



**HAL**  
open science

# Decidability of a Sound Set of Inference Rules for Computational Indistinguishability

Adrien Koutsos

► **To cite this version:**

Adrien Koutsos. Decidability of a Sound Set of Inference Rules for Computational Indistinguishability. ACM Transactions on Computational Logic, 2021, 22 (1), pp.1 - 44. 10.1145/3423169 . hal-03469091

**HAL Id: hal-03469091**

**<https://hal.science/hal-03469091>**

Submitted on 7 Dec 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# 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<sub>2</sub> 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. 2021. Decidability of a Sound Set of Inference Rules for Computational Indistinguishability. *ACM Trans. Comput. Logic* 22, 1, Article 3 (December 2021), 113 pages. <https://doi.org/10.1145/3423169>

## 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 are bit-strings, random numbers are sampled uniformly among bit-strings in  $\{0, 1\}^\eta$  (where  $\eta$  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

---

Author's address: Adrien Koutsos, [adrien.koutsos@inria.fr](mailto:adrien.koutsos@inria.fr), INRIA Paris, Paris, France.

2021. 1529-3785/2021/12-ART3 \$15.00  
<https://doi.org/10.1145/3423169>

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  $\psi$  expressing the security of the protocol. Showing the unsatisfiability of the conjunction of the axioms  $Ax$  and the negation of  $\psi$  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  $\psi$  of the form  $\vec{u} \sim \vec{v}$ .

**Question:** Is  $Ax \wedge \neg\psi$  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  $Ax \wedge \neg\psi$  we use resolution with a negative strategy: we see axioms in  $Ax$  as inference rules and look for a derivation of the goal  $\psi$ . 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<sup>1</sup> 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 exists 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  $\{\_ \}_{pk}$ , 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  $g(pk)$ :  $g$  is interpreted as the adversary's computation. Then the two cipher-texts, corresponding to the encryptions of the first and second components of  $g(pk)$ , should be indistinguishable. This yields the atomic axiom:

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

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

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

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

<sup>1</sup>Meaning that any ground formula derivable using the  $Ax$  is derivable using the ordered strategy.

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  $\psi$  (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  $\text{CCA}_2$  axiom schema, by allowing instances of the  $\text{CCA}_2$  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  $\{\sim_n \mid n \in \mathbb{N}\}$ , 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  $Ax \wedge \vec{u} \approx \vec{v}$  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  $\pi_1, \pi_2$ , 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

<sup>2</sup>A name is a constant function symbols used to model random samplings.

their types below:

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

Moreover all the names in  $\mathcal{N}$  have sort  $\text{term}$ , 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  $\text{term}$  is interpreted as the set of probabilistic polynomial time Turing machines equipped with a working tape and two random tapes  $\rho_1, \rho_2$ :  $\rho_1$  is for the protocol random values, while  $\rho_2$  is for the adversary random samplings. Moreover its input is of length  $\eta$ , where  $\eta$  is the security parameter.  $\text{bool}$  is the restriction of  $\text{term}$  to machines that return 0 or 1.
- A name  $n \in \mathcal{N}$  is interpreted as a machine that, on input of length  $\eta$ , extracts a word of length  $\eta$  from the first random tape  $\rho_1$ . Furthermore we require that different names extract disjoint parts of  $\rho_1$ . This ensures that distinct names are interpreted as independent random variables.
- $\text{true}$ ,  $\text{false}$ ,  $\mathbf{0}(\_)$ ,  $\text{eq}(\_, \_)$ , and  $\text{if\_then\_else\_}$  are interpreted as expected. For instance,  $\text{eq}(\_, \_)$  takes two machines  $M_1, M_2$ , and returns  $M$  such that on input  $w$  and random tapes  $\rho_1, \rho_2$ ,  $M$  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  $\llbracket g \rrbracket$  such that there is a polynomial-time Turing machine  $M_g$  such that for every machines  $(m_i)_{i \leq n}$  in the interpretation domains, and for every inputs  $w, \rho_1, \rho_2$ :

$$\llbracket g \rrbracket((m_i)_{i \leq n})(w, \rho_1, \rho_2) = M_g((m_i(w, \rho_1, \rho_2))_{i \leq n}, \rho_2)$$

Observe that  $M_g$  cannot access directly the tape  $\rho_1$ .

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

$$\begin{array}{l}
\left| \Pr(\rho_1, \rho_2 : \mathcal{A}((m_i(1^\eta, \rho_1, \rho_2))_{1 \leq i \leq n}, \rho_2) = 1) - \right. \\
\left. \Pr(\rho_1, \rho_2 : \mathcal{A}((m'_i(1^\eta, \rho_1, \rho_2))_{1 \leq i \leq n}, \rho_2) = 1) \right|
\end{array}$$

is negligible in  $\eta$  (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  $\llbracket u \rrbracket \approx \llbracket v \rrbracket$  in  $\mathcal{M}_c$ .

*Example 1.* Let  $n_0, n_1, n \in \mathcal{N}$  and  $g \in \mathcal{F}$  of arity zero. For every computational model  $\mathcal{M}_c$ :

$$\mathcal{M}_c \models \text{if } g() \text{ then } n_0 \text{ else } n_1 \sim n$$

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.  $\diamond$

### 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<sub>2</sub>).

All our axioms Ax are universally quantified Horn clauses. To show the unsatisfiability of  $\text{Ax} \wedge \vec{u} \approx \vec{v}$ , 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  $\mathcal{F}_{\mathbf{0}}$ ,  $\mathcal{F}_{\text{if}}$  and  $\mathcal{F}_{\text{if}, \mathbf{0}}$  be the subsets of  $\mathcal{F}$  defined by:

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

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:

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

But there is a simple PPTM that can distinguish between  $\langle n, n \rangle$  and  $\langle n, n' \rangle$ : 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  $\text{eq}(s, t) \sim \text{true}$  holds in all models then, using the semantics of  $\text{eq}(\_, \_)$ , in every computational model  $\mathcal{M}_c$ ,  $\llbracket s \rrbracket$  and  $\llbracket t \rrbracket$  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  $\text{eq}(s, t) \sim \text{true}$ , and we introduce a set of equalities  $R$  (given in Figure 1) and its congruence closure  $=_R$ . We split  $R$  in four



$$\begin{aligned}
R_1 & \begin{cases} \pi_i(\langle x_1, x_2 \rangle) = x_i \\ \text{dec}(\{x\}_{\text{pk}(y)}^z, \text{sk}(y)) = x \\ \text{eq}(x, x) = \text{true} \end{cases} \\
R_2 & \begin{cases} f(\vec{u}, \text{if } b \text{ then } x \text{ else } y, \vec{v}) = \text{if } b \text{ then } f(\vec{u}, x, \vec{v}) \text{ else } f(\vec{u}, y, \vec{v}) & (f \in \mathcal{F}_{\text{if}}) \\ \text{if } (\text{if } b \text{ then } a \text{ else } c) \text{ then } x \text{ else } y = \text{if } b \text{ then } (\text{if } a \text{ then } x \text{ else } y) \text{ else } (\text{if } c \text{ then } x \text{ else } y) \end{cases} \\
R_3 & \begin{cases} \text{if } b \text{ then } x \text{ else } x = x \\ \text{if } \text{true} \text{ then } x \text{ else } y = x \\ \text{if } \text{false} \text{ then } x \text{ else } y = y \\ \text{if } b \text{ then } (\text{if } b \text{ then } x \text{ else } y) \text{ else } z = \text{if } b \text{ then } x \text{ else } z \\ \text{if } b \text{ then } x \text{ else } (\text{if } b \text{ then } y \text{ else } z) = \text{if } b \text{ then } x \text{ else } z \end{cases} \\
R_4 & \begin{cases} \text{if } b \text{ then } (\text{if } a \text{ then } x \text{ else } y) \text{ else } z = \text{if } a \text{ then } (\text{if } b \text{ then } x \text{ else } z) \text{ else } (\text{if } b \text{ then } y \text{ else } z) \\ \text{if } b \text{ then } x \text{ else } (\text{if } a \text{ then } y \text{ else } z) = \text{if } a \text{ then } (\text{if } b \text{ then } x \text{ else } y) \text{ else } (\text{if } b \text{ then } x \text{ else } z) \end{cases}
\end{aligned}$$

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

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:

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

It turns out that there exists a convergent orientation  $\rightarrow_R$  of  $=_R$ .<sup>3</sup> 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.
- 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:

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

<sup>3</sup>Actually, there are many such orientations, as we will see later.

$$\begin{array}{c}
\frac{u_{\pi(1)}, \dots, u_{\pi(n)} \sim v_{\pi(1)}, \dots, v_{\pi(n)}}{u_1, \dots, u_n \sim v_1, \dots, v_n} \text{ Perm} \quad \frac{\vec{u}, t \sim \vec{v}, t'}{\vec{u} \sim \vec{v}} \text{ Restr} \quad \text{for any } s \stackrel{=}{=}_R t, \frac{\vec{u}, t \sim \vec{v}}{\vec{u}, s \sim \vec{v}} R \\
\\
\frac{\vec{u}_1, \vec{v}_1 \sim \vec{u}_2, \vec{v}_2}{f(\vec{u}_1), \vec{v}_1 \sim f(\vec{u}_2), \vec{v}_2} \text{ FA} \quad \text{where } f \in \mathcal{F}_{\setminus 0} \quad \frac{\vec{u}, t \sim \vec{v}, t'}{\vec{u}, t, t \sim \vec{v}, t', t'} \text{ Dup} \quad \frac{\vec{v} \sim \vec{u}}{\vec{u} \sim \vec{v}} \text{ Sym} \\
\\
\text{for any } b, b' \in \mathcal{T}(\mathcal{F}_{\text{if}}, \mathcal{N}), \frac{\vec{w}, b, (u_i)_i \sim \vec{w}', b', (u'_i)_i \quad \vec{w}, b, (v_i)_i \sim \vec{w}', b', (v'_i)_i}{\vec{w}, (\text{if } b \text{ then } u_i \text{ else } v_i)_i \sim \vec{w}', (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_i} \text{ CS}
\end{array}$$

**Conventions:**  $\pi$  is a permutation of  $\{1, \dots, n\}$ .

Fig. 2. The Axioms Struct-Ax.

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 yields the axiom:

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

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  $\mathcal{T}(\mathcal{F}_{\text{if}}, \mathcal{N})$ ). 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:

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

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  $\mathcal{F}_{\setminus 0}$ ); 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  $\text{IND-CCA}_2$  game (Indistinguishability against Chosen Ciphertexts Attacks [13]). First, the challenger computes a public/private key pair  $(\text{pk}(n), \text{sk}(n))$  (using a uniformly sampled nonce  $n$  of length  $\eta$ ), and sends  $\text{pk}(n)$  to the attacker. The adversary then has access to two oracles:

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

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

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

*CCA<sub>1</sub> Axiom.* Before introducing the  $\text{CCA}_2$  axioms, we recall informally the  $\text{CCA}_1$  axioms from [8]. First, we define a syntactic property on secret keys used as a side-condition of the  $\text{CCA}_1$  axioms:

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

We now define the  $\text{CCA}_1$  axioms:

*Definition 4.*  $\text{CCA}_1$  is the recursive set of atomic axioms:

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

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

- $n_r$  does not appear in  $t, u, v, \vec{w}$ .
- $n$  appears only in  $\text{pk}(n)$  or  $\text{sk}(n)$  in  $t, u, v, \vec{w}$ .
- $\text{sk}(n)$  does not appear in  $t, \vec{w}$ , and  $\text{sk}(n)$  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<sub>1</sub> is valid in every computational model where the encryption scheme interpretation is IND-CCA<sub>1</sub>.*

**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  $t[\{u\}_{\text{pk}(n)}^{n_r}]$  and  $t[\{v\}_{\text{pk}(n)}^{n_r}]$ , builds a winning adversary against the  $\text{IND-CCA}_1$  game.

We define the adversary. First, it computes  $\llbracket u \rrbracket$  and  $\llbracket v \rrbracket$ , calling the decryption oracle if necessary. It then sends them to the challenger who answers  $c$ , which is either  $\llbracket \{u\}_{\text{pk}(n)}^{n_r} \rrbracket$  or  $\llbracket \{v\}_{\text{pk}(n)}^{n_r} \rrbracket$ . 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  $\llbracket t[c] \rrbracket$ , which it can do because the secret key does not appear in  $t$ , and then returns the bit  $\mathcal{A}(\llbracket t[c] \rrbracket)$ . The advantage of the adversary is exactly the advantage of  $\mathcal{A}$ , which we assumed non-negligible, hence the adversary wins the game.  $\square$

*Remark 1.* In the  $\text{CCA}_1$  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).<sup>4</sup> To simplify the presentation, we omit all length constraints for now. We describe them later, in Appendix B.2.  $\diamond$

*CCA<sub>2</sub> Axiom.* To extend this axiom to the  $\text{IND-CCA}_2$  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  $\alpha \equiv \{u\}_{\text{pk}(n)}^{n_r}$  and  $\alpha' \equiv \{v\}_{\text{pk}(n)}^{n_r}$  (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  $\text{dec}(s, \text{sk}(n)) \sim \text{dec}(s', \text{sk}(n))$ . 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  $\alpha$  on the left (resp.  $\alpha'$  on the right): if not, we return the decryption  $\text{dec}(s, \text{sk}(n))$  (resp.  $\text{dec}(s', \text{sk}(n))$ ) asked for, otherwise we return a dummy message  $\mathbf{0}(\text{dec}(s, \text{sk}(n)))$  (resp.  $\mathbf{0}(\text{dec}(s', \text{sk}(n)))$ ).

