{h}_0)$, $b^h \in \text{cs-path}^{\mathsf{h}_0,l}(t, P)$ and $(b^h \downarrow_R) \in \text{leave-st}(\beta \downarrow_R)$. Since $\mathsf{h} \in \text{cs-pos}(\mathsf{h}_0)$ 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 $\beta \in \text{cs-path}^{\mathsf{h},l}(t,P)$: this case is done exactly like case (ii).

(c) either $\beta \not\leq_{\mathsf{l}}^{\mathsf{h},l} (t,P)$ and $\beta \notin \text{cs-path}^{\mathsf{h},l}(t,P)$: this case is done exactly like case (iii).

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 $\beta \leq_{\text{bt}}^{\mathsf{h},l} (t,P)$ or $\beta \in \text{cs-path}^{\mathsf{h},l}(t,P)$, *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 CCA_2 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 $\mathsf{EQL}(\_,\_)$.

Conclusion. This concludes the proof of the second bullet point of the definition $\vdash_\alpha^{\text{npf}}$. The third bullet point is much simpler. We want to show that for all $l \in \text{label}(\epsilon)$, for every path $\vec{\rho}$ of $\mathcal{S}_l^P$-normalized basic condition 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 $(\beta,\beta'_0) \leq_{\text{c}\sim\text{c}}^{\epsilon,l} (t,P)$ and $(\beta,\beta'_1) \leq_{\text{c}\sim\text{c}}^{\epsilon,l} (t,P)$, 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. □

H.2 Bounding the Number of Nested Basic Conditions

We use the previous lemma to bound the number of basic conditions appearing in a proof $P \vdash_\alpha^{\text{npf}} t \sim t'$. Looking at the definition of (t,P)-α-bounded terms, one may try to show that for every $\beta \in (\leq_{\text{bt}}^{\mathsf{h},l} (t,P) \cup \text{cs-path}^{\mathsf{h},l}(t,P))$, if β is (t,P)-α-bounded then there exists $u \in \text{leave-st}(\beta\downarrow_R)$ such that $u \in \text{st}(t\downarrow_R) \cup \text{st}(t'\downarrow_R)$. Since $\text{st}(t\downarrow_R) \cup \text{st}(t'\downarrow_R)$ 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 $P \vdash_\alpha^{\text{npf}} t \sim t'$.

Unfortunately, this is not always the case. Indeed, consider $(\beta,\beta') \leq_{\text{c}}^{\mathsf{h},l} (t \sim t', P)$ such that β' has a leaf term appearing in t', but β shares no leaf term with β' nor t:

$$\text{leave-st}(\beta\downarrow_R) \cap \text{leave-st}(\beta'\downarrow_R) = \emptyset \qquad \text{leave-st}(\beta\downarrow_R) \cap \text{st}(t\downarrow_R) = \emptyset$$

$$\text{leave-st}(\beta'\downarrow_R) \cap \text{st}(t'\downarrow_R) \neq \emptyset$$

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 $(\beta,\beta') \leq_{\text{c}}^{\mathsf{h},l} (t \sim t', P)$, we observe that $\beta \equiv B[\vec{w},(\alpha_i)_i,(\text{dec}_j)_j]$ and $\beta' \equiv B[\vec{w},(\alpha'_i)_i,(\text{dec}'_j)_j]$. Using the fact that $\text{leave-st}(\beta'\downarrow_R) \cap \text{st}(t'\downarrow_R)$ and that β is a $\mathcal{S}_l$-normalized

basic term, we know that every leaf $u \in \text{leave-st}(\beta\downarrow_R)$ is in $\text{st}(t'\downarrow_R)$, *modulo the content of the $\mathcal{S}_l$-encryption oracle calls*. This motivate the introduction of the notion of *leaf frame*.

Leaf frame. Let β be a $\mathcal{S}_l$-normalized basic term, and $u, v \in \text{leave-st}(\beta\downarrow_R)$ be leaf terms of β. Then u and v only differ by their encryptions. That is, if one replace all the zero decryptions $\mathbf{0}(\text{dec}(\_,\text{sk}))$ by $\text{dec}(\_,\text{sk})$, and all the leaves of encryptions $\{m\}^{\mathsf{n}}_{\mathsf{pk}}$ by $\{[]_\alpha\}^{\mathsf{n}}_{\mathsf{pk}}$ (where α is the unique term of $\mathcal{E}_l$ such that $\alpha \equiv \{\_\}^{\mathsf{n}}_{\mathsf{pk}}$) in u and in v then you get the same context. We formalize this below, and use it to generalize Proposition 17.

Definition 69. Let $P \vdash^{\text{npf}}_{\alpha} t \sim t'$ and l be a branch label in $\text{label}(P)$. We define the left *leaf frame* l-frame^P_l of $\beta \in (\leq^{\mathsf{h},l}_{\text{bt}}(t,P) \cup \text{cs-path}^{\mathsf{h},l}(t,P))$ inductively as follows:

$$\text{l-frame}^P_l(s) \equiv \begin{cases} \{[]_\alpha\}^{\mathsf{n}}_{\mathsf{pk}} & \text{if } \exists \alpha \equiv \{m\}^{\mathsf{n}}_{\mathsf{pk}} \in \mathcal{E}^P_l \wedge s \equiv \{\_\}^{\mathsf{n}}_{\mathsf{pk}} \\ \text{dec}(\text{l-frame}^P_l(s),\text{sk}) & \text{if } \text{sk} \in \mathcal{K}^P_l \wedge s \equiv \mathbf{0}(\text{dec}(s,\text{sk})) \\ \text{l-frame}^P_l(v) & \text{if } s \equiv \text{if } b \text{ then } u \text{ else } v \\ f((\text{l-frame}^P_l(u_i))_i) & \text{otherwise} \end{cases}$$

We also let $\underline{\text{l-frame}}^P_l(\beta)$ be $\text{l-frame}^P_l(\beta)$ 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* r-frame^P_l (and its underlined version $\underline{\text{r-frame}}^P_l$) of $\beta \in (\leq^{\mathsf{h},l}_{\text{bt}}(t',P) \cup \text{cs-path}^{\mathsf{h},l}(t',P))$, using $\mathcal{E}'^P_l$ instead of $\mathcal{E}^P_l$.