*Definition 5.* Let  $\alpha \equiv \{u\}_{\text{pk}(n)}^{n_r}$  and  $\alpha' \equiv \{v\}_{\text{pk}(n)}^{n_r}$ , then  $\text{CCA}_s^s$  is the recursive set of atomic axioms:  $\vec{w}, t[\alpha]$ , if  $\text{eq}(t[\alpha], \alpha)$  then  $\mathbf{0}(\text{dec}(t[\alpha], \text{sk}(n))) \sim \vec{w}, t[\alpha']$ , if  $\text{eq}(t[\alpha'], \alpha')$  then  $\mathbf{0}(\text{dec}(t[\alpha'], \text{sk}(n)))$  else  $\text{dec}(t[\alpha], \text{sk}(n))$  else  $\text{dec}(t[\alpha'], \text{sk}(n))$

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  $\text{pk}(n)$  or  $\text{sk}(n)$  in  $t, u, v, \vec{w}$ .
- $\text{sk}(n)$  does not appear in  $t, \vec{w}$ , and  $\text{sk}(n)$  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  $\text{IND-CCA}_2$ .

**PROPOSITION 3.** *CCA<sub>s</sub><sup>s</sup> is valid in every computational model where the encryption scheme interpretation is  $\text{IND-CCA}_2$ .*

We do not prove validity of these axioms yet, as we actually use an extended version  $\text{CCA}_2$  of  $\text{CCA}_s^s$  (given in Appendix B) where:

- 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  $\text{CCA}_2$  axiom schema is closed under alpha-renaming of names in  $\mathcal{N}$ .
- We restrict  $t$  to be without  $\text{if\_then\_else\_}$  and  $\mathbf{0}(\_)$ . This is needed in the completeness proof.

<sup>4</sup>Basically, we add to the  $\text{CCA}_1$  axioms a premise ensuring that the lengths are the same, which we let the user axiomatize (in a restricted manner) through a length function  $\text{Length}(\_)$  and a length equality predicate  $\text{EQL}(\_, \_)$ .

- Finally, the axioms allow for an arbitrary number of public/private key pairs to be used simultaneously (e.g., with keys  $\text{pk}(n)$  and  $\text{pk}(n')$ , we have the instance  $\{A\}_{\text{pk}(n)}^{n_0}, \{B\}_{\text{pk}(n')}^{n_1} \sim \{C\}_{\text{pk}(n)}^{n_0}, \{D\}_{\text{pk}(n')}^{n_1}$ ), and an instance of the axiom can contain any number of interleaved left-right and decryption oracles calls. <sup>5</sup>

### 3.3 Transitivity

The last point is what allows us to eliminate transitivity in many examples. E.g, consider four encryptions, two of them ( $\alpha$  and  $\gamma$ ) using the public key  $\text{pk}(n)$ , and the other two ( $\beta$  and  $\delta$ ) using the public key  $\text{pk}(n')$ :

$$\alpha \equiv \{A\}_{\text{pk}(n)}^{n_0} \quad \beta \equiv \{B\}_{\text{pk}(n')}^{n_1} \quad \gamma \equiv \{C\}_{\text{pk}(n)}^{n_0} \quad \delta \equiv \{D\}_{\text{pk}(n')}^{n_1}$$

Then the following formula is a valid instance of the  $\text{CCA}_2$  axioms on, simultaneously,  $\text{pk}(n)$  and  $\text{pk}(n')$ :

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

However, proving the above formula using  $\text{CCA}_2$  only on one key at a time, as in [8], requires transitivity:

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

*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 where an agent sends a message, say 0, encrypted with a key  $\text{pk}(n)$ , and then encrypts the corresponding secret key  $\text{sk}(n)$ , with some long-term key  $\text{pk}(n_t)$ . To prove the secrecy of the message sent, consider to goal:

$$\{0\}_{\text{pk}(n)}^{n_0}, \{\text{sk}(n)\}_{\text{pk}(n_t)}^{n_1} \sim \{1\}_{\text{pk}(n)}^{n_0}, \{\text{sk}(n)\}_{\text{pk}(n_t)}^{n_1}$$

where we changed the message encrypted from 0 to 1. We cannot apply  $\text{CCA}_2$  on  $\text{pk}(n)$  because the secret key appears outside a decryption position. And we cannot apply  $\text{CCA}_2$  only on the key  $\text{pk}(n_t)$ , 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  $\text{sk}(n)$  by  $\text{sk}(n')$  (where  $n'$  is a fresh name) using  $\text{CCA}_2$  on  $\text{pk}(n_t)$ , and then applying  $\text{CCA}_2$  on key  $\text{pk}(n)$ .

### 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 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.

<sup>5</sup>Axioms for the IND- $\text{CCA}_2$  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.

$$\begin{array}{c}
\frac{\{0\}_{pk(n)}^{n_e^1}, \{sk(n)\}_{pk(n_{it})}^{n_e^2} \quad \text{CCA}_2(pk(n_{it}))}{\sim \{0\}_{pk(n)}^{n_e^1}, \{sk(n')\}_{pk(n_{it})}^{n_e^2}} \quad \text{CCA}_2(pk(n)) \\
\{1\}_{pk(n)}^{n_e^1}, \{sk(n)\}_{pk(n_{it})}^{n_e^2} \quad \text{CCA}_2(pk(n_{it})) \quad \sim \quad \{1\}_{pk(n)}^{n_e^1}, \{sk(n')\}_{pk(n_{it})}^{n_e^2}
\end{array}$$

Fig. 3. Derivation using the transitivity rule.

$$\begin{array}{c}
\frac{\frac{\frac{\{0\}_{pk_A}^{n_A}, pk_B, n_B, sk_B, n}{\sim \{1\}_{pk_A}^{n_A}, pk_B, n_B, sk_B, n} \quad \text{CCA}_2(pk_A, sk_A)}{\{\{0\}_{pk_A}^{n_A}\}_{pk_B}^{n_B}, sk_B, n} \quad \text{FA}}{\sim \{\{1\}_{pk_A}^{n_A}\}_{pk_B}^{n_B}, sk_B, n} \quad \text{FA}^{(2)} + \text{Dup}}{\frac{\frac{\{\{0\}_{pk_A}^{n_A}\}_{pk_B}^{n_B}, dec(t_0, sk_B), n}{\sim \{\{1\}_{pk_A}^{n_A}\}_{pk_B}^{n_B}, dec(t_1, sk_B), n} \quad \text{FA}^{(3)} + \text{Dup}}{eq(t_0, \{\{0\}_{pk_A}^{n_A}\}_{pk_B}^{n_B}), \langle dec(t_0, sk_B), n \rangle} \quad \text{FA}^{(3)} + \text{Dup}}{\sim} \\
eq(t_1, \{\{1\}_{pk_A}^{n_A}\}_{pk_B}^{n_B}), \langle dec(t_1, sk_B), n \rangle} \quad \text{CS} \\
\frac{\text{if } eq(t_0, \{\{0\}_{pk_A}^{n_A}\}_{pk_B}^{n_B}) \text{ then } \langle dec(t_0, sk_B), n \rangle \sim \text{if } eq(t_1, \{\{1\}_{pk_A}^{n_A}\}_{pk_B}^{n_B}) \text{ then } \langle dec(t_1, sk_B), n \rangle}{\text{else } \langle s_0, n \rangle \quad \text{else } \langle s_1, n_A \rangle} \quad \text{R} \\
\frac{\langle dec(t_0, sk_B), n \rangle \sim \text{if } eq(t_1, \{\{1\}_{pk_A}^{n_A}\}_{pk_B}^{n_B}) \text{ then } \langle dec(t_1, sk_B), n \rangle}{\text{else } \langle dec(t_1, sk_B), n_A \rangle} \quad \text{R}
\end{array}$$

**Remark:** in the right branch, when we apply the  $\text{CCA}_2$  axiom to:

$$\{\{0\}_{pk_A}^{n_A}\}_{pk_B}^{n_B}, s_0, n \sim \{\{1\}_{pk_A}^{n_A}\}_{pk_B}^{n_B}, s_1, n_A$$

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

Fig. 4. Derivation of  $\phi$ .

*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  $\text{CCA}_2$ , therefore we do not include it in  $Ax$ ).

$$\frac{\frac{\frac{\overline{g(), n_0 \sim g(), n} \quad \text{Refl} \quad \overline{g(), n_1 \sim g(), n}}{\text{if } g() \text{ then } n_0 \text{ else } n_1 \sim \text{if } g() \text{ then } n \text{ else } n} \quad \text{CS}}{\text{if } g() \text{ then } n_0 \text{ else } n_1 \sim n} \quad \text{R}$$

◇

*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  $pk_A \equiv pk(n_0)$ ,  $sk_A \equiv sk(n_0)$ ,  $pk_B \equiv pk(n_1)$  and  $sk_B \equiv sk(n_1)$ . Let  $t_m \equiv g(\{\{m\}_{pk_A}^{n_A}\}_{pk_B}^{n_B})$ . Consider the formula below:

$$\phi \equiv \langle \text{dec}(t_0, sk_B), n \rangle \sim \text{if eq}(t_1, \{\{1\}_{pk_A}^{n_A}\}_{pk_B}^{n_B}) \text{ then } \langle \text{dec}(t_1, sk_B), n \rangle \\ \text{else } \langle \text{dec}(t_1, sk_B), n_A \rangle$$

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  $\text{dec}(t_1, sk_B)$  is under the correct guard and to apply  $CCA_2$  on keys  $(pk_B, sk_B)$ . For the then branch, we can directly use the  $CCA_2$  axioms on keys  $(pk_A, sk_A)$ . Let  $s_m$  be the following term:

$$s_m \equiv \text{if eq}(t_m, \{\{m\}_{pk_A}^{n_A}\}_{pk_B}^{n_B}) \text{ then } \mathbf{0}(\text{dec}(t_m, sk_B)) \text{ else } \text{dec}(t_m, sk_B)$$

We give the derivation of  $\phi$  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  $pk_A \equiv pk(n_A)$  and  $sk_A \equiv sk(n_A)$  be the public/private key pair of agent A (we define similarly  $(pk_B, sk_B)$ ). Since A does not wait for any input before sending its first message, we put it into the initial frame:

$$\phi_0 \equiv pk_A, pk_B, \{\langle n_A, A \rangle\}_{pk_B}^{n_0}$$

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

$$t_A[\mathbf{x}_A] \equiv \text{if eq}(\pi_1(\text{dec}(\mathbf{x}_A, sk_A)), n_A) \quad \text{then} \\ \text{if eq}(\pi_2(\pi_2(\text{dec}(\mathbf{x}_A, sk_A))), B) \text{ then} \\ \{\pi_1(\pi_2(\text{dec}(\mathbf{x}_A, sk_A)))\}_{pk_B}^{n_2} \\ t_B[\mathbf{x}_B] \equiv \text{if eq}(\pi_2(\text{dec}(\mathbf{x}_B, sk_B)), A) \quad \text{then} \\ \{\langle \pi_1(\text{dec}(\mathbf{x}_B, sk_B)), \langle n_B, B \rangle \rangle\}_{pk_A}^{n_1}$$

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  $\{\langle n_A, A \rangle\}_{pk_B}^{n_0}$  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  $\mathbf{x}_B \equiv \{\langle n_A, A \rangle\}_{pk_B}^{n_0}$ , and B answers:

$$t_B[\mathbf{x}_B] =_R \{\langle n_A, \langle n_B, B \rangle \rangle\}_{pk_A}^{n_1}$$

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

$$s \equiv \text{if eq}(g(\phi_1), t_B[\mathbf{x}_B]) \text{ then } \mathbf{0} \text{ else } t_A[g(\phi_1)] \quad (1)$$

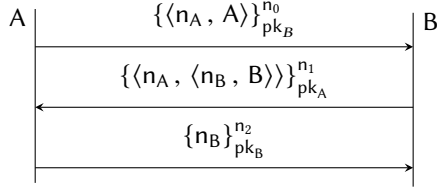


Fig. 5. The NSL protocol.

We show the secrecy of the nonce  $n_B$ : we let  $t'_B[\mathbf{x}_B]$  (resp.  $s'$ ) be the term  $t_B[\mathbf{x}_B]$  (resp.  $s$ ) where we replaced all occurrences of  $n_B$  by 0. For example,  $t'_B[\mathbf{x}_B] =_R \{\langle n_A, \langle 0, B \rangle \rangle\}_{pk_A}^{n_1}$ . This yields the goal:

$$\phi_0, t_B[\mathbf{x}_B], s \sim \phi_0, t'_B[\mathbf{x}_B], s' \quad (2)$$

We let  $\delta$  be the guarded decryption that will be used in the  $CCA_2$  axiom:

$$\delta \equiv \text{if eq}(g(\phi_1), t_B[\mathbf{x}_B]) \text{ then } \mathbf{0}(\text{dec}(g(\phi_1), sk_A)) \text{ else } \text{dec}(g(\phi_1), sk_A) \quad (3)$$

and  $s_\delta$  be the term  $s$  where all occurrences of  $\text{dec}(g(\phi_1), sk_A)$  have been replaced by  $\delta$ . We have  $s =_R s_\delta$ . We also introduce shorthands for some subterms of  $s_\delta$ : we let  $a_\delta, b_\delta$  and  $e_\delta$  be the terms  $\text{eq}(\pi_1(\delta), n_A), \text{eq}(\pi_2(\pi_2(\delta))), B$  and  $\{\pi_1(\pi_2(\delta))\}_{pk_B}^{n_2}$ . We define  $\delta', s'_{\delta'}, a'_{\delta'}, b'_{\delta'}$ , and  $e'_{\delta'}$ , similarly.

We then rewrite  $s$  and  $s'$  into  $s_\delta$  and  $s'_{\delta'}$ , using  $R$ . Then we apply FA several times, first to deconstruct  $s_\delta$  and  $s'_{\delta'}$ , and then to deconstruct  $a_\delta, b_\delta$  and  $a'_{\delta'}, b'_{\delta'}$ . Finally, we use Dup to remove duplicates, and we apply  $CCA_2$  simultaneously on key pairs  $(pk_A, sk_A)$  and  $(pk_B, sk_B)$  (we omit here the details of the syntactic side-conditions that have to be checked):

$$\frac{\frac{\frac{\phi_0, t_B[\mathbf{x}_B], n_A, \delta, e_\delta \sim \phi_0, t'_B[\mathbf{x}_B], n_A, \delta', e'_{\delta'}}{(\text{FA} + \text{Dup})^*} \quad \phi_0, t_B[\mathbf{x}_B], a_\delta, b_\delta, e_\delta \sim \phi_0, t'_B[\mathbf{x}_B], a'_{\delta'}, b'_{\delta'}, e'_{\delta'}}{(\text{FA} + \text{Dup})^*} \quad \phi_0, t_B[\mathbf{x}_B], s_\delta \sim \phi_0, t'_B[\mathbf{x}_B], s'_{\delta'}}{R} \quad \phi_0, t_B[\mathbf{x}_B], s \sim \phi_0, t'_B[\mathbf{x}_B], s' \quad \diamond$$

*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  $\text{eq}(g(\phi_1), t_B[\mathbf{x}_B])$  (which does not contain nested conditions, as required by the CS side-condition).  $\diamond$

### 3.5 Comparison with Other Axiomatizations

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

<sup>6</sup>There are other axiomatizations 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.



axiom EqCong, which states that the relation  $=$  (which is syntactic sugar for  $\text{eq}(\_, \_) \sim \text{true}$ ) is a congruence relation:

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

This axiom (actually axioms) allow to do equational reasoning inside the logic, using the  $\sim$  predicate and the  $\text{eq}(\_, \_)$  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  $\sim$  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  $\sim$  axioms to prove an equality  $\text{eq}(x, y) \sim \text{true}$ , while we only allow for a fixed set of equalities in  $=_R$ .

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

$$\frac{\vec{u} \sim \vec{v} \quad \vec{v} \sim \vec{w}}{\vec{u} \sim \vec{w}} \text{Trans} \quad \frac{}{\text{true} \approx \text{false}} \text{TFDist} \quad \frac{\vec{u} \sim \vec{v} \quad \text{if } n \notin \text{st}(\vec{u}), n' \notin \text{st}(\vec{v})}{\vec{u}, n \sim \vec{v}, n'} \text{FreshInd}$$

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

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.<sup>7</sup> FreshNEq states that if  $n$  is independent from  $t$  (i.e.  $n \notin \text{st}(t)$ ), 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:

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

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  $\text{eq}(u, v)$  could have been introduced earlier, e.g. using the rewriting  $t[u] =_r \text{if } \text{eq}(u, v) \text{ then } t[u] \text{ else } t[u]$ . For these reasons, automatically handling the rule in (4) seems very challenging.

#### 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  $>_f$  be a precedence over function symbols (i.e. an order over function symbols). The lexicographic path ordering  $>$  associated with  $>_f$  is the relation defined by:

$$s = f(s_1, \dots, s_n) > t = g(t_1, \dots, t_m) \text{ iff } \begin{cases} \exists i \in \llbracket 1, n \rrbracket \text{ s.t. } s_i \geq t \\ \text{or } f = g \wedge \forall j \in \llbracket 1, m \rrbracket, s > t_j \wedge s_1, \dots, s_n >_{\text{lex}} t_1, \dots, t_n \\ \text{or } f >_f g \wedge \forall j \in \llbracket 1, m \rrbracket, s > t_j \end{cases}$$

<sup>7</sup>Actually, we believe that  $\text{Ax} \cup \text{FreshInd}$  is equivalent to  $\text{Ax}$ , though we did not prove it.

where  $>_{lex}$  is the lexicographic ordering obtained from  $>$ .

**PROPOSITION 4** (DERSHOWITZ, JOUANNAUD [26]). *If  $>_f$  is a total precedence, then the corresponding LPO  $>$  is a reduction ordering:  $>$  is a well-founded order closed under substitutions and context application. Moreover,  $>$  is a total ordering on ground terms.*

Let  $>_f$  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  $>$  on ground terms using  $>_f$ .

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

Now, we want to have some leeway in the ordering of terms. We do this by letting  $>_u$  be an arbitrary total order on if-free conditions that are  $R_{\leq 3}$ -irreducible. We define the extension  $>_u^{lpo}$  of  $>_u$  to arbitrary ground conditions. Basically,  $>_u^{lpo}$  compares if-free  $R_{\leq 3}$ -irreducible conditions using  $>_u$ ; conditions that are *not* if-free or *not*  $R_{\leq 3}$ -irreducible are compared using  $>$ ; and we choose the behavior of  $>_u^{lpo}$  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  $>_u$  on ground if-free  $R_{\leq 3}$ -irreducible terms, we let  $>_u^{lpo}$  be the relation defined on ground terms by:

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

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

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

$$\begin{aligned} \text{if } b \text{ then (if } a \text{ then } x \text{ else } y) \text{ else } z &\rightarrow \text{if } a \text{ then (if } b \text{ then } x \text{ else } z) && \text{(when } b >_u^{lpo} a) \\ &\text{else (if } b \text{ then } y \text{ else } z) \\ \text{if } b \text{ then } x \text{ else (if } a \text{ then } y \text{ else } z) &\rightarrow \text{if } a \text{ then (if } b \text{ then } x \text{ else } y) && \text{(when } b >_u^{lpo} a) \\ &\text{else (if } b \text{ then } x \text{ else } z) \end{aligned}$$

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

Moreover, we let  $\rightarrow_{R^{>_u}} = \rightarrow_{R_1} \cup \rightarrow_{R_2} \cup \rightarrow_{R_3} \cup \rightarrow_{R_4}^{>_u}$ .

The term rewriting system  $\rightarrow_{R^{>_u}}$  is an orientation of the rules given in Figure 1 on ground terms. When the ordering  $>_u$  is irrelevant, we write  $\rightarrow_R$  instead of  $\rightarrow_{R^{>_u}}$ . We state the convergence theorem.

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

Observe that this result subsumes Theorem 1.

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

$$\begin{aligned}
& \rightarrow_{R'_2} \{ f(\vec{u}, \text{if}_b(x, y), \vec{v}) \rightarrow \text{if}_b(f(\vec{u}, x, \vec{v}), f(\vec{u}, y, \vec{v})) \quad (f \in \mathcal{F}_{\setminus \text{if}}) \\
& \rightarrow_{R'_3} \left\{ \begin{array}{l} \text{if}_{\text{true}}(x, y) \rightarrow x \\ \text{if}_{\text{false}}(x, y) \rightarrow y \\ \text{if}_b(x, x) \rightarrow x \\ \text{if}_b(\text{if}_b(x, y), z) \rightarrow \text{if}_b(x, z) \\ \text{if}_b(x, (\text{if}_b(y, z))) \rightarrow \text{if}_b(x, z) \end{array} \right. \\
& \rightarrow_{R'_4} \left\{ \begin{array}{l} \text{if } b \text{ then } (\text{if } a \text{ then } x \text{ else } y) \text{ else } z \rightarrow \text{if } a \text{ then } (\text{if } b \text{ then } x \text{ else } z) \text{ else } (\text{if } b \text{ then } y \text{ else } z) \\ \hspace{15em} (b > a, a, b \text{ not if-free or not } R_{\leq 3}\text{-irreducible}) \\ \text{if } b \text{ then } x \text{ else } (\text{if } a \text{ then } y \text{ else } z) \rightarrow \text{if } a \text{ then } (\text{if } b \text{ then } x \text{ else } y) \text{ else } (\text{if } b \text{ then } x \text{ else } z) \\ \hspace{15em} (b > a, a, b \text{ not if-free or not } R_{\leq 3}\text{-irreducible}) \end{array} \right. \\
& \rightarrow_{R^0_4} \left\{ \begin{array}{l} \text{if } b \text{ then } (\text{if}_a(x, y)) \text{ else } z \rightarrow \text{if}_a((\text{if } b \text{ then } x \text{ else } z), (\text{if } b \text{ then } y \text{ else } z)) \\ \hspace{15em} (b \text{ not if-free or not } R_{\leq 3}\text{-irreducible}) \\ \text{if } b \text{ then } x \text{ else } (\text{if}_a(y, z)) \rightarrow \text{if}_a((\text{if } b \text{ then } x \text{ else } y), (\text{if } b \text{ then } x \text{ else } z)) \\ \hspace{15em} (b \text{ not if-free or not } R_{\leq 3}\text{-irreducible}) \end{array} \right. \\
& \rightarrow_{R^1_4} \left\{ \begin{array}{l} \text{if}_b((\text{if}_a(x, y)), z) \rightarrow \text{if}_a((\text{if}_b(x, z)), (\text{if}_b(y, z))) \quad (b >_u a) \\ \text{if}_b(x, (\text{if}_a(y, z))) \rightarrow \text{if}_a((\text{if}_b(x, y)), (\text{if}_b(x, z))) \quad (b >_u a) \end{array} \right. \\
& \rightarrow_{R^i} \{ \text{if } b \text{ then } u \text{ else } v \rightarrow \text{if}_b(u, v) \quad (b \text{ if-free and } R_{\leq 3}\text{-irreducible})
\end{aligned}$$

Fig. 6. The Relations  $\rightarrow_{R'_2}, \rightarrow_{R'_3}, \rightarrow_{R^0_4}, \rightarrow_{R^1_4}, \rightarrow_{R^2_4}$  and  $\rightarrow_{R^i}$  used for termination

*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  $\mathcal{F}_{\text{term}}$  be the signature  $\mathcal{F}$  to which we added a symbol  $\text{if}_b(, )$  for every if-free  $R_{\leq 3}$ -irreducible condition  $b$ :

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

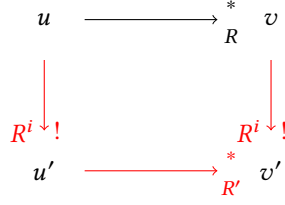
This yields an infinite countable signature. We extend the precedence  $>_f$  to  $\mathcal{F}_{\text{term}} \cup \mathcal{N}$  by having the function symbols  $\{ \text{if}_b(, ) \}$  be smaller than all the other function symbols, and  $\text{if}_b(, ) >_f \text{if}_a(, )$  if and only if  $b >_u a$ . Observe that the extended precedence is still a total order.

We then consider the term rewriting system  $\rightarrow_{R'}$  on  $\mathcal{T}(\mathcal{F}_{\text{term}}, \mathcal{N})$ , defined by removing  $\rightarrow_{R_4}$  from  $\rightarrow_R$  and adding all the rules in Figure 6:

$$\rightarrow_{R'} = \rightarrow_{R_1} \cup \rightarrow_{R_2} \cup \rightarrow_{R'_2} \cup \rightarrow_{R_3} \cup \rightarrow_{R'_3} \cup \rightarrow_{R^0_4} \cup \rightarrow_{R^1_4} \cup \rightarrow_{R^2_4} \cup \rightarrow_{R^i}$$

One can easily (but tediously) check that  $>$  is compatible with  $\rightarrow_{R'}$ : the only non-trivial cases are the cases in  $\rightarrow_{R_2}$  (the first rule is decreasing because  $f >_f \text{if\_then\_else\_}$ , the second rule using the lexicographic order), in  $\rightarrow_{R'_2}$  (same arguments than for  $R_2$ ) and the cases in  $\rightarrow_{R^0_4}, \rightarrow_{R^1_4}, \rightarrow_{R^2_4}$  (where we use the side conditions  $b > a, b >_u a \dots$ ).

Since  $>$  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.



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

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

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

*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.