Remark 11. We have two remarks:

- We state some results only for l-frame. The corresponding results for r-frame are obtained by symmetry.
- The hole variables in $\underline{\text{l-frame}}^P_l(\beta)$ 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 $\underline{\text{l-frame}}^P_l(\beta)$ and $\underline{\text{l-frame}}^P_l(\beta')$ 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 $\underline{\text{l-frame}}^P_l$ to uniquely characterize normalized basic terms. ◇

Example 24. For all $\mathcal{S}^P_l$-decryption oracle call dec guarding $\text{dec}(s[(\alpha_i)_i,(\text{dec}_j)_j],\text{sk})$, if for all i, $\alpha_i \equiv \{\_\}^{\mathsf{n}_i}_{\mathsf{pk}_i}$ then:

$$\text{l-frame}^P_l(\text{dec}) \equiv \text{dec}\Big(s\Big[\big(\{[]_{\alpha_i}\}^{\mathsf{n}_i}_{\mathsf{pk}_i}\big)_i, \big(\text{l-frame}^P_l(\text{dec}_j)\big)_j\Big], \text{sk}\Big)$$

We also give an example of $\underline{\text{l-frame}}^P_l$. Assuming that $\alpha_0 \equiv \{A\}^{\mathsf{n}_0}_{\mathsf{pk}}$ and $\alpha_1 \equiv \{B\}^{\mathsf{n}_1}_{\mathsf{pk}}$ are encryptions in $\mathcal{E}^P_l$:

$$\underline{\text{l-frame}}^P_l(\langle \alpha_0, \langle \alpha_1, \alpha_0\rangle\rangle) \equiv \langle \{[]_{\alpha_0,00}\}^{\mathsf{n}_0}_{\mathsf{pk}}, \langle \{[]_{\alpha_1,100}\}^{\mathsf{n}_1}_{\mathsf{pk}}, \{[]_{\alpha_0,110}\}^{\mathsf{n}_0}_{\mathsf{pk}}\rangle\rangle \quad \diamond$$

Proposition 27. *Let $P \vdash^{npf}_{\alpha} t \sim t'$ and $l \in \text{label}(P)$. 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 $\text{l-frame}^P_l(b) \equiv \text{l-frame}^P_l(\gamma)$.*

Proof. This is by induction on the size of γ. □

Proposition 28. *Let $P \vdash^{npf}_{\alpha} t \sim t'$ and $l \in \text{label}(P)$. For every $\mathcal{S}_l$-normalized basic terms β, β', if $\text{l-frame}^P_l(\beta) \equiv \text{l-frame}^P_l(\beta')$ then $\beta \equiv \beta'$.*

Proof. The proof is exactly the same than for Proposition 17. □

Proposition 29. *Let $P \vdash_\alpha^{npf} t \sim t'$ and $l \in label(P)$. For all h, if $(b, b') \leq_{cs\sim cs}^{h,l} (t \sim t', P)$ then there exists h' and (γ, γ') $(\leq_{c\sim c}^{h',l} \cup \leq_{l\sim l}^{h',l})$ $(t \sim t', P)$ such that $b \in$ leave-st$(\gamma \downarrow_R)$ and $b' \in$ leave-st$(\gamma' \downarrow_R)$.*

Proof. Let h, x be such that $\mathsf{h} = h_\mathsf{x}$. Let $h_0 \in$ 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 $\mathsf{CS}_\square$ conditions of P do not use the $\overline{\mathsf{BFA}}$ rule, we know that Q lies in the fragment:

$$\mathfrak{F}(\mathsf{CS}_\square \cdot \mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \overline{\mathsf{CCA_2}})$$

Let $(\gamma, \gamma') \leq_{l\sim l}^{\epsilon,l} (b \sim b', Q)$. Using the property (c) of Lemma 22 (which holds thanks to $\vdash_\alpha^{\mathsf{npf}}$), we know that $b \in$ leave-st$(\gamma \downarrow_R)$ and $b \in$ leave-st$(\gamma' \downarrow_R)$. □

Proposition 30. *Let $P \vdash_\alpha^{npf} t \sim t'$ and $l \in label(P)$. For all h, if $(\beta, \beta')(\leq_{c\sim c}^{h,l} \cup \leq_{l\sim l}^{h,l} \cup cs\text{-}path_\sim^{h,l})(t \sim t', P)$ then* $\mathsf{l\text{-}frame}_l^P(\beta) \equiv \mathsf{r\text{-}frame}_l^P(\beta')$.

Proof. First we deal with the case (β, β') $(\leq_{\mathsf{c}\sim\mathsf{c}}^{\mathsf{h},l} \cup \leq_{\mathsf{l}\sim\mathsf{l}}^{\mathsf{h},l})$ $(t \sim t', P)$. 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 $\mathfrak{F}(\mathsf{FA_s}^* \cdot \mathsf{Dup}^* \cdot \overline{\mathsf{CCA_2}})$. The result follows from the definitions of $\mathsf{l\text{-}frame}_l^P$ and $\mathsf{r\text{-}frame}_l^P$.

Now we deal with the case $(\beta, \beta')(\mathsf{cs\text{-}path}_\sim^{\mathsf{h},l})(t \sim t', P)$. Using Proposition 29 we know that there exists h' and (γ, γ') $(\leq_{\mathsf{c}\sim\mathsf{c}}^{\mathsf{h}',l} \cup \leq_{\mathsf{l}\sim\mathsf{l}}^{\mathsf{h}',l})$ $(t \sim t', P)$ such that $\beta \in$ leave-st$(\gamma \downarrow_R)$ and $\beta' \in$ leave-st$(\gamma' \downarrow_R)$. Since β is if-free and in R-normal form, we obtain that $\mathsf{l\text{-}frame}_l^P(\beta) \equiv \mathsf{l\text{-}frame}_l^P(\gamma)$ by applying Proposition 27. Similarly $\mathsf{r\text{-}frame}_l^P(\beta') \equiv \mathsf{r\text{-}frame}_l^P(\gamma')$. Moreover, from the previous case, we get that $\mathsf{l\text{-}frame}_l^P(\gamma) \equiv \mathsf{r\text{-}frame}_l^P(\gamma')$. Hence $\mathsf{l\text{-}frame}_l^P(\beta) \equiv \mathsf{r\text{-}frame}_l^P(\beta')$. □

Proposition 31. *Let $P \vdash_\alpha^{npf} t \sim t'$ and $l \in label(P)$. For every $\mathcal{S}_l$-normalized basic terms β, β',* $\mathsf{l\text{-}frame}_l^P(\beta) \equiv \mathsf{l\text{-}frame}_l^P(\beta')$ *if and only if* $\underline{\mathsf{l\text{-}frame}}_l^P(\beta) \equiv \underline{\mathsf{l\text{-}frame}}_l^P(\beta')$.

Proof. This is obvious, since the hole variable annotations in $\underline{\mathsf{l\text{-}frame}}_l^P$ uniquely characterize both the position of the hole and the encryption appearing at this position. □

Proposition 32. *Let $P \vdash_\alpha^{npf} t \sim t'$ and $l \in label(P)$. For every $\mathcal{S}_l$-normalized basic terms β, β' and substitutions θ, θ', if* $\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \equiv \underline{\mathsf{l\text{-}frame}}_l^P(\beta')\theta'$ *then* $\underline{\mathsf{l\text{-}frame}}_l^P(\beta) \equiv \underline{\mathsf{l\text{-}frame}}_l^P(\beta')$.

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 $\mathsf{l\text{-}frame}_l^P$, and using the fact that $\mathsf{fresh}(\mathcal{R}_l^P; \vec{w})$, we have:

$$\mathsf{l\text{-}frame}_l^P(\beta) \equiv B[\vec{w}, (\{[]_{\alpha_i}\}_{\mathsf{pk}_i}^{\mathsf{n}_i})_i, \mathsf{dec}(\mathsf{l\text{-}frame}_l^P(s_j), \mathsf{sk}_j)]$$

Similarly:

$$\mathsf{l\text{-}frame}_l^P(\beta') \equiv B'[\vec{w}', (\{[]_{\alpha_i'}\}_{\mathsf{pk}_i'}^{\mathsf{n}_i'})_i, \mathsf{dec}(\mathsf{l\text{-}frame}_l^P(s_j'), \mathsf{sk}_j')]$$

We have three cases:

- Either $\beta \equiv \{m\}_{\mathsf{pk}}^{\mathsf{n}} \in \mathcal{E}_l^P$. Then $\underline{\mathsf{l\text{-}frame}}_l^P(\beta) \equiv \{[]_{\beta,0}\}_{\mathsf{pk}}^{\mathsf{n}}$. By definition of l-frame, and using the fact that $\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \equiv \underline{\mathsf{l\text{-}frame}}_l^P(\beta')\theta'$, 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 $\underline{\mathsf{l\text{-}frame}}_l^P(\beta) \equiv \mathsf{dec}(\underline{\mathsf{l\text{-}frame}}_l^P(s), \mathsf{sk})\mu$, 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 $\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \equiv \underline{\mathsf{l\text{-}frame}}_l^P(\beta')\theta'$ 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 $\underline{\mathsf{l\text{-}frame}}_l^P(s)\mu\theta \equiv \mathsf{l\text{-}frame}_l^P(s')\mu\theta$, and s, s' are $\mathcal{S}_l^P$-normalized basic terms. Hence by induction hypothesis $\underline{\mathsf{l\text{-}frame}}_l^P(s) \equiv \underline{\mathsf{l\text{-}frame}}_l^P(s')$, which concludes this case.
- Or we are not in one of the two cases above. Then, there exists $f \in \mathcal{F}_{\setminus \mathsf{if},\mathbf{0}}$ s.t. $\beta \equiv f(u_1, \ldots, u_n)$ and $\beta' \equiv f(u_1', \ldots, u_n')$, where $u_1, \ldots, u_n$ and $u_1', \ldots, u_n'$ are $\mathcal{S}_l^P$-normalized basic term. Hence $\underline{\mathsf{l\text{-}frame}}_l^P(\beta)$ and $\underline{\mathsf{l\text{-}frame}}_l^P(\beta')$ both starts with the function symbol f.
Moreover, if we let, for very $1 \leq i \leq n$, 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:
$$\underline{\mathsf{l\text{-}frame}}_l^P(\beta) \equiv f(\underline{\mathsf{l\text{-}frame}}_l^P(u_1)\mu_1, \ldots, \underline{\mathsf{l\text{-}frame}}_l^P(u_n)\mu_n)$$
$$\underline{\mathsf{l\text{-}frame}}_l^P(\beta') \equiv f(\underline{\mathsf{l\text{-}frame}}_l^P(u_1')\mu_1, \ldots, \underline{\mathsf{l\text{-}frame}}_l^P(u_n')\mu_n)$$
We apply θ to the equations above, and use the fact that $\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \equiv \underline{\mathsf{l\text{-}frame}}_l^P(\beta')\theta$:
$$\begin{aligned} f(\underline{\mathsf{l\text{-}frame}}_l^P(u_1)\mu_1\theta, \ldots, \underline{\mathsf{l\text{-}frame}}_l^P(u_n)\mu_n\theta) &\equiv \underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \\ &\equiv \underline{\mathsf{l\text{-}frame}}_l^P(\beta')\theta \\ &\equiv f(\underline{\mathsf{l\text{-}frame}}_l^P(u_1')\mu_1\theta, \ldots, \underline{\mathsf{l\text{-}frame}}_l^P(u_n')\mu_n\theta) \end{aligned}$$
Hence, for every $1 \leq i \leq n$, $\underline{\mathsf{l\text{-}frame}}_l^P(u_i)\mu_i\theta \equiv \underline{\mathsf{l\text{-}frame}}_l^P(u_i')\mu_i\theta$. By induction hypothesis, we deduce that $\underline{\mathsf{l\text{-}frame}}_l^P(u_i) \equiv \underline{\mathsf{l\text{-}frame}}_l^P(u_i')$. Therefore $\underline{\mathsf{l\text{-}frame}}_l^P(\beta) \equiv \underline{\mathsf{l\text{-}frame}}_l^P(\beta')$. □

Definition 70. We let $<_{\mathsf{st}}$ be the strict, well-founded, subterm ordering.

Nested Sequences of Basic Conditions. We want to bound the number of nested basic condition appearing in $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$. Using the contrapositive of Proposition 28, we know that when $\beta <_{\mathsf{st}} \beta'$ we have $\mathsf{l\text{-}frame}_l^P(\beta) \not\equiv \mathsf{l\text{-}frame}_l^P(\beta')$. Moreover, using Proposition 31 and Proposition 32, we know that $\mathsf{l\text{-}frame}_l^P(\beta) \not\equiv \mathsf{l\text{-}frame}_l^P(\beta')$ implies that $\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta \not\equiv \underline{\mathsf{l\text{-}frame}}_l^P(\beta')\theta'$ (for every substitutions θ, θ').

Therefore, for any sequence of nested $\mathcal{S}_l^P$-normalized basic conditions:
$$\beta_1 <_{\mathsf{st}} \cdots <_{\mathsf{st}} \beta_n$$
and for any substitutions $\theta_1, \ldots, \theta_n$, we know that $(\underline{\mathsf{l\text{-}frame}}_l^P(\beta_i)\theta_i)_{1\leq i\leq n}$ is a sequence of pair-wise distinct terms. Tu use this, we prove that there there exists a sequence of substitutions $\theta_1, \ldots, \theta_n$ such that:
$$\left\{\underline{\mathsf{l\text{-}frame}}_l^P(\beta_1)\theta_1, \ldots, \underline{\mathsf{l\text{-}frame}}_l^P(\beta_n)\theta_n\right\} \subseteq \mathcal{B}(t, t')$$
where $\mathcal{B}(t, t')$ is a set of bounded size w.r.t. $|t| + |t'|$. Since the $(\mathsf{l\text{-}frame}_l^P(\beta_i)\theta_i)_{1\leq i\leq n}$ are pair-wise distinct, using a pigeon-hole argument we get that $n \leq |\mathcal{B}(t, t')|$.