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

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

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

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

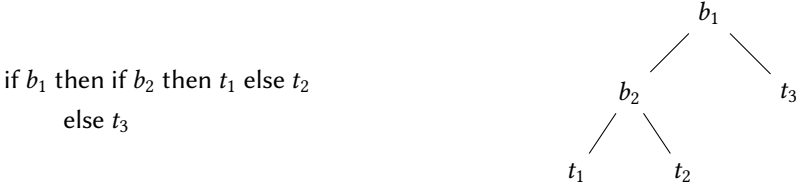
*Example 5.* Let  $\vec{x} = x_1, x_2, x_3$  and  $\vec{y} = y_1, y_2, y_3, y_4$ , we can define the if-context  $D[\ ]_{\vec{x} \diamond \vec{y}}$ :

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

The normal form of term  $t$  by  $\xrightarrow{R^>u}$  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  $t \downarrow_{R^>u}$ , 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  $\text{cond-st}(t)$  be the set of conditions  $\vec{b}$ , and  $\text{leave-st}(t)$  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):



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

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

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

$$\text{leave-st}(t \downarrow_{R>u}) = \text{leave-st}(t \downarrow_{R>'_u}) \quad \text{and} \quad \text{cond-st}(t \downarrow_{R>u}) = \text{cond-st}(t \downarrow_{R>'_u})$$

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

$$\vec{b} = \text{leave-st}(t \downarrow_{R>u}) \quad \vec{u} = \text{cond-st}(t \downarrow_{R>u}) \quad \vec{b}' = \text{leave-st}(t \downarrow_{R>'_u}) \quad \vec{u}' = \text{cond-st}(t \downarrow_{R>'_u})$$

We know that  $C[\vec{b} \diamond \vec{u}] \xrightarrow{*}_{R>'_u} C'[\vec{b}' \diamond \vec{u}']$ . 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:

$$\begin{aligned} & \text{if } b \text{ then } x \text{ else } x \rightarrow x \\ & \text{if true then } x \text{ else } y \rightarrow x \\ & \text{if false then } x \text{ else } y \rightarrow y \\ & \text{if } b \text{ then (if } b \text{ then } x \text{ else } y) \text{ else } z \rightarrow \text{if } b \text{ then } x \text{ else } z \\ & \text{if } b \text{ then } x \text{ else (if } b \text{ then } y \text{ else } z) \rightarrow \text{if } b \text{ then } x \text{ else } z \\ & \text{if } b \text{ then (if } a \text{ then } x \text{ else } y) \text{ else } z \rightarrow \text{if } a \text{ then (if } b \text{ then } x \text{ else } z) \\ & \quad \text{else (if } b \text{ then } y \text{ else } z) \quad (\text{when } b \succ_u^{\text{lpo}} a) \\ & \text{if } b \text{ then } x \text{ else (if } a \text{ then } y \text{ else } z) \rightarrow \text{if } a \text{ then (if } b \text{ then } x \text{ else } y) \\ & \quad \text{else (if } b \text{ then } x \text{ else } z) \quad (\text{when } b \succ_u^{\text{lpo}} a) \end{aligned}$$

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  $C'[\vec{b}' \diamond \vec{u}'] \xrightarrow{*}_{R>u} C[\vec{b} \diamond \vec{u}]$ , 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  $\text{leave-st}(t \downarrow_R)$  and  $\text{cond-st}(t \downarrow_R)$  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  $\text{Struct-Ax}$  and  $\text{CCA}_2$ . 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  $Ax \wedge \vec{u} \approx \vec{v}$  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  $\text{Sym}$  rule commutes with all the other rules, and the  $\text{CCA}_2$  axiom schema is closed under  $\text{Sym}$ .

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  $\phi$ , we write  $P \vdash \phi$  if  $P$  is a proof of  $\phi$ .

*Definition 13.* Let  $\Sigma$  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  $\phi$  is in the fragment iff there exists a proof  $P$  such that  $P \vdash \phi$  and, for every branch  $\rho$  of  $P$ , the word  $w$  obtained by collecting the axiom names along  $\rho$  (starting from the root) is in  $\mathcal{L}$ .

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

$$\mathfrak{F}(R \cdot CS \cdot \text{Refl}) \quad \text{and} \quad \mathfrak{F}(R \cdot CS \cdot (FA + Dup)^* \cdot CCA_2) \quad \diamond$$

*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:

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

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  $\text{dec}(s, sk)$  are of the form  $\text{eq}(s, \alpha)$  where  $\alpha$  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 an 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:

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

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  $s \sim t$ .  $\diamond$

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:*

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

then there exists  $P'$  such that  $P' \vdash \vec{u} \sim \vec{v}$  and  $P'$  contains no Restr applications. Moreover:

- the height of  $P'$  is no larger than the height of  $P$ .
- 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:

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

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:

$$\frac{\vec{u}, \vec{t} \sim \vec{v}, \vec{s}}{\vec{u} \sim \vec{v}} \text{ Restr}$$

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:

$$\frac{\begin{array}{c} \vdots (A) \\ \vec{u}, \vec{u}', \vec{t} \sim \vec{v}, \vec{v}', \vec{t}' \end{array}}{f(\vec{u}), \vec{u}', \vec{t} \sim f(\vec{v}), \vec{v}', \vec{t}'} \text{ FA} \\ \frac{\quad}{f(\vec{u}), \vec{u}' \sim f(\vec{v}), \vec{v}'} \text{ Restr}$$

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:

$$\frac{\begin{array}{c} \vdots (A) \\ \vec{u}, \vec{u}', \vec{t} \sim \vec{v}, \vec{v}', \vec{t}' \end{array}}{\vec{u}, \vec{u}' \sim \vec{v}, \vec{v}'} \text{ Restr} \xrightarrow{\text{ind. hyp.}} \begin{array}{c} \vdots (A') \\ \vec{u}, \vec{u}' \sim \vec{v}, \vec{v}' \end{array} \xrightarrow{\text{apply FA}} \frac{\begin{array}{c} \vdots (A') \\ \vec{u}, \vec{u}' \sim \vec{v}, \vec{v}' \end{array}}{f(\vec{u}), \vec{u}' \sim f(\vec{v}), \vec{v}'} \text{ FA}$$

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

$$\frac{\begin{array}{c} \vdots (A) \\ \vec{u}, \vec{u}', \vec{u}'' \sim \vec{v}, \vec{v}', \vec{v}'' \end{array}}{\vec{u}, \vec{u}', f(\vec{u}'') \sim \vec{v}, \vec{v}', f(\vec{v}'')} \text{ FA} \\ \frac{\quad}{\vec{u} \sim \vec{v}} \text{ Restr}$$

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

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

- If it is CS:

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

We apply the induction hypothesis twice:

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

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

We obtain the derivation:

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

- 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  $\mathfrak{F}(\mathcal{L})$ .  $\square$

*Remark 3.* In the proof, we need the atomic axioms  $U$  to be closed under Restr. Therefore, we are going to close the  $\text{CCA}_2$  axiom schema under Restr. This adds a new difficulty: let  $\text{CCA}_2^a$  be the axiom schema before closing it under Restr, then a formula  $\vec{u} \sim \vec{v}$  is an instance of  $\text{CCA}_2$  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  $\text{CCA}_2^a$ :

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

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  $\text{CCA}_2$ , 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 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  $\text{CCA}_2$ : 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  $\text{CCA}_2^a$  (which can be done in polynomial time).  $\diamond$

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:

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

where  $p \equiv \text{if } c \text{ then } s \text{ else } t$ . 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,  $\llbracket b, b \rrbracket \approx \llbracket c, d \rrbracket$ . Therefore, no adversary can distinguish between getting twice the same value sampled from  $\llbracket b \rrbracket$  and getting a pair of values sampled from  $\llbracket c, d \rrbracket$ . In particular,  $\llbracket c \rrbracket_{\eta, \rho} = \llbracket d \rrbracket_{\eta, \rho}$ , except for a negligible number of random tapes  $\rho$ .  $\diamond$

*A First Key Lemma.* A natural question is to ask whether the semantic equality between  $\llbracket c \rrbracket$  and  $\llbracket d \rrbracket$  implies a syntactic equality.<sup>8</sup> 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  $\text{FA}_s = \{\text{FA}_f \mid f \in \mathcal{F}_{\setminus \text{if}, \mathbf{0}}\}$  be the restriction of FA to function symbols different from `if_then_else_`. Formulas that can be proven in the fragment  $\mathfrak{F}(\text{FA}_s^* \cdot \text{Dup}^* \cdot \text{CCA}_2)$  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  $\mathfrak{F}(\text{FA}_s^* \cdot \text{Dup}^* \cdot \text{CCA}_2)$  then  $b \equiv C[\vec{w}, (\alpha_i)_i, (\text{dec}_j)_j]$ ,  $b' \equiv C[\vec{w}, (\alpha'_i)_i, (\text{dec}'_j)_j]$  and the  $\text{CCA}_2$  instance applied is (up-to  $\alpha$ -renaming):*

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

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

**PROOF.** This is easy immediate by induction on the proof derivation.  $\square$

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  $\mathfrak{F}(\text{FA}_s^* \cdot \text{Dup}^* \cdot \text{CCA}_2)$  then  $b' \equiv b''$ .*

**PROOF.** From Proposition 6 we have:

$$\begin{array}{ll}
 b \equiv C^l[\vec{w}^l, (\alpha_i^l)_{i \in I^l}, (\text{dec}_j^l)_{j \in J^l}] & b' \equiv C^l[\vec{w}^l, (\alpha_i^l)_{i \in I^l}, (\text{dec}_j^l)_{j \in J^l}] \\
 b \equiv C^r[\vec{w}^r, (\alpha_i^r)_{i \in I^r}, (\text{dec}_j^r)_{j \in J^r}] & b'' \equiv C^r[\vec{w}^r, (\alpha_i^r)_{i \in I^r}, (\text{dec}_j^r)_{j \in J^r}]
 \end{array}$$

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.

<sup>8</sup>We say that  $\llbracket c \rrbracket$  and  $\llbracket d \rrbracket$  are semantically equal if the bit-strings distributions  $\llbracket c \rrbracket$  and  $\llbracket d \rrbracket$  are equal, except for a negligible number of samplings.

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

$$\frac{\frac{\dots, n_e \sim \dots, n'_e}{\dots, m, pk(n), n_e \sim \dots, m', pk(n), n_e} CCA_2}{\dots, \{m\}_{pk(n)}^{n_e} \sim \dots, \{m'\}_{pk(n)}^{n_e}} FA$$

Since  $n_e$  is a name in  $\mathcal{N}$  and cannot be modified by any rules in  $\{R, FA_s, Dup\}$ . Therefore  $n_e \in \vec{w}^r$ . This contradict the freshness side-condition. Absurd.

- If the hole at  $b_{|p}$  is mapped to a decryption oracle call  $dec_{i_0}^l$  in  $b$ . Since  $C_{|p}^r$  is not a hole, and since function applications on  $FA_s$  cannot be applied on the `if_then_else_` function symbols we know that there exists  $m, m'$  such that  $dec_{i_0}^l \equiv dec(m, sk(n))$  and  $dec_{i_0}^l \equiv dec(m', sk(n))$ . Moreover, since  $dec_{i_0}^l$  is a decryption in the  $CCA_2$  application, we know from the key-usability side-condition that  $sk(n)$  appears only in decryption position in  $\vec{w}^r$ . Then the reasoning we have in the previous cases applies here. Indeed, we know that  $C_{|p}^r$  is not a hole, hence the proof of  $b, b \sim b', b''$  includes one of the following sub-proofs:

$$\frac{\frac{\dots, sk(n) \sim \dots, sk(n)}{\dots, m, sk(n) \sim \dots, m', sk(n)} CCA_2}{\dots, dec(m, sk(n)) \sim \dots, dec(m', sk(n))} FA \quad \text{or} \quad \frac{\frac{\dots, m, n \sim \dots, m', n}{\dots, m, sk(n) \sim \dots, m', sk(n)} FA_{sk}}{\dots, m, sk(n) \sim \dots, m', sk(n)} FA$$

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

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  $\mathfrak{F}(FA_s^* \cdot Dup^* \cdot CCA_2)$ . 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:

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

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

$$\frac{\frac{\frac{\vdots (A')}{a_1, u_4, v_7 \sim c_1, s_4, p_7} FA}{\begin{array}{c} a_1 \\ / \quad \backslash \\ u_4 \quad v_7 \end{array}} \sim \begin{array}{c} c_1 \\ / \quad \backslash \\ s_4 \quad c \\ \quad / \quad \backslash \\ \quad s \quad t \end{array}}{\text{if } a \text{ then } u \text{ else } v \sim \text{if } c \text{ then } s \text{ else } t} R$$

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  $\alpha$ -bounded. Intuitively a condition is  $\alpha$ -bounded if it is a subterm of the conclusion or if it guards a decryption appearing in an  $\alpha$ -bounded term.
- **Decision Procedure:** we give a procedure that, given a goal formula  $t \sim t'$ , computes the set of  $\alpha$ -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:*

$Dup \cdot R \Rightarrow R \cdot Dup$	$FA \cdot R \Rightarrow R \cdot FA$
$Dup \cdot FA \Rightarrow FA^* \cdot Dup$	$FA \cdot CS \Rightarrow R \cdot CS \cdot FA$
$Dup \cdot CS \Rightarrow CS \cdot Dup$	

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

PROOF. Using Lemma 3, we commute all the *Dup* to the right, which yields  $\mathfrak{F}((CS + R + FA)^* \cdot Dup^* \cdot U)$ . Then, we commute all *FA* to the right, stopping at the first *Dup*. □

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

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

*Delay FA.*

- $FA \cdot CS \Rightarrow R \cdot CS \cdot FA$ :

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

Can be rewritten into:

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

- $FA \cdot R \Rightarrow R \cdot FA$ :

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

*Delay Dup.*

- $Dup \cdot R \Rightarrow R \cdot Dup$ .

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  $t' =_R s'$ :

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

- $Dup \cdot FA \Rightarrow FA^* \cdot Dup$ .