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 condition n using a pigeon-hole argument.

Definition 71. 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 $\mathbf{0}(\mathrm{dec}(w,\mathsf{sk}))$ by $\mathrm{dec}(w,\mathsf{sk})$ (where $\mathsf{sk}\in\mathcal{K}$), non-deterministically stopping at some encryptions. Formally:

$$\zeta_{\mathcal{K}}(u)=\begin{cases}\{\mathrm{dec}(v,\mathsf{sk})\mid w\in v\in\zeta_{\mathcal{K}}(w)\} & \text{if } u\equiv\mathbf{0}(\mathrm{dec}(w,\mathsf{sk})) \text{ and } \mathsf{sk}\in\mathcal{K}\\ \{u\}\cup\{\{v\}^{\mathsf{n}_r}_{\mathsf{pk(n)}}\mid v\in\zeta_{\mathcal{K}}(m)\} & \text{if } u\equiv\{m\}^{\mathsf{n}_r}_{\mathsf{pk(n)}} \text{ and } \mathsf{sk(n)}\in\mathcal{K}\\ \{f(v_1,\dots,v_n)\mid\forall i,v_i\in\zeta_{\mathcal{K}}(u_i)\} & \text{otherwise, where } u\equiv f(u_1,\dots,u_n)\end{cases}$$

Moreover, given a set of ground terms $\mathcal{S}$, we let $\mathrm{guards}_{\mathcal{K}}(\mathcal{S})$ be an over-approximation of the set of guards of terms in $\mathcal{S}$:

$$\mathrm{guards}_{\mathcal{K}}(\mathcal{S})=\left\{\mathrm{eq}(s,\alpha)\mid\mathrm{dec}(s,\mathsf{sk(n)})\in\mathcal{S}\wedge\alpha\equiv\{\_\}^{-}_{\mathsf{pk(n)}}\in\mathrm{st}(s)\wedge\mathsf{sk(n)}\in\mathcal{K}\right\}$$

Definition 72. Let $\mathcal{S}_{\mathsf{k}}(t)$ be the set of private keys appearing in $t\downarrow_R$, i.e. $\mathcal{S}_{\mathsf{k}}(t)=\{\mathsf{sk(n)}\mid\mathsf{sk(n)}\in\mathrm{st}(t\downarrow_R)\}$. For every term t, we let $\mathcal{B}(t)$ be the set:

$$\mathcal{B}(t)=\bigcup_{\mathcal{K}\subseteq\mathcal{S}_{\mathsf{k}}(t)}\ \bigcup_{\substack{u\in\mathrm{st}(\text{leave-st}(t\downarrow_R))\\ \vee u\in\mathrm{st}(\text{cond-st}(t\downarrow_R))}}\zeta_{\mathcal{K}}(u)\cup\mathrm{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u))$$

Moreover, we let $\mathcal{B}(t,t')=\mathcal{B}(t)\cup\mathcal{B}(t')$.

Proposition 33. *Let $P\vdash^{npf}_{\alpha}t\sim t'$ and $l\in\mathit{label}(P)$. Let β be a $\mathcal{S}^P_l$-normalized basic condition. Then, for every $u\in\text{leave-st}(\beta\downarrow_R)$, there exists θ such that $\underline{\text{l-frame}}^P_l(\beta)\theta\in\zeta_{\mathcal{K}}(u)$.*

Proof. We show this by induction on $|\beta|$.

- If β is an encryption $\{m\}^{\mathsf{n}}_{\mathsf{pk}}\in\mathcal{E}^P_l$, then $\underline{\text{l-frame}}^P_l(\beta)\equiv\{[]_{\beta,0}\}^{\mathsf{n}}_{\mathsf{pk}}$ and:

$$\text{leave-st}(\beta\downarrow_R)=\left\{\{v\}^{\mathsf{n}}_{\mathsf{pk}}\mid v\in\text{leave-st}(m\downarrow_R)\right\}$$

  Let $u\in\text{leave-st}(\beta\downarrow_R)$, there exists $u_m\in\text{leave-st}(m\downarrow_R)$ such that $u\equiv\{u_m\}^{\mathsf{n}}_{\mathsf{pk}}$. Let θ be the substitution mapping $[]_{\beta,0}$ to u_m. Then:

$$\underline{\text{l-frame}}^P_l(\beta)\theta\equiv\{u_m\}^{\mathsf{n}}_{\mathsf{pk}}\equiv u\in\zeta_{\mathcal{K}^P_l}(u)$$

- If β is a decryption oracle call in $\mathcal{D}^P_l$ for $\mathrm{dec}(s,\mathsf{sk})$, the:

$$\text{leave-st}(\beta\downarrow_R)\subseteq\{\mathrm{dec}(u_s,\mathsf{sk})\mid u_s\in\text{leave-st}(s\downarrow_R)\}\cup\{\mathbf{0}(\mathrm{dec}(u_s,\mathsf{sk}))\mid u_s\in\text{leave-st}(s\downarrow_R)\}$$

  Let $u\in\text{leave-st}(\beta\downarrow_R)$, there exists $u_s\in\text{leave-st}(s\downarrow_R)$ such that $u\equiv\mathrm{dec}(u_s,\mathsf{sk})$ or $u\equiv\mathbf{0}(\mathrm{dec}(u_s,\mathsf{sk}))$. Since s is a $\mathcal{S}^P_l$-normalized basic term, by induction hypothesis we have θ such that $\underline{\text{l-frame}}^P_l(s)\theta\in\zeta_{\mathcal{K}^P_l}(u_s)$. Moreover:

$$\underline{\text{l-frame}}^P_l(\beta)\equiv\mathrm{dec}(\underline{\text{l-frame}}^P_l(s)\mu,\mathsf{sk})$$

  where μ is a renaming of hole variables. Let $\theta'=\mu^{-1}\theta$, then:

$$\underline{\text{l-frame}}^P_l(\beta)\theta'\equiv\mathrm{dec}(\underline{\text{l-frame}}^P_l(s)\mu\mu^{-1}\theta,\mathsf{sk})\equiv\mathrm{dec}(\underline{\text{l-frame}}^P_l(s)\theta,\mathsf{sk})\in\zeta_{\mathcal{K}^P_l}(u)$$

- Otherwise, $\beta\equiv f(\beta_1,\dots,\beta_n)$ where, for every $1\le i\le n$, β_i is a $\mathcal{S}^P_l$-normalized basic term. Then, using the fact that β is a $\mathcal{S}^P_l$-normalized basic term, we check that:

$$\text{leave-st}(\beta\downarrow_R)\subseteq\{f(v_1,\dots,v_n)\mid\forall i,v_i\in\text{leave-st}(\beta_i\downarrow_R)\}$$

Let $u \in \text{leave-st}(\beta\downarrow_R)$, there exists $v_1,\dots,v_n$ such that for every $1 \le i \le n$ $v_i \in \text{leave-st}(\beta_i\downarrow_R)$ and $u \equiv f(v_1,\dots,v_n)$. By induction hypothesis, there exists $\theta_1,\dots,\theta_n$ such that for every $1 \le i \le n$:

$$\underline{\text{l-frame}}_l^P(\beta_i)\theta_i \in \zeta_{\mathcal{K}_l^P}(v_i)$$

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:

$$\underline{\text{l-frame}}_l^P(\beta) \equiv f(\underline{\text{l-frame}}_l^P(\beta_1)\mu_1,\dots,\underline{\text{l-frame}}_l^P(\beta_n)\mu_n)$$

Observe that the substitutions $(\mu_i\theta_i)_{1\le i\le n}$ have disjoint domains. Let $\theta = \mu_1\theta_1\dots\mu_n\theta_n$. Then:

$$\underline{\text{l-frame}}_l^P(\beta)\theta \equiv f(\underline{\text{l-frame}}_l^P(\beta_1)\mu_1\theta_1,\dots,\underline{\text{l-frame}}_l^P(\beta_n)\mu_n\theta_n)$$

We know that f cannot be the function symbol $\mathbf{0}(\_)$ (since FA cannot be applied on $\mathbf{0}(\_)$). It follows that:

$$f(\underline{\text{l-frame}}_l^P(\beta_1)\mu_1\theta_1,\dots,\underline{\text{l-frame}}_l^P(\beta_n)\mu_n\theta_n) \in \zeta_{\mathcal{K}_l^P}(u) \qquad \square$$

We lift the previous result to α-bounded conditions.

Lemma 23. *Let $P \vdash_\alpha^{npf} t \sim t'$, l a branch label in label(P), h a proof index and $\beta \in (\le_{bt}^{h,l}(t,P) \cup$ cs-path$^{h,l}(t,P))$. If β is (t,P)-α-bounded then there exists a substitution θ s.t. $\underline{\text{l-frame}}_l^P(\beta)\theta \in \mathcal{B}(t,t')$.*

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 $\text{leave-st}(\beta\downarrow_R) \cap \text{st}(t\downarrow_R) \neq \emptyset$. Let $u \in \text{leave-st}(\beta\downarrow_R) \cap \text{st}(t\downarrow_R)$, we have u in R-normal form and if-free, therefore $u \in \text{st}(\text{leave-st}(t\downarrow_R) \cup \text{cond-st}(t\downarrow_R))$. Moreover, by Proposition 33, there exists θ such that $\underline{\text{l-frame}}_l^P(\beta)\theta \in \zeta_{\mathcal{K}_l^P}(u)$. Hence $\underline{\text{l-frame}}_l^P(\beta)\theta \in \mathcal{B}(t,t')$.
- **Base case:** Assume $\mathsf{h} = \epsilon$ and there exists β' such that:

  $$(\beta,\beta')\,(\le_{\mathsf{l}\sim\mathsf{l}}^{\epsilon,l} \cup \le_{\mathsf{c}\sim\mathsf{c}}^{\epsilon,l} \cup \le_{\mathsf{cs}\sim\mathsf{cs}}^{\epsilon})\,(t \sim t', P) \quad \text{and} \quad \text{leave-st}(\beta'\downarrow_R) \cap \text{st}(t'\downarrow_R) \neq \emptyset$$

  By Proposition 30 we know that $\text{l-frame}_l^P(\beta) \equiv \text{r-frame}_l^P(\beta')$. By Proposition 31, we deduce that $\underline{\text{l-frame}}_l^P(\beta) \equiv \underline{\text{r-frame}}_l^P(\beta')$. From the previous case we know that there exists θ such that $\underline{\text{r-frame}}_l^P(\beta')\theta \in \mathcal{B}(t')$. Therefore $\underline{\text{l-frame}}_l^P(\beta)\theta \in \mathcal{B}(t')$.
- **Inductive case, same label:** Assume $\beta \in \text{cs-path}^{\mathsf{h},l}(t,P)$ and that there exists $\varepsilon \le_{\mathsf{bt}}^{\mathsf{h},l}(t,P)$ such that ε is (t,P)-α-bounded and $\beta \in \text{leave-st}(\varepsilon\downarrow_R)$. By induction hypothesis we have θ such that $\underline{\text{l-frame}}_l^P(\varepsilon)\theta \in \mathcal{B}(t,t')$. 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 27, we have $\text{l-frame}_l^P(\beta) \equiv \text{l-frame}_l^P(\varepsilon)$. Hence, using Proposition 31, $\underline{\text{l-frame}}_l^P(\beta)\theta \in \mathcal{B}(t,t')$.
- **Inductive case, different labels:** Similar to the previous case.
- **Inductive case, guard:** If there exists $\varepsilon \le_{\mathsf{bt}}^{\mathsf{h},l}(t,P)$ such that:
  – $\varepsilon \equiv B[\vec{w},(\alpha_i)_i,(\text{dec}_j)_j]$ is (t,P)-α-bounded.
  – β is a guard of a $\mathcal{S}_l^P$-decryption oracle call $d \in (\text{dec}_j)_j$.
  By induction hypothesis there exists θ such that $\underline{\text{l-frame}}_l^P(\varepsilon)\theta \in \mathcal{B}(t,t')$. 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:

  $$\text{l-frame}_l^P(\varepsilon) \equiv B\left[\vec{w},\left(\{[]_{\alpha_i}\}_{\mathsf{pk}_i}^{\mathsf{n}_i}\right)_i,\left(\text{l-frame}_l^P(\text{dec}_j)\right)_j\right]$$

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:

$$\mathsf{l\text{-}frame}_l^P(d) \equiv \mathsf{dec}(\mathsf{l\text{-}frame}_l^P(s),\mathsf{sk}) \qquad \mathsf{l\text{-}frame}_l^P(\beta) \equiv \mathsf{eq}(\mathsf{l\text{-}frame}_l^P(s),\{[]_\alpha\}^{\mathsf{n}}_{\mathsf{pk}})$$

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:

$$\underline{\mathsf{l\text{-}frame}}_l^P(\beta)\theta' \in \mathsf{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u)) \subseteq \mathcal{B}(t,t') \qquad \square$$

We now bound the size of $\mathcal{B}(t)$.