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

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

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

Fig. 8. Function Application and Duplicate Rules Commutations

*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  $FA(b, b')$ , where  $b, b'$  are the involved conditions; if the deconstructed symbol  $f$  is in  $\mathcal{F}_{\text{if},0}$ , then we denote the

function application by  $FA_f$ . We give below the two new rules:

$$\frac{\vec{w}, a, u, v \sim \vec{r}, b, s, t}{\vec{w}, \text{if } a \text{ then } u \text{ else } v \sim \vec{r}, \text{if } b \text{ then } s \text{ else } t} FA(b, b') \quad \frac{\vec{u}, \vec{v} \sim \vec{s}, \vec{t}}{\vec{u}, f(\vec{v}) \sim \vec{s}, f(\vec{t})} FA_f$$

The set of rule names is now infinite, since there is a rule  $FA(b, b')$  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:

$$\frac{\frac{a_1, u_1 \sim c_1, s_1}{a, u \sim c, s} R \quad \frac{a_2, v_1 \sim c_2, t_1}{a, v \sim c, t} R}{\text{if } a \text{ then } u \text{ else } v \sim \text{if } c \text{ then } s \text{ else } t} CS \quad (5)$$

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:

$$\frac{\vec{u}, C[\boxed{a} \boxed{a}]_a \sim \vec{v}, C'[\boxed{c} \boxed{c}]_c}{\vec{u}, C[a] \sim \vec{v}, C'[c]} 2Box^s \quad \frac{a_1, u \sim c_1, s \quad a_2, v \sim c_2, t}{\text{if } \boxed{a_1} \boxed{a_2}]_a \text{ then } u \text{ else } v \sim \text{if } \boxed{c_1} \boxed{c_2}]_c \text{ then } s \text{ else } t} CS_{\square}^s$$

where  $\boxed{\quad} \boxed{\quad}_a$  is a new symbol of sort  $\text{bool}^2 \rightarrow \text{bool}$ , and of fixed semantics: it ignores its arguments and has the semantics  $\llbracket a \rrbracket$ . 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):

$$\frac{\frac{a_1, u_1 \sim c_1, s_1 \quad a_2, v_1 \sim c_2, t_1}{\text{if } \boxed{a_1} \boxed{a_2}]_a \text{ then } u_1 \text{ else } v_1 \sim \text{if } \boxed{c_1} \boxed{c_2}]_c \text{ then } s_1 \text{ else } t_1} CS_{\square}^b}{\frac{\text{if } \boxed{a} \boxed{a}]_a \text{ then } u \text{ else } v \sim \text{if } \boxed{c} \boxed{c}]_c \text{ then } s \text{ else } t}{\text{if } a \text{ then } u \text{ else } v \sim \text{if } c \text{ then } s \text{ else } t} R} 2Box^s$$

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

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

We need the functions in  $\mathcal{B}$  to block the if-homomorphism to ensure that for all  $\boxed{a} \boxed{c}]_b \in \text{st}(t)$ ,  $\llbracket a \rrbracket = \llbracket c \rrbracket = \llbracket b \rrbracket$ . Therefore the set of equalities  $R_2$  is *not* extended to  $\mathcal{B}$ . For example we have:

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

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

*Definition 15.* We let  $R_{\square}$  be the following axiom schema:

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

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  $CS_{\square}$  rule.

*Example 11.* We give two examples of valid application of the  $R_{\square}$  rules. The first  $R_{\square}$  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  $R_{\square}$  application is valid because the introduced boxed condition already appears in the conclusion:

$$\frac{\text{if } \text{eq}(g(\{0\}_{pk}^n), \{0\}_{pk}^n) \text{ then } \text{dec}(g(\{0\}_{pk}^n), \text{sk}) \sim v \quad \text{else } \text{dec}(g(\{0\}_{pk}^n), \text{sk})}{\text{dec}(g(\{0\}_{pk}^n), \text{sk}) \sim \text{if } \boxed{a \mid c}_b \text{ then } v \text{ else } v} R_{\square} \quad \frac{\text{if } \boxed{a \mid c}_b \text{ then } (\text{if } \boxed{a \mid c}_b \text{ then } u \text{ else } w) \sim t \quad \text{else } v}{\text{if } \boxed{a \mid c}_b \text{ then } u \text{ else } v \sim t} R_{\square} \diamond$$

When boxing a condition  $c$ , we want the term  $c_{\square}$  indexing the box  $\boxed{c \mid c}_{c_{\square}}$  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  $2\text{erase}$  be the function defined on if-free ground terms by:

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

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

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

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

*Definition 17.* We let 2Box and  $CS_{\square}$  be the axioms:

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

$$\frac{\vec{w}, a_1, (u_i)_i \sim \vec{w}', a'_1, (u'_i)_i \quad \vec{w}, a_2, (v_i)_i \sim \vec{w}', a'_2, (v'_i)_i}{\vec{w}, (\text{if } \boxed{a_1 \mid a_2}_a \text{ then } u_i \text{ else } v_i)_i \sim \vec{w}', (\text{if } \boxed{a'_1 \mid a'_2}_{a'} \text{ then } u'_i \text{ else } v'_i)_i} CS_{\square} \quad \text{when } a, a' \in \mathcal{T}(\mathcal{F}_{\text{if}}, \mathcal{N})$$

Remark that for the  $CS_{\square}$  rule to be sound we need  $\llbracket a_1 \rrbracket$ ,  $\llbracket a_2 \rrbracket$  and  $\llbracket a \rrbracket$  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  $\boxed{a \mid c}_b \in \text{st}(t)$ ,  $a =_R c =_R b$ . We lift this to formulas as expected.

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

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

Besides,  $R_{\square}$ ,  $CS_{\square}$  and 2Box are sound on well-formed formulas.

**PROOF.** The only rule not obviously preserving well-formedness is  $R_{\square}$ , but its side-conditions guarantee the well-formedness invariant. The only rule that is not always sound is  $CS_{\square}$ , and it is trivially sound on well-formed formulas.  $\square$

*Remark 4.* We extend  $\text{cond-st}$  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:

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

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

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

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

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

This allows to have  $R_{\square}$  rules only at the beginning of the proof.

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

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

*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:

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

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

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

Finally we commute the  $CS_{\square}$  applications to the right. □

## 6.2 The Freeze Strategy

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

*Example 13.* Let  $a_i \equiv \text{if } b_i \text{ then } c_i \text{ else } d_i$  ( $i \in \{1, 2\}$ ), we want to forbid the following partial derivation to appear:

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

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

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

We replace, in the premise,  $a_1$  by  $\bar{a}_1$  in  $b_1$  and  $a_2$  by  $\bar{a}_2$  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.

- $FA_S \cdot FA(b, b') \Rightarrow R \cdot FA(b, b') \cdot FA_S^* \cdot Dup$

$$\frac{\frac{\frac{\frac{\bar{u}, \bar{v}, b, s, t \sim \bar{u}', \bar{v}', b', s', t'}{FA(b, b')}}{\bar{u}, \bar{v}, \text{if } b \text{ then } s \text{ else } t}}{\sim \bar{u}', \bar{v}', \text{if } b' \text{ then } s' \text{ else } t'}}{FA_f} \Rightarrow \frac{\frac{\frac{\frac{\bar{u}, b, s, \bar{v}, t \sim \bar{u}', b', s', \bar{v}', t'}{Dup}}{\bar{u}, b, \bar{v}, s, \bar{v}, t \sim \bar{u}', b', \bar{v}', s', \bar{v}', t'}}{FA_f^{(2)}}}{\bar{u}, \text{if } b \text{ then } f(\bar{v}, s) \text{ else } f(\bar{v}, t)}}{\sim \bar{u}', \text{if } b' \text{ then } f(\bar{v}', s') \text{ else } f(\bar{v}', t')}}{R}}{\bar{u}, f(\bar{v}, \text{if } b \text{ then } s \text{ else } t)} \sim \bar{u}', f(\bar{v}', \text{if } b' \text{ then } s' \text{ else } t')}$$

- $CS_{\square} \cdot R_{\square} \Rightarrow R_{\square} \cdot CS_{\square}$ :

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

can be rewritten into:

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

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

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

Then the following proof:

$$\frac{\frac{\frac{(w_j[b_{\square}])_j, a_1[b_{\square}], (u_i[b_{\square}])_i \sim (w'_j[b'_{\square}])_j, a'_1[b'_{\square}], (u'_i[b'_{\square}])_i}{2Box} \quad \frac{(w_j[b])_j, a_2[b], (v_i[b])_i}{\sim (w'_j[b'])_j, a'_2[b'], (v'_i[b'])_i}}{(w_j[b])_j, a_1[b], (u_i[b])_i \sim (w'_j[b'])_j, a'_1[b'], (u'_i[b'])_i} CS_{\square}}{\frac{(w_j[b])_j, (\text{if } \boxed{a_1[b] \mid a_2[b]}_a \text{ then } u_i[b] \text{ else } v_i[b])_i}{\sim (w'_j[b'])_j, (\text{if } \boxed{a'_1[b'] \mid a'_2[b']}_{a'} \text{ then } u'_i[b'] \text{ else } v'_i[b'])_i}}$$

can be rewritten into:

$$\frac{\frac{\frac{(w_j[b_{\square}])_j, a_1[b_{\square}], (u_i[b_{\square}])_i}{\sim (w'_j[b'_{\square}])_j, a'_1[b'_{\square}], (u'_i[b'_{\square}])_i} \quad \frac{(w_j[b])_j, a_2[b], (v_i[b])_i}{\sim (w'_j[b'])_j, a'_2[b'], (v'_i[b'])_i}}{(\text{if } \boxed{a_1[b_{\square}] \mid a_2[b]}_a \text{ then } w_j[b_{\square}] \text{ else } w_j[b])_j, (\text{if } \boxed{a_1[b_{\square}] \mid a_2[b]}_a \text{ then } u_i[b_{\square}] \text{ else } v_i[b])_i} CS_{\square}}{\sim (\text{if } \boxed{a'_1[b'_{\square}] \mid a'_2[b']}_{a'} \text{ then } w'_j[b'_{\square}] \text{ else } w'_j[b'])_j, (\text{if } \boxed{a'_1[b'_{\square}] \mid a'_2[b']}_{a'} \text{ then } u'_i[b'_{\square}] \text{ else } v'_i[b'])_i} 2Box}}{\frac{(\text{if } \boxed{a_1[b] \mid a_2[b]}_a \text{ then } w_j[b] \text{ else } w_j[b])_j, (\text{if } \boxed{a_1[b] \mid a_2[b]}_a \text{ then } u_i[b] \text{ else } v_i[b])_i}{\sim (\text{if } \boxed{a'_1[b'] \mid a'_2[b']}_{a'} \text{ then } w'_j[b'] \text{ else } w'_j[b'])_j, (\text{if } \boxed{a'_1[b'] \mid a'_2[b']}_{a'} \text{ then } u'_i[b'] \text{ else } v'_i[b'])_i} R_{\square}}$$

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



*Definition 19.* Let  $\bar{\phantom{x}}$  be a new function symbol of arity one. For every ground term  $s$ , we let  $\tilde{s}$  be:

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

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

*Definition 20.* We let BFA be the rule:

$$\frac{\tilde{w}_1, \tilde{b}_1, u_1, v_1 \sim \tilde{w}_2, \tilde{b}_2, u_2, v_2}{\tilde{w}_1, \text{if } b_1 \text{ then } u_1 \text{ else } v_1 \sim \tilde{w}_2, \text{if } b_2 \text{ then } u_2 \text{ else } v_2} \text{BFA}(b_1, b_2)$$

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

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

$$\frac{\begin{array}{c} \bar{b}_0 \\ / \quad \backslash \\ a_0 \quad b_1 \\ / \quad \backslash \\ a_1 \quad a_2 \end{array}, s, t \sim \bar{b}', s', t'}{\begin{array}{c} \left( \begin{array}{c} \bar{b}_0 \\ / \quad \backslash \\ a_0 \quad b_1 \\ / \quad \backslash \\ a_1 \quad a_2 \end{array} \right) \sim \begin{array}{c} b' \\ / \quad \backslash \\ s' \quad t' \end{array} \\ / \quad \backslash \\ s \quad t \end{array}} \text{BFA} \quad \text{and} \quad \frac{\begin{array}{c} b_0 \\ / \quad \backslash \\ a_0 \quad b_1 \\ / \quad \backslash \\ a_1 \quad a_2 \end{array}, s, t \sim b', s', t'}{\begin{array}{c} \bar{b}_0 \\ / \quad \backslash \\ a_0 \quad b_1 \\ / \quad \backslash \\ a_1 \quad a_2 \end{array}, s, t \sim \bar{b}', s', t'} \text{UnF} \quad \diamond$$

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

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

$$\tilde{\mathfrak{F}}((\text{CS}_{\square} + 2\text{Box} + \text{FA} + \text{Dup} + U + \text{Restr})^*)$$

then there exists  $P'$  such that  $P' \vdash \bar{u} \sim \bar{v}$  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  $\tilde{\mathfrak{F}}(\mathcal{L})$  where  $\mathcal{L}$  is closed by sub-words then  $P'$  is in  $\tilde{\mathfrak{F}}(\mathcal{L})$ .