Proposition 34. *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 $\mathbf{0}$s). Hence:

$$\begin{aligned}\left|\bigcup_{\mathcal{K}\subseteq\mathcal{S}_{\mathsf{k}}(t)} \zeta_{\mathcal{K}}(u) \cup \mathsf{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u))\right| &\leq \left|\bigcup_{\mathcal{K}\subseteq\mathcal{S}_{\mathsf{k}}(t)} \zeta_{\mathcal{K}}(u)\right| + \left|\bigcup_{\mathcal{K}\subseteq\mathcal{S}_{\mathsf{k}}(t)} \mathsf{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u))\right| \\ &\leq |\zeta_{\mathcal{K}}(u)| + (|t\downarrow_R| - 1).|\zeta_{\mathcal{K}}(u)| \leq |t\downarrow_R|.2^{|t\downarrow_R|}\end{aligned}$$

By consequence:

$$|\mathcal{B}(t)| \leq |t\downarrow_R|.|t\downarrow_R|.2^{|t\downarrow_R|} \leq |t\downarrow_R|^2.2^{|t\downarrow_R|} \qquad \square$$

Finally, we apply a pigeon-hole argument to bound the number of nested basic terms.

Lemma 24. *Let $P \vdash_\alpha^{npf} t \sim t'$. Let l be a branch label in label(P), h a proof index. Let $(\beta_i)_{i\leq n}$ such that for all i, $\beta_i \leq_{bt}^{h,l} (t,P)$. If $\beta_1 <_{st} \cdots <_{st} \beta_n$ then $n \leq |\mathcal{B}(t,t')|$.*

Proof. For every $i \neq j$, we know, using Proposition 28, that $\mathsf{l\text{-}frame}_l^P(\beta_i) \not\equiv \mathsf{l\text{-}frame}_l^P(\beta_j)$. By Proposition 31, we deduce that $\underline{\mathsf{l\text{-}frame}}_l^P(\beta_i) \not\equiv \underline{\mathsf{l\text{-}frame}}_l^P(\beta_j)$. Since $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$, we know that for every i, β_i is (t,P)-α-bounded. Using Lemma 23, we deduce that for every i, there exists a substitution θ_i such that:

$$\underline{\mathsf{l\text{-}frame}}_l^P(\beta_i)\theta_i \in \mathcal{B}(t,t')$$

Using the contrapositive of Proposition 32, we have that for every $i \neq j$:

$$\underline{\mathsf{l\text{-}frame}}_l^P(\beta_i)\theta_i \not\equiv \underline{\mathsf{l\text{-}frame}}_l^P(\beta_j)\theta_j$$

Therefore, by a pigeon-hole argument, $n \leq |\mathcal{B}(t,t')|$. □

H.3 Candidate Sequences

Let $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$. 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 73. 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 12. For every $\beta \leq_{\mathsf{bt}}^{\mathsf{h},l} (t,P)$, the context $\mathsf{l\text{-}frame}_l^P(\beta)$ 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 $\vdash_\alpha^{\mathsf{npf}} t \sim t'$, while $\mathcal{A}_n$ is the set of left encryptions oracle calls built using basic terms in $\mathcal{U}_{n-1}$.

Definition 74. 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 $\{s_i\}^{\mathsf{n}_r}_{\_} \equiv \alpha_i \in \mathcal{A}_n$.
  - 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 35. *Let $P \vdash_\alpha^{npf} t \sim t'$. For $l \in$ label(P), there exists a valid candidate sequence $(\mathcal{U}_n, \mathcal{A}_n)_{n\in\mathbb{N}}$ for t, t' such that:*

$$\bigcup_h \leq_{bt}^{h,l}(t,P) \subseteq \bigcup_{n<|\mathcal{B}(t,t')|} \mathcal{U}_n \qquad \text{and} \qquad \bigcup_h cs\text{-}path^{h,l}(t,P) \subseteq \bigcup_{n<|\mathcal{B}(t,t')|} \text{leave-st}(\mathcal{U}_n \downarrow_R)$$

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 $\mathcal{E}_l^P$, i.e.:

$$\mathcal{S} = \bigcup_{\mathsf{h}} \leq_{\mathsf{bt}}^{\mathsf{h},l}(t,P) \subseteq \bigcup_{n<+\infty} \mathcal{U}_n \qquad \text{and} \qquad \bigcup_{n\in\mathbb{N}} \mathcal{A}_n \subseteq \mathcal{E}_l^P \tag{25}$$

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 $<_{\text{st}}$ in $\mathcal{S}\backslash\mathcal{S}'$. Then we add β to $\mathcal{S}'$ and update $(\mathcal{A}_n)_{n\in\mathbb{N}}$ as follows:

Case 1. If β is minimal for $<_{\text{st}}$ in $\mathcal{S}$, we have β of the form $B[\vec{w}, (\alpha_i)_{i\in I}, (\text{dec}_j)_{j\in J}]$. By minimality of β, we have $I = \emptyset$ and for all $j \in J$, dec_j has no encryptions in $\mathcal{E}_l^P$, and by consequence no guards. It follows that β is if-free and in R-normal form, hence $\text{l-frame}_l^P(\beta) \equiv \beta$. By consequence, using Lemma 23, we get that $\beta \in \mathcal{B}(t,t') = \mathcal{U}_0$ (since $\mathcal{U}_0$ does not depends on the sets $(\mathcal{A}_n)_{n\in\mathbb{N}}$).

Case 2. Let β such that for all $\beta' <_{\text{st}} \beta$, $\beta' \in \mathcal{S}'$. Since $\mathcal{S}' \subseteq \cup_{n\in\mathbb{N}}\mathcal{U}_n$, and since $\{\beta' \mid \beta' <_{\text{st}} \beta\}$ is finite, there exists n_m such that:

$$\{\beta' \mid \beta' <_{\text{st}} \beta\} \cap \left(\leq_{\text{bt}}^{\mathsf{h},l}(t,P) \cup \text{cs-path}^{\mathsf{h},l}(t,P)\right) \subseteq \bigcup_{0\leq n\leq n_m} \mathcal{U}_n$$

From Lemma 23 we have a substitution θ such that:

$$\underline{\text{l-frame}}_l^P(\beta)\theta \in \mathcal{B}(t,t')$$

We then just need to show that we can obtain β from $\underline{\text{l-frame}}_l^P(\beta)$ using the procedure defining $\mathcal{U}_{n_m+1}$:

- For all encryption $\alpha \equiv \{m\}_{\text{pk}}^{\mathsf{n}} \in \text{st}(\beta) \cap \mathcal{E}_l^P$, we know that $m \equiv C[\vec{b} \diamond \vec{u}]$ where $\vec{b}, \vec{u} <_{\text{st}} \beta$. 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 $\cup_{n\in\mathbb{N}}\mathcal{A}_n \subseteq \mathcal{E}_l^P$, and using the side-condition of the CCA_2 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 $\vdash_\alpha^{\mathsf{npf}}$).

Then we replace in $\underline{\text{l-frame}}_l^P(\beta)$ the holes $[]_\alpha, \_$ by $\{C[\vec{b} \diamond \vec{u}]\}_{\mathsf{pk}}^{\mathsf{n}}$. 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 $\mathsf{st}(\text{l-frame}_l^P(\beta))$ by the corresponding $\mathcal{S}_l^P$-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 $\cup_{0 \leq n \leq n_m} \mathcal{U}_n$.

Conclusion. We show that when $\mathcal{S} = \mathcal{S}'$ we have:

$$\mathcal{S} \cap \bigcup_{n<+\infty} \mathcal{U}_n \;=\; \mathcal{S} \cap \bigcup_{n<|\mathcal{B}(t,t')|} \mathcal{U}_n \tag{26}$$

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 $\alpha \equiv \{C[\vec{b} \diamond \vec{u}]\}_{\mathsf{pk}}^{\mathsf{n}}$, 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 $\mathcal{S}_l^P$-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 $n \in \{0, \ldots, |\mathcal{B}(t,t')|\}$ 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 24. Absurd.

To finish, it remains to show that:

$$\bigcup_{\mathsf{h}} \mathsf{cs\text{-}path}^{\mathsf{h},l}(t,P) \subseteq \bigcup_{n<|\mathcal{B}(t,t')|} \mathsf{leave\text{-}st}\,(\mathcal{U}_n \downarrow_R)$$

Let b in $\bigcup_{\mathsf{h}} \mathsf{cs\text{-}path}^{\mathsf{h},l}(t,P)$. Using Proposition 29 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 $b \in \bigcup_{n<|\mathcal{B}(t,t')|} \mathsf{leave\text{-}st}\,(\mathcal{U}_n \downarrow_R)$. □

Proposition 36. *For all terms u, let C_u be the set of α-contexts:*

$$C_u = \{C \mid \exists \theta.\, C\theta \equiv u \wedge \textit{every hole appears at most once}\}$$

and C_u^α be C_u quotiented by the α-renaming of holes relation. Then $|C_u^\alpha| \leq 2^{|u|}$.

Proof. The set of contexts C_u^α 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 C_u^α than subsets of $\mathsf{pos}(u)$, i.e. $2^{|\mathsf{pos}(u)|} = 2^{|u|}$. □

Proposition 37. *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:

$$\bigcup_{u \in \mathcal{B}(t,t')} C_u^\alpha$$

which, using Proposition 34 and Proposition 36, we can bound by:

$$\left| \bigcup_{u \in \mathcal{B}(t,t')} C_u^\alpha \right| \leq \sum_{u \in \mathcal{B}(t,t')} \left|C_u^\alpha\right| \leq \sum_{u \in \mathcal{B}(t,t')} 2^{2.N} \leq N^2.2^N.2^{2.N} = N^2.2^{3.N} \quad \square$$

Proposition 38. *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 $\{C[\vec{b} \diamond \vec{u}]\}^{\mathsf{n}}_{\mathsf{pk}}$, 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 conditions*, 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 $2^{|\mathcal{U}_n|+2} - 1$ condition 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 37, 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 34. Hence, assuming $U_n \geq N$ (which will be the case):

$$\begin{aligned}\left|C[(\alpha_i)_{i\in I}, (\mathsf{dec}_j)_{j\in J}]\right| &\leq |C| + |I|.A_n + |J|.(N + N.U_n + (N + 1).(|C| + |I|.A_n + 1)) \\ &\leq 2N + 2N.A_n + 2N.(N + N.U_n + (N + 1).(2N + 2N.A_n + 1))\end{aligned}$$

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 3. *Let $P \vdash_\alpha^{npf} t \sim t'$ and $N = |\mathcal{B}(t,t')|$. For $l \in label(P)$ and for all proof index h:*

$$\forall u \in \left(\leq_{bt}^{h,l} (t,P) \cup cs\text{-}path^{h,l}(t,P)\right),\ |u| \leq 2^{Q(N).2^{4.N}}$$

Proof. Direct consequence of Proposition 35 and Proposition 38. □

To conclude, we only need to bound the number of nested $CS_\square$ conditions.

Proposition 39. *Let $P \vdash_\alpha^{npf} t \sim t'$ and $(h_i)_{1\leq i\leq n}$ be a sequence of indices of P such that for every $1 \leq i < n$, $h_{i+1} \in cs\text{-}pos_P(h_i)$ and $h_1 = \epsilon$. Then $n \leq |\mathcal{B}(t,t')| + 1$. Moreover $|label(P)| \leq 2^{|\mathcal{B}(t,t')|}$.*

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 $\mathcal{S}_l^P$-normalized basic terms $\beta_1 <_{\mathrm{st}} \cdots <_{\mathrm{st}} \beta_m$ from $(\mathsf{h}_i)_{1\leq i\leq n}$ of length $m \geq n$. We then concludes using Lemma 24.

If $\mathsf{h}_n \neq \epsilon$, then h_n is of the form $h^n_{\mathsf{x}_n}$. We know that $\mathrm{extract}_{\mathsf{x}_n}(h^n, P)$ is a proof of $b^n \sim b'^n$ in $\mathcal{A}_{\mathrm{CS}_\square}$. 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 $1 \leq j \leq k_n$, $\beta_{n,j} \leq_{\mathsf{bt}}^{\mathsf{h}_{n-1},l} (t,P)$.
- $b^n \downarrow_R \in \mathsf{leave\text{-}st}(\beta_{n,1}\downarrow_R)$.
- $\beta_{n,k_n} \leq_{\mathsf{l}}^{\mathsf{h}_{n-1},l} (t,P)$.
- for all $1 \leq j < k_n$, $\beta_{n,j}$ is a guard of a decryption in $\beta_{n,j+1}$, and therefore $\beta_{n,j} <_{\mathrm{st}} \beta_{n,j+1}$.

If $\mathsf{h}_{n-1} \neq \epsilon$, then since $\beta_{n,k_n} \leq_{\mathsf{l}}^{\mathsf{h}_{n-1},l} (t,P)$ is (t,P)-α-bounded, and since for any $\beta \leq_{\mathsf{bt}}^{\mathsf{h}_{n-1},l} (t,P)$, $\beta_{n,j}$ 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 17, that $\beta_{i,k_i} \equiv \beta_{i-1,1}$. Therefore we have:

$$\beta_{n,1} <_{\mathrm{st}} \cdots <_{\mathrm{st}} \beta_{n,k_n} \equiv \beta_{n-1,1} <_{\mathrm{st}} \cdots <_{\mathrm{st}} \beta_{n-1,k_{n-1}} \cdots <_{\mathrm{st}} \beta_{3,k_3} \equiv \beta_{2,1} <_{\mathrm{st}} \cdots <_{\mathrm{st}} \beta_{2,k_2}$$

Moreover, for all i we have $k_i \geq 1$, therefore we built an increasing sequence of $\mathcal{S}_l^P$-normalized basic terms of length at least $n-1$. It follows, using Lemma 24, that $n - 1 \leq |\mathcal{B}(t,t')|$.