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

We can state the following ordered strategy lemma:

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

$$\tilde{\mathfrak{F}}((2\text{Box} + R_{\square})^* \cdot \text{CS}_{\square}^* \cdot \{\overline{\text{BFA}}(b, b')\}^* \cdot \text{UnF} \cdot \text{FA}_s^* \cdot \text{Dup}^* \cdot U)$$

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

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  $\bar{\cdot}$ . We then order these multi-set using the multi-set ordering  $>_{\text{mult}}$ , which is induced by the product ordering  $>_{\times}$ , which itself is built upon an arbitrary total rewrite ordering on ground terms without boxes  $>$  (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:

$$\frac{\frac{\frac{b, a, c, u_1, v_1 \sim b', a', c', u_2, v_2}{\bar{b}, a, c, u_1, v_1 \sim \bar{b}', a', c', u_2, v_2} \text{ UnF}}{\tilde{b}_1, u_1, v_1 \sim \tilde{b}_2, u_2, v_2} \text{ BFA}(\bar{b}, \bar{b}')}{\text{if } b_1 \text{ then } u_1 \text{ else } v_1 \sim \text{if } b_2 \text{ then } u_2 \text{ else } v_2} \text{ BFA}(b_1, b_2)$$

And  $P_2$  be the derivation:

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

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:

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

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  $(b_1, b_2) >_{\times} (b, b')$ ,  $(b_1, b_2) >_{\times} (a, a')$  and  $(b_1, b_2) >_{\times} (c, c')$ . Hence:

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

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  $P_1$ .  $\diamond$

**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:

$$\frac{\tilde{w}_1, \tilde{b}_1, u'_1, v'_1 \sim \tilde{w}_2, \tilde{b}_2, u'_2, v'_2}{\tilde{w}_1, \text{if } b_1 \text{ then } u_1 \text{ else } v_1 \sim \tilde{w}_2, \text{if } b_2 \text{ then } u_2 \text{ else } v_2} \text{ BFA}(b_1, b_2)$$

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  $>_{\text{mult}}$ . We consider the first conditions  $(b_1, b_2)$  (starting from the bottom) which are deconstructed. We let  $b_1 \equiv \text{if } b \text{ then } a \text{ else } c$  and  $b_2 \equiv \text{if } b' \text{ then } a' \text{ else } c'$ , we know that our proof has the following

shape:

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

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  $P_2$ :

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

One can check that  $A_1$  remains the same in the second proof tree since the hole in  $C$  is not in a 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:

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

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.  $\square$

## 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  $(2\text{Box} + R_{\square})^*$  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  $\text{CCA}_2$ , to get some further restrictions.

When applying the atomic axioms  $\text{CCA}_2$ , we would like to require that terms are in  $R$ -normal form, e.g. to avoid the application of  $\text{CCA}_2$  to terms with an unbounded component, such as  $\pi_1(\langle u, v \rangle)$ . Unfortunately, the side-conditions of  $\text{CCA}_2$  are not stable by  $R$ . E.g., consider the  $\text{CCA}_2$  instance:

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

The  $R$ -normal form of the left term is:

$$\text{if eq}(g(n_u), n_u) \text{ then } \{A\}_{\text{pk}(n)}^{n_r} \text{ else } \{B\}_{\text{pk}(n)}^{n_r}$$

which cannot be used in a valid  $\text{CCA}_2$  instance, since the condition  $\text{eq}(g(n_u), n_u)$  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  $\text{CCA}_2$  axiom, as well as its side-conditions. In other word, if  $\vec{u} \sim \vec{v}$  is a valid  $\text{CCA}_2$  instance then its normalization  $\vec{u}_n \sim \vec{v}_n$  is also a valid  $\text{CCA}_2$  instance. We illustrate this on an example. The term:

$$\{\text{if } ( \text{if } b \text{ then } a \text{ else } c ) \text{ then } \{\text{if } d \text{ then } u \text{ else } v\}_{\text{pk}}^{n_1} \text{ else } w\}_{\text{pk}}^{n_2}$$

is normalized as follows:

$$\left\{ \begin{array}{l} \text{if } b \text{ then if } a \text{ then } \{\text{if } d \text{ then } u \text{ else } v\}_{\text{pk}}^{n_1} \text{ else } w \\ \text{else if } c \text{ then } \{\text{if } d \text{ then } u \text{ else } v\}_{\text{pk}}^{n_1} \text{ else } w \end{array} \right\}_{\text{pk}}^{n_2} \quad (6)$$

Observe that  $\text{CCA}_2$  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  $\{\text{if } d \text{ then } u \text{ else } v\}_{\text{pk}}^{n_1}$ .

*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  $\mathcal{A}_>$  be the ordered strategy from Lemma 8, and we define several restriction of  $\mathcal{A}_>$ :

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

The rule  $\text{CS}_\square$  is the only branching rule, therefore, after applying all the  $\text{CS}_\square$  rules, we can associate to each branch  $l$  of the proof a left  $\text{CCA}_2$  trace  $\mathcal{S}_l = (\mathcal{K}_l, \mathcal{R}_l, \mathcal{E}_l, \mathcal{D}_l)$  of the  $\text{CCA}_2$  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  $\text{CCA}_2$  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  $\mathcal{F}_{\text{if}, \mathbf{0}}$  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  $\mathcal{F}_{\text{if}, \mathbf{0}}$  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  $\text{CCA}_2$  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  $\text{fresh}(\vec{u}; \vec{v})$  holds if and only if no term in  $\vec{v}$  is a subterm of a term in  $\vec{u}$ , i.e.:

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

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

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

*Definition 23.* A  $\mathcal{S}_I$ -basic term is a term of the form  $U[\vec{w}, (\alpha_j)_j, (\text{dec}_k)_k]$  where:

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

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

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

*Definition 24.* A  $\mathcal{S}_I$ -basic term is *normalized* if it is of the form  $U[\vec{w}, (\alpha_j)_j, (\text{dec}_k)_k]$  where:

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

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

*Eager Reduction.* We state here an crucial property of the  $\mathcal{A}_{\text{FA}_s} = \mathfrak{F}(\text{FA}_s^* \cdot \text{Dup}^* \cdot \text{CCA}_2)$  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:

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

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

**LEMMA 9 (INFORMAL).** *Let  $P \vdash_{\mathcal{A}_{\text{FA}_s}} \beta \sim \beta'$  where  $\beta$  and  $\beta'$  are basic terms. Then there exist  $\gamma =_R \beta$  and  $\gamma' =_R \beta'$  such that:*

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

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

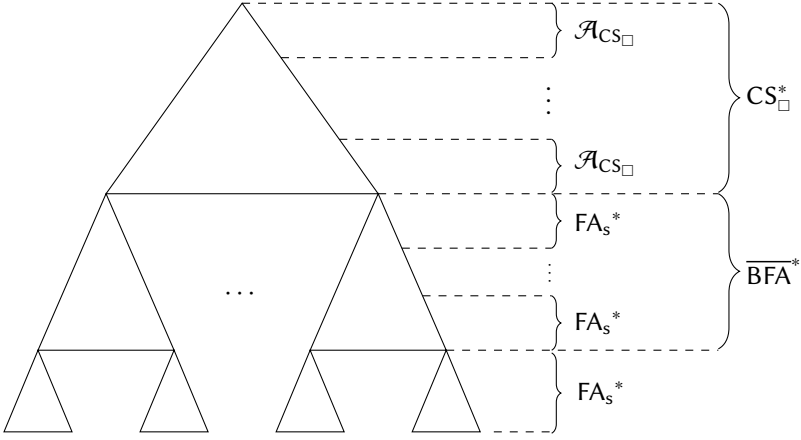


Fig. 10. The Shape of the Term is Determined by the Proof.

*Normalized Proof Form.* Every application of  $CS_{\square}$ :

$$\frac{a_1, u \sim b_1, s \quad a_2, v \sim b_2, t}{\text{if } \boxed{a_1 \mid a_2}_a \text{ then } u \text{ else } v \sim \text{if } \boxed{b_1 \mid b_2}_b \text{ then } s \text{ else } t} CS_{\square}$$

is such that if we extract the sub-proof of  $a_i \sim b_i$  (for  $i \in \{1, 2\}$ ), we get a proof in  $\mathcal{A}_{CS_{\square}}$ . Therefore, we can check that terms after  $(2\text{Box} + R_{\square})^*$  are of the form informally described in Figure 10. We define a normal form for such proofs, called *normalized proof form*, and we define  $\vdash^{\text{npf}}$  by  $P \vdash^{\text{npf}} t \sim t'$  if and only if  $P \vdash t \sim t'$ , the proof  $P$  is in  $\mathcal{A}_{>}$  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  $\mathfrak{F}(FA_s^* \cdot \text{Dup}^* \cdot \cup)$  then  $s$  is a  $S_l$ -normalized basic term.

LEMMA 10. *Every formula in  $\mathfrak{F}((CS + FA + R + \text{Dup} + cCA_2)^*)$  is provable using the strategy  $\vdash^{\text{npf}}$ .*

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 yields a proof  $P$  where every term  $s$  appearing at the conclusion of a sub-proof of  $P$  in the fragment  $\mathcal{A}_{FA_s}$  is a  $S_l$ -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  $\mathcal{A}_{FA_s}$  is a  $S_l$ -normalized basic term.  $\square$

## 7.2 Key Properties

*Characterization of Basic Terms.* We give a key characterization proposition for basic terms: if two  $S_l$ -normalized basic terms  $\beta$  and  $\beta'$  are such that, when  $R$ -normalizing them, they share a leaf term, then they are identical.

PROPOSITION 8. *For all  $S_l$ -normalized basic terms  $\beta, \beta'$ , if we have:*

$$\text{leave-st}(\beta \downarrow_R) \cap \text{leave-st}(\beta' \downarrow_R) \neq \emptyset$$

*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  $\beta \equiv U[\vec{w}, (\alpha_j)_j, (\text{dec}_k)_k]$ ,  $\beta' \equiv U'[\vec{w}', (\alpha'_j)_j, (\text{dec}'_k)_k]$  and:

$$U[\vec{w}, (\{\llbracket j \rrbracket\}_{pk_j}^n)_j, (\text{dec}(\llbracket k, sk_k \rrbracket))_k]$$

$$U'[\vec{w}', (\{\llbracket j \rrbracket\}_{pk'_j}^n)_j, (\text{dec}(\llbracket k, sk'_k \rrbracket))_k]$$

are in  $R$ -normal form. Using the fact that  $U, U', \vec{w}, \vec{w}'$  are if-free, and the hypothesis that  $\beta$  and  $\beta'$  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  $\beta$  and  $\beta'$ ). 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  $\alpha_j \equiv \{m\}_{pk}^n$ , then we know that  $n$  appears in  $\vec{w}$ , which is not possible since, as  $\beta$  is a  $\mathcal{S}_l$ -normalized basic term,  $n \in \mathcal{R}_l$  does not appear in  $\vec{w}$ .
- If  $\beta_{|p}$  is in some decryption  $\text{dec}_k \equiv \text{dec}(u_k, sk_k)$  then, similarly to the previous case, we have  $sk_k$  appearing in  $\vec{w}$ , which contradicts the fact that  $sk_k \in \mathcal{K}_l$  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  $(\text{dec}_k)_k$  and  $(\text{dec}'_k)_k$  are identical. The former follows from the fact that, for a given encryption randomness  $n \in \mathcal{R}_l$ , there exists a unique  $m$  such  $\{m\}^n \in \mathcal{E}_l$ ; 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  $\text{CCA}_2$  side-conditions).  $\square$

*Proofs of  $b \sim \text{false}$  or  $\text{true}$ .* Using the previous proposition, we can show that for all  $b$ , if  $b$  is if-free then there is no derivation of  $b \sim \text{true}$  or  $b \sim \text{false}$  in  $\mathcal{A}_>$ . Such derivations would be problematic since  $\text{true}$  and  $\text{false}$  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  $b \neq \text{false}$  (resp.  $b \neq \text{true}$ ). Then there exists no derivation of  $b \sim \text{false}$  (resp.  $b \sim \text{true}$ ) in  $\mathcal{A}_>$ .*

PROOF. This is shown by induction on the size of the derivation. The full proof is in Appendix G, and relies on Proposition 8.  $\square$

## 8 BOUNDING THE PROOF AND DECISION PROCEDURE

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

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

$$\frac{a_1, a_2, u_3, v_4, w_5 \sim b_1, c_2, r_3, s_4, t_5}{\begin{array}{c} a_1 \\ \swarrow \quad \searrow \\ a_2 \quad w_5 \\ \swarrow \quad \searrow \\ u_3 \quad v_4 \\ \underbrace{\hspace{2cm}} \\ \sigma \end{array} \sim \begin{array}{c} b_1 \\ \swarrow \quad \searrow \\ c_2 \quad t_5 \\ \swarrow \quad \searrow \\ r_3 \quad s_4 \\ \underbrace{\hspace{2cm}} \\ \tau \end{array}}{\overline{\text{BFA}}^{(2)}} \quad (7)$$

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

$$\frac{\frac{\frac{a_1, u_3, w_5 \sim b_1, r_3, t_5}{\text{BFA}}}{\begin{array}{c} a_1 \\ / \quad \backslash \\ u_3 \quad w_5 \end{array}} \sim \begin{array}{c} b_1 \\ / \quad \backslash \\ r_3 \quad t_5 \end{array}}{\sigma \sim \tau} R$$

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

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

$$a_1^\square \equiv \boxed{a_1^l} \boxed{a_1^r} \bigg|_a \quad (i \in \{1, 2\}) \quad b_1^\square \equiv \boxed{b_1^l} \boxed{b_1^r} \bigg|_b \quad c_2^\square \equiv \boxed{c_2^l} \boxed{c_2^r} \bigg|_c$$

Consider the following cut:

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

As we did for  $\overline{\text{BFA}}$ , 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  $\mathcal{A}_{\text{CS}_\square}$ , which we recall is the fragment:

$$\mathfrak{F}(\text{CS}_\square^* \cdot \{\overline{\text{BFA}}(b, b')\}^* \cdot \text{UnF} \cdot \text{FA}_s^* \cdot \text{Dup}^* \cdot \text{CCA}_2)$$

Therefore we cannot apply Lemma 2. To deal with this cut, we generalize Lemma 2 to the case where the proof is in  $\mathcal{A}_{\text{CS}_\square}$ . 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  $\text{CS}_\square$ .

**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  $P \vdash^{npl} a, a' \sim b, c$ , 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.  $\square$

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  $b_1^\square =_{R_\square} b =_{R_\square} c_2^\square$  (using well-formedness). Similarly  $a_1^\square =_{R_\square} a =_{R_\square} a_2^\square$ . This yields the (cut-free) proof:

$$\frac{\begin{array}{c} \vdots (A') \\ a_1^l, u_3 \sim b_1^l, r_3 \end{array} \quad \begin{array}{c} \vdots (C) \\ a_1^r, w_5 \sim b_1^r, t_5 \end{array}}{\text{CS}_\square} \frac{\begin{array}{c} a_1^\square \\ / \quad \backslash \\ u_3 \quad w_5 \end{array} \sim \begin{array}{c} b_1^\square \\ / \quad \backslash \\ r_3 \quad t_5 \end{array}}{\sigma \sim \tau} R_\square$$

where (A') is extracted from (A) by Lemma 7. Finally, to get a proof in  $\mathcal{A}_>$ , we commute the  $R_\square$  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  $\text{eq}(n_0, n_1)$  is spurious in:

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

We say that a basic condition  $\beta$ , 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  $\alpha$ -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  $\alpha$ -bounded condition of  $t \sim t'$  (indeed, we cannot remove a decryption's guards, as this would not yield a valid  $\text{CCA}_2$  instance).

We let  $\vdash_\alpha^{\text{npf}}$  be the restriction of  $\vdash^{\text{npf}}$  to proofs such that all basic conditions appearing in the derivation are  $\alpha$ -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.  $\vdash_\alpha^{\text{npf}}$  is complete with respect to  $\vdash^{\text{npf}}$ .

*Bounding  $\alpha$ -bounded Basic Conditions.* Finally, it remains to bound the size of  $\alpha$ -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  $\beta_1 <_{\text{st}} \dots <_{\text{st}} \beta_n$ , (where  $u <_{\text{st}} v$  iff  $u \not\equiv v$  and  $u \in \text{st}(v)$ ), we show that we can associate to each  $\beta_i$  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,  $\lambda_i$  is obtained from  $\beta_i$  by “flattening” it: we remove all decryption guards, and replace the content of every encryption  $\{m\}_{\text{pk}}^n$  by a term  $\{\tilde{m}\}_{\text{pk}}^n$ , where  $\tilde{m}$  is if-free and in  $\mathcal{B}(t, t')$ . Moreover, we show that, for every  $\mathcal{S}_I$ -normalized basic terms  $\beta, \gamma$  and their associated frame terms  $\lambda, \mu$ , if  $\lambda \equiv \mu$  then  $\beta \equiv \gamma$  (this result is similar to Proposition 8).

Since the  $\beta_i$ s are all pair-wise distinct (as  $<_{\text{st}}$  is strict), and since for every  $i$ , the frame term  $\lambda_i$  uniquely characterizes  $\beta_i$ , we know that the  $\lambda_i$ s 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  $\alpha$ -bounded basic terms that can appear in a proof  $P$  of  $P \vdash_\alpha^{\text{npf}} t \sim t'$  (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  $Ax \wedge \vec{u} \approx \vec{v}$  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  $\text{CCA}_2$  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  $\theta : \mathcal{N} \rightarrow \mathcal{N}$  from names to names such that  $\vec{u} \equiv \vec{v}\theta$ . Then,  $\text{Refl}_\alpha$  is the axiom schema:

$$\overline{\vec{u} \sim \vec{v}} \text{ Refl}_\alpha \quad \text{when } \vec{u} \equiv_\alpha \vec{v}\theta$$

We let  $\text{Ax}_{\text{pure}}$  be the set of axioms obtained from  $\text{Ax}$  by removing the  $\text{CCA}_2$  axiom schema, and adding the  $\text{Refl}_\alpha$  axiom schema, for reflexivity module alpha-renaming of names. The following lemma gives a necessary condition on  $t$  and  $t'$  for the formula  $t \sim t'$  to be derivable using  $\text{Ax}_{\text{pure}}$ . 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  $\text{Ax}_{\text{pure}}$  of  $t \sim t'$ , then:*

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

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:

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

Moreover, the names of  $s_i$  and  $s'_i$  all appear in  $\text{st}(t) \cup \text{st}(t')$ .

- there exists a derivation of  $u \sim u'$  in  $\mathfrak{F}(\text{CS}^* \cdot \{BFA(b, b')\}^* \cdot \text{FA}_s^* \cdot \text{Dup}^* \cdot \text{Refl}_\alpha)$ .
- No condition appears twice on the same branch of  $u$  (resp.  $u'$ ).

PROOF (SKETCH). First, we note that since the  $\text{CCA}_2$  axiom schema already allow for reflexivity module alpha-renaming of names, we have  $\text{Ax}_{\text{pure}} \subseteq \text{Ax}$ . 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 module alpha-renaming, to deduce the shape of  $t$  and  $t'$ . The details are given in Appendix H.  $\square$

Example 16. let  $g() \in \mathcal{G}$  and  $n, n', n_1, n_2 \in \mathcal{N}$ . Consider the goal  $t \sim t'$  where:

$$\begin{aligned} t &\equiv \text{if } g(n) \text{ then } \langle n, n_1 \rangle & \text{and } t' &\equiv \langle n, n' \rangle \\ &\quad \text{else } \langle n, n_2 \rangle \end{aligned}$$

Then we have the derivation:

$$\frac{\frac{\text{where } \theta = \{n' \mapsto n_1\}}{g(n), \langle n, n_1 \rangle \sim g(n), \langle n, n' \rangle} \text{ Refl}_\alpha \quad \frac{\text{where } \theta = \{n' \mapsto n_2\}}{g(n), \langle n, n_2 \rangle \sim g(n), \langle n, n' \rangle} \text{ Refl}_\alpha}{\text{if } g(n) \text{ then } \langle n, n_1 \rangle \quad \text{if } g(n) \text{ then } \langle n, n' \rangle} \text{ CS} \\ \frac{\text{else } \langle n, n_2 \rangle \quad \text{else } \langle n, n' \rangle}{t \sim t'} R$$

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:

$$\begin{aligned} C[x \diamond (y, z)] &\equiv \text{if } x \text{ then } y \text{ else } z & s_1 &\equiv g(n) \equiv s'_1 \\ s'_2 &\equiv \langle n, n' \rangle \text{ and } s_2 &\equiv s'_2 \{n' \mapsto n_1\} & s'_3 &\equiv \langle n, n' \rangle \text{ and } s_3 &\equiv s'_3 \{n' \mapsto n_2\} \end{aligned}$$

Remark that we have a different substitution in each branch.  $\diamond$

*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 BFA( $b, b'$ ) application by a CS application. Moreover, we can assume that all  $\text{Refl}_\alpha$  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:

$$\frac{}{\overline{\vec{u}, f(\vec{v}) \sim \vec{u}', f(\vec{v}')}} \text{Refl}_\alpha \implies \frac{\overline{\vec{u}, \vec{v} \sim \vec{u}', \vec{v}'}}{\overline{\vec{u}, f(\vec{v}) \sim \vec{u}', f(\vec{v}')}} \text{Refl}_\alpha \text{FA}_s$$

$$\frac{\overline{\vec{u}, v \sim \vec{u}', v'}}{\overline{\vec{u}, v, v \sim \vec{u}', v', v'}} \text{Dup} \implies \frac{}{\overline{\vec{u}, v, v \sim \vec{u}', v', v'}} \text{Refl}_\alpha$$

Therefore, we can assume that  $P$  is in  $\mathfrak{F}(R \cdot \text{CS}^* \cdot \text{FA}_s^* \cdot \text{Refl}_\alpha)$ , 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  $1 \leq i \leq m$ ,  $\sigma_i$  be a term in  $\{s_i; s'_i\} \cap (\text{st}(t \downarrow_R) \cup \text{st}(t' \downarrow_R))$  (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  $\text{st}(t \downarrow_R) \cup \text{st}(t' \downarrow_R)$ , it follows that that each branch of  $w$  contains at most  $2 \cdot N$  conditions, where  $N$  is the cardinal of  $\text{st}(t \downarrow_R) \cup \text{st}(t' \downarrow_R)$ . We deduce that  $w$  is a binary tree of depth at most  $2 \cdot N + 1$  ( $2 \cdot N$  different conditions plus the leaf), labeled by terms in  $\text{st}(t \downarrow_R) \cup \text{st}(t' \downarrow_R)$ . 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  $\text{Refl}_\alpha$ . To guess an application of  $\text{Refl}_\alpha$ , 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  $\text{Refl}_\alpha$  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  $\text{Ax}_{\text{pure}}$ :

- Guess the term  $w$ , by non-deterministically guessing a tree of depth at most  $2 \cdot N + 1$ , and a labeling of the tree using  $\text{st}(t \downarrow_R) \cup \text{st}(t' \downarrow_R)$ . 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  $t \equiv_R u$ . This can be done in polynomial time in  $|u| + |t \downarrow_R|$ .<sup>9</sup> Similarly, we check that  $t' \equiv_R u'$ . If both checks succeed, we know that  $t \sim t'$  can be derived using  $\text{Ax}_{\text{pure}}$ .

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

<sup>9</sup>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  $3\text{-NEXP TIME}$ , 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  $\text{CCA}_2$  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  $\text{IND-CCA}_2$  cryptographic assumption (in particular the characterization of Section 7). Consequently, extending our decidability result to other cryptographic assumptions (e.g.  $\text{EUF-CMA}$ ) 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. Héraud, 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://prosecco.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 Chréten, 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 Semantics (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  $\text{cca}_2$ , 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  $>_u$ , the term rewriting system  $\rightarrow_{R>_u}$  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  $\text{eq}(\_, \_)$ . The critical pairs involving  $\text{dec}(\_, \_)$  are similar to the critical pairs involving  $\pi_1(\_)$ , and the critical pairs for  $\mathbf{0}(\_)$  are trivial.

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

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

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

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

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

- **Critical Pairs  $R_2/R_3$ :**

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

→ if  $b$  then (if  $b$  then  $f(u, s)$  else  $f(v, s)$ ) else  $f(w, s)$   
 → if  $b$  then  $f(u, s)$  else  $f(w, s)$

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

if  $d$  then (if  $b$  then (if  $a$  then  $u$  else  $v$ ) else  $w$ ) else (if  $c$  then (if  $a$  then  $u$  else  $v$ ) else  $w$ ) ←\*

if  $a$  then if  $d$  then (if  $b$  then  $u$  else  $w$ ) else (if  $c$  then  $u$  else  $w$ ) ←<sup>2</sup>  
 else if  $d$  then (if  $b$  then  $v$  else  $w$ ) else (if  $c$  then  $v$  else  $w$ )

if  $a$  then (if (if  $d$  then  $b$  else  $c$ ) then  $u$  else  $w$ ) else (if (if  $d$  then  $b$  else  $c$ ) then  $v$  else  $w$ ) ←  
if (if  $d$  then  $b$  else  $c$ ) then (if  $a$  then  $u$  else  $v$ ) else  $w$

→ if  $d$  then (if  $b$  then (if  $a$  then  $u$  else  $v$ ) else  $w$ ) else (if  $c$  then (if  $a$  then  $u$  else  $v$ ) else  $w$ )

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

$u \leftarrow$  if true then  $u$  else  $u$  →  $u$

$u \leftarrow$  if true then  $u$  else  $v \leftarrow$  if true then (if true then  $u$  else  $v$ ) else  $w$   
 → if true then  $u$  else  $w \rightarrow u$

if  $b$  then  $u$  else  $v \leftarrow$  if  $b$  then (if  $b$  then  $u$  else  $v$ ) else (if  $b$  then  $u$  else  $v$ )  
 → if  $b$  then  $u$  else (if  $b$  then  $u$  else  $v$ ) → if  $b$  then  $u$  else  $v$

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

if  $a$  then  $u$  else  $v \leftarrow$

if  $b$  then (if  $a$  then  $u$  else  $v$ ) else (if  $a$  then  $u$  else  $v$ )

→ if  $a$  then (if  $b$  then  $u$  else (if  $a$  then  $u$  else  $v$ )) else (if  $b$  then  $v$  else (if  $a$  then  $u$  else  $v$ ))

→<sup>2</sup> if  $a$  then if  $a$  then (if  $b$  then  $u$  else  $u$ ) else (if  $b$  then  $u$  else  $v$ )  
 else if  $a$  then (if  $b$  then  $v$  else  $u$ ) else (if  $b$  then  $v$  else  $v$ )

→<sup>2</sup> if  $a$  then (if  $b$  then  $u$  else  $u$ ) else (if  $b$  then  $v$  else  $v$ )

→<sup>2</sup> if  $a$  then  $u$  else  $v$

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

if  $c$  then if  $b$  then (if  $a$  then  $u$  else  $s$ ) else (if  $a$  then  $v$  else  $s$ ) ←<sup>2</sup>  
 else if  $b$  then (if  $a$  then  $u$  else  $t$ ) else (if  $a$  then  $v$  else  $t$ )

if  $c$  then (if  $a$  then (if  $b$  then  $u$  else  $v$ ) else  $s$ ) else (if  $a$  then (if  $b$  then  $v$  else  $u$ ) else  $t$ ) ←  
if  $a$  then (if  $b$  then  $u$  else  $v$ ) else (if  $c$  then  $s$  else  $t$ )

→ if  $b$  then (if  $a$  then  $u$  else (if  $c$  then  $s$  else  $t$ )) else (if  $a$  then  $v$  else (if  $c$  then  $s$  else  $t$ ))

→<sup>2</sup> if  $b$  then if  $c$  then (if  $a$  then  $u$  else  $s$ ) else (if  $a$  then  $u$  else  $t$ )  
 else if  $c$  then (if  $a$  then  $v$  else  $s$ ) else (if  $a$  then  $v$  else  $t$ )

→\* if  $c$  then if  $b$  then (if  $a$  then  $u$  else  $s$ ) else (if  $a$  then  $v$  else  $s$ )  
 else if  $b$  then (if  $a$  then  $u$  else  $t$ ) else (if  $a$  then  $v$  else  $t$ )

## B THE $\text{CCA}_2$ AXIOMS

We define and prove correct a recursive set of axioms for an  $\text{IND-CCA}_2$  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  $\text{IND-CCA}_2$  Game.* Consider the following multi-users  $\text{IND-CCA}_2$  game: the adversary receives  $n$  public-keys. For each key  $\text{pk}_i$ , he has access to a left-right oracle  $\mathcal{O}_{\text{LR}}(\text{pk}_i, b)$  that takes two messages  $m_0, m_1$  as input and returns  $\{m_b\}_{\text{pk}_i}^{n_r}$ , where  $b$  is an internal random bit uniformly drawn at the beginning by the challenger (the same  $b$  is used for all left-right oracles) and  $n_r$  is a fresh nonce. Moreover, for all key pairs  $(\text{pk}_i, \text{sk}_i)$ , the adversary has access to an  $\text{sk}_i$  decryption oracle  $\mathcal{O}_{\text{dec}}(\text{sk}_i)$ , but cannot call  $\mathcal{O}_{\text{dec}}(\text{sk}_i)$  on a cipher-text returned by  $\mathcal{O}_{\text{LR}}(\text{pk}_i, b)$  (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  $\text{IND-CCA}_2$  security are defined as usual.