To upper-bound $|\mathsf{label}(P)|$, we only need to observe that we cannot have two $CS_\square$ applications on the same condition in a given branch. Consider the binary tree associated to the $CS_\square$ applications in P, labelled by the corresponding $CS_\square$ conditions (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 in 3-NExpTime:*
Input: *A ground formula $\vec{u} \sim \vec{v}$.*
Question: *Is $Ax \wedge \vec{u} \nsim \vec{v}$ unsatisfiable?*

Proof. Let $\vec{u} = u_1, \ldots, u_n$, $\vec{v} = v_1, \ldots, v_n$ and:

$$t \equiv \langle u_1, \langle \ldots, \langle u_{n-1}, u_n\rangle\rangle\rangle \qquad t' \equiv \langle v_1, \langle \ldots, \langle v_{n-1}, v_n\rangle\rangle\rangle$$

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:

$$\forall n > i \geq 1, \rho_i \equiv \pi_1(\pi_2^{i-1}([])) \qquad \rho_n[] \equiv \pi_2^{n-1}([])$$

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 39, we have bounded the number of branches of the proof tree (by $2^{N^2.2^N}$), and the number of nested $\mathsf{CS}_\Box$ conditions. For every branch, we non-deterministically guesses a set of α-bounded basic terms that can appear in a proof P of $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$ using the valid candidate sequence algorithm (in polynomial time in $O(N.2^{3.N}.2^{Q(N).2^{4.N}})$, using Proposition 37 and Proposition 38). 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 CCA_2 instances we use the polynomial-time algorithm describe in Proposition 11. 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. □

H.4 The Pure Fragment

Proof of Lemma 13. Consider a proof $P \vdash_\alpha^{\mathsf{npf}} t \sim t'$ that only uses Refl_α. For every h, l, for every $\gamma \leq_{\mathsf{bt}}^{\mathsf{h},l} (t, P)$, γ is a normalized basic term. Since we are in the pure fragment, γ contains no encryptions and decryptions. Consequently, γ is if-free and in R-normal form.

Proof cut. First, we simplify the proof P. We look at the following case in Definition 67. Let h, l, and β $(\leq_{\mathsf{bt}}^{\mathsf{h},l} \cup \mathsf{cs\text{-}path}^{\mathsf{h},l})$ (t, P) such that:

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 $b \in \mathsf{leave\text{-}st}(\beta \downarrow_R)$.

Since β is if-free and in R-normal form, $\mathsf{leave\text{-}st}(\beta \downarrow_R) = \{\beta\}$, hence $\beta \equiv b$. As this holds for both the then and else branch of the case-study on b, we have a proof-cut elimination: simply use the proof used for β. Therefore, we can consider a proof of $t \sim t'$ where for every (b, b') $\mathsf{cs\text{-}path}_{\sim}^{\mathsf{h},l}$ $(t \sim t', P)$, the proof of $b \sim b'$ extracted from P does not use $\mathsf{CS}_\Box$. Hence P can be taken in the fragment with no boxed CS conditions:

$$\mathfrak{F}(R \cdot \mathsf{CS}^* \cdot \{\mathsf{BFA}(b, b')\}^* \cdot {\mathsf{FA}_\mathsf{s}}^* \cdot \mathsf{Dup}^* \cdot \mathsf{Refl}_\alpha) \tag{27}$$

Remark that if $u \sim v$ is provable in $\mathfrak{F}(\{\mathsf{BFA}(b, b')\}^* \cdot {\mathsf{FA}_\mathsf{s}}^* \cdot \mathsf{Dup}^* \cdot \mathsf{Refl}_\alpha)$, then u and v are alpha-equal.

Conclusion. We prove by induction on the α-bounded inductive predicate that for every h, l, and (β, β') $(\leq_{\mathsf{l\sim l}}^{\mathsf{h},l} \cup \leq_{\mathsf{c\sim c}}^{\mathsf{h},l} \cup \mathsf{cs\text{-}path}_{\sim}^{\mathsf{h},l})$ (t, P), β or β' is a member of $\mathsf{st}(t \downarrow_R) \cup \mathsf{st}(t' \downarrow_R)$, and β contains only names in $\mathsf{st}(t) \cup \mathsf{st}(t')$.

We look at all possible cases in Definition 67. Let h, l, and β $(\leq_{\mathsf{bt}}^{\mathsf{h},l} \cup \mathsf{cs\text{-}path}^{\mathsf{h},l})$ (t, P).

- **Base case:** Since β is if-free and in R-normal form, $\mathsf{leave\text{-}st}(\beta \downarrow_R) = \{\beta\}$, hence $\beta \in \mathsf{st}(t \downarrow_R)$.

- **Base case:** There exists β' such that:
$$(\beta, \beta')\,(\leq^{\epsilon,l}_{\mathsf{l}\sim\mathsf{l}} \cup \leq^{\epsilon,l}_{\mathsf{c}\sim\mathsf{c}} \cup \mathsf{cs\text{-}path}^{\epsilon,l}_{\sim})\,(t \sim t', P)$$
and $\mathsf{leave\text{-}st}(\beta' \downarrow_R) \cap \mathsf{st}(t' \downarrow_R) \neq \emptyset$. We deduce that $\beta' \in \mathsf{st}(t' \downarrow_R)$.
- **Inductive case, same label:** if $\beta \in \mathsf{cs\text{-}path}^{\mathsf{h},l}(t, P)$ and there exists $\varepsilon \leq^{\mathsf{h},l}_{\mathsf{bt}} (t, P)$ such that ε is (t, P)-α-bounded and $\beta \in \mathsf{leave\text{-}st}(\varepsilon \downarrow_R)$.
From (27), we now that $\mathsf{h} = \epsilon$. Since ε is if-free and in R-normal form, $\mathsf{leave\text{-}st}(\varepsilon \downarrow_R) = \{\varepsilon\}$. Hence $\beta \equiv \varepsilon$. We conclude by induction hypothesis and Lemma 2.
- **Inductive case, different labels:** cannot happen because P is in the fragment (27).
- **Inductive case, guard:** since we are in the pure fragment, this case cannot happen.

Finally, we ensure that no condition appears twice on the same branch using Lemma 2 and the usual proof cut elimination. □