It is known that if an encryption scheme is  $\text{IND-CCA}_2$  then it is also multi-users  $\text{IND-CCA}_2$  (see [12]). Therefore, we allow multiple key pairs to appear in the  $\text{CCA}_2$  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

$$\text{dec}\left(\underbrace{t[\{u_1\}_{\text{pk}}^{n_1}, \dots, \{u_n\}_{\text{pk}}^{n_n}]}_s, \text{sk}\right) \sim \text{dec}\left(\underbrace{t[\{v_1\}_{\text{pk}}^{n_1}, \dots, \{v_n\}_{\text{pk}}^{n_n}]}_{s'}, \text{sk}\right)$$

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

To do this, we use the following function:

*Definition 25.* We define the function `else*` by induction:

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

*Example 17.* Let  $u \equiv t[\{v_1\}_{\text{pk}}^{n_1}, \{v_2\}_{\text{pk}}^{n_2}]$ . Then:

$$\begin{aligned} \text{else}^*(\text{eq}(u, \{v_1\}_{\text{pk}}^{n_1}), \text{eq}(u, \{v_2\}_{\text{pk}}^{n_2}), \text{dec}(u, \text{sk})) &\equiv \\ \text{if eq}(u, \{v_1\}_{\text{pk}}^{n_1}) \text{ then } \mathbf{0}(\text{dec}(u, \text{sk})) &\text{ else if eq}(u, \{v_2\}_{\text{pk}}^{n_2}) \text{ then } \mathbf{0}(\text{dec}(u, \text{sk})) \text{ else } \text{dec}(u, \text{sk}) \end{aligned}$$

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

*Definition of  $\text{CCA}_2$ .* 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  $\text{sk} \in \mathcal{K}$ , the secret key  $\text{sk}$  appears only in decryption position in  $\vec{u}$ ;  $\text{nodec}(\mathcal{K}, \vec{u})$  denotes that for all  $\text{sk}(n) \in \mathcal{K}$ , the only occurrences of  $n$  are in subterms  $\text{pk}(n)$ ;  $\text{hidden-rand}(\vec{r}; \vec{u})$  denotes that for all  $n_r \in \vec{r}$ ,  $n_r$  appears only in encryption randomness position and is not used with two distinct plaintexts.

We are now going to define by induction the  $\text{CCA}_2$  axiom. In order to do this we define by induction a binary relation  $R_{\text{CCA}_2}^{\mathcal{K}}$  on  $\text{CCA}_2$  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.  $(\phi, \mathcal{X}_{\text{enc}}, \mathcal{X}_{\text{dec}}, \sigma_{\text{rand}}, \theta_{\text{enc}}, \lambda_{\text{dec}})$  is a  $\text{CCA}_2$  execution if:

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

$\sigma_{\text{rand}}$ ,  $\theta_{\text{enc}}$  and  $\lambda_{\text{dec}}$  co-domains are the sets of, respectively, encryption randomness, encryption oracle calls and decryption oracle calls in  $\phi$ . Intuitively, we have:

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

when we can build  $\phi$  and  $\psi$  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  $R_{\text{CCA}_2^{\mathcal{K}}}^{\mathcal{K}}$  by induction:

- (1) **No Call to the Oracles:** if  $\mathcal{K} \sqsubseteq_d \phi$  then  $(\phi, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset) R_{\text{CCA}_2^{\mathcal{K}}}^{\mathcal{K}}(\phi, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$  for every sequence  $\phi$  of ground terms in  $\mathcal{T}(\mathcal{F}, \mathcal{N})$  such that  $\text{nodec}(\mathcal{K}; \phi)$ .
- (2) **Encryption Case:** Let  $x$  a fresh variable that does not appear in  $\mathcal{X}_{\text{enc}} \cup \mathcal{X}_{\text{dec}}$ ,  $\text{sk}$  be a secret key in  $\mathcal{K}$  and  $\text{pk}$  the corresponding public key. Then:

$$((\phi, \{u\}_{\text{pk}}^{n_r}, \mathcal{X}_{\text{enc}} \cup \{x\}, \mathcal{X}_{\text{dec}}, \sigma_{\text{rand}} \cup \{x \mapsto n_r\}, \theta_{\text{enc}} \cup \{x \mapsto \{u\}_{\text{pk}}^{n_r}\}, \lambda_{\text{dec}})$$

$$R_{\text{CCA}_2^{\mathcal{K}}}^{\mathcal{K}}((\psi, \{v\}_{\text{pk}}^{n'_r}, \mathcal{X}_{\text{enc}} \cup \{x\}, \mathcal{X}_{\text{dec}}, \sigma'_{\text{rand}} \cup \{x \mapsto n'_r\}, \theta'_{\text{enc}} \cup \{x \mapsto \{v\}_{\text{pk}}^{n'_r}\}, \lambda'_{\text{dec}})$$

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

- $(\phi, \mathcal{X}_{\text{enc}}, \mathcal{X}_{\text{dec}}, \sigma_{\text{rand}}, \theta_{\text{enc}}, \lambda_{\text{dec}}) R_{\text{CCA}_2^{\mathcal{K}}}^{\mathcal{K}}(\psi, \mathcal{X}_{\text{enc}}, \mathcal{X}_{\text{dec}}, \sigma'_{\text{rand}}, \theta'_{\text{enc}}, \lambda'_{\text{dec}})$
  - $u \equiv t \lambda_{\text{dec}}$ ,  $v \equiv t' \lambda'_{\text{dec}}$
  - $\text{nodec}(\mathcal{K}; t, t')$ , which ensures that the only decryptions are calls to the oracle.
  - $\text{fresh}(n_r, n'_r; \phi, u, \psi, v)$  and  $\text{hidden-rand}(\mathcal{X}_{\text{enc}} \sigma_{\text{rand}} \cup \mathcal{X}_{\text{enc}} \sigma'_{\text{rand}}; \phi, u, \psi, v)$
- (3) **Decryption Case:** Let  $\text{sk} \in \mathcal{K}$ ,  $\text{pk}$  the corresponding public key and  $z$  be a fresh variable. Then:

$$((\phi, \text{else}^*(l, \text{dec}(u, \text{sk}))), \mathcal{X}_{\text{enc}}, \mathcal{X}_{\text{dec}} \cup \{z\}, \sigma_{\text{rand}}, \theta_{\text{enc}}, \lambda_{\text{dec}} \cup \{z \mapsto \text{else}^*(l, \text{dec}(u, \text{sk}))\})$$

$$R_{\text{CCA}_2^{\mathcal{K}}}^{\mathcal{K}}((\psi, \text{else}^*(l', \text{dec}(v, \text{sk}))), \mathcal{X}_{\text{enc}}, \mathcal{X}_{\text{dec}} \cup \{z\}, \sigma'_{\text{rand}}, \theta'_{\text{enc}}, \lambda'_{\text{dec}} \cup \{z \mapsto \text{else}^*(l', \text{dec}(v, \text{sk}))\})$$

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

- $(\phi, \mathcal{X}_{\text{enc}}, \mathcal{X}_{\text{dec}}, \sigma_{\text{rand}}, \theta_{\text{enc}}, \lambda_{\text{dec}}) R_{\text{CCA}_2^{\mathcal{K}}}^{\mathcal{K}}(\psi, \mathcal{X}_{\text{enc}}, \mathcal{X}_{\text{dec}}, \sigma'_{\text{rand}}, \theta'_{\text{enc}}, \lambda'_{\text{dec}})$
- $u \equiv t \theta_{\text{enc}} \lambda_{\text{dec}}$  and  $v \equiv t \theta'_{\text{enc}} \lambda'_{\text{dec}}$ .
- Consider the set  $\mathcal{Y}_u$  of variables  $x \in \mathcal{X}_{\text{enc}}$  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  $\{\_ \}_{\text{pk}}^{n_x} \in \text{codom}(\theta_{\text{enc}})$  by  $\{0\}_{\text{pk}}^{n_x}$ :

$$x \sigma_{\text{rand}} \in u \{ \{0\}_{\text{pk}}^{n_x} / \{\_ \}_{\text{pk}}^{n_x} \mid \{\_ \}_{\text{pk}}^{n_x} \in \text{codom}(\theta_{\text{enc}}) \} \downarrow R$$

Then  $l$  is the sequence of guards  $l \equiv (\text{eq}(u, y_1), \dots, \text{eq}(u, y_m))$  where  $(y_1, \dots, y_m) = \text{sort}(\mathcal{Y}_u \theta_{\text{enc}})$ .

Similarly,  $l' \equiv (\text{eq}(v, y'_1), \dots, \text{eq}(v, y'_m))$  where  $(y'_1, \dots, y'_m) = \text{sort}(\mathcal{Y}_u \theta'_{\text{enc}})^{10}$ .

- $\text{nodec}(\mathcal{K}; t)$  and  $\text{hidden-rand}(\mathcal{X}_{\text{enc}} \sigma_{\text{rand}} \cup \mathcal{X}_{\text{enc}} \sigma'_{\text{rand}}; \phi, u, \psi, v)$

<sup>10</sup>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  $\text{sort}$  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  $\text{eq}(u, x\theta_{\text{enc}})$  for every  $x \in \mathcal{X}_{\text{enc}}$  such that  $x\theta_{\text{enc}}$  is an encryption using public-key  $\text{pk}$ .

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

$$\alpha_0 \mapsto \{m_0\}_{\text{pk}}^{n_0} \quad \alpha_1 \mapsto \{m_1\}_{\text{pk}}^{n_1} \quad \alpha_2 \mapsto \{\alpha_1\}_{\text{pk}}^{n_2}$$

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

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

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

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

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

Finally, the  $\text{sort}$  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.  $\diamond$

We can now define the recursive set of axioms  $\text{CCA}_2^a$  and show their validity. We also state and prove a key property of these axioms.

*Definition 28.*  $\text{CCA}_2^a$  is the set of atomic axioms  $\phi \sim \psi\mu$ , where  $\mu$  is a renaming of names in  $\mathcal{N}$  and there exist two  $\text{CCA}_2$  executions  $\mathcal{Y}, \mathcal{Y}'$  such that:

$$\mathcal{Y} = (\phi, \mathcal{X}_{\text{enc}}, \mathcal{X}_{\text{dec}}, \sigma_{\text{rand}}, \theta_{\text{enc}}, \lambda_{\text{dec}}) \quad \mathcal{Y}' = (\psi, \mathcal{X}_{\text{enc}}, \mathcal{X}_{\text{dec}}, \sigma'_{\text{rand}}, \theta'_{\text{enc}}, \lambda'_{\text{dec}}) \quad \mathcal{Y} R_{\text{CCA}_2^a}^{\mathcal{K}} \mathcal{Y}'$$

In that case, we say that  $(\mathcal{Y}, \mathcal{Y}')$  is a valid  $\text{CCA}_2^a$  application, and  $\phi \sim \psi\mu$  is a valid  $\text{CCA}_2^a$  instance.

**PROPOSITION 10.** *All formulas in  $\text{CCA}_2^a$  are computationally valid if the encryption scheme is  $\text{IND-CCA}_2$ .*

**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  $\mu$  empty. Let  $\mathcal{M}_c$  be a computational model where the encryption and decryption symbol are interpreted as an  $\text{IND-CCA}_2$  encryption scheme. Let  $\phi \sim \psi$  be a valid instance of  $\text{CCA}_2^a$  such that  $\llbracket \phi \rrbracket \not\approx_{\mathcal{M}_c} \llbracket \psi \rrbracket$  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  $\text{CCA}_2$  we know that there exist two  $\text{CCA}_2$  executions such that:

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

We are going to build from  $\phi$  and  $\psi$  a winning attacker against the multi-user  $\text{IND-CCA}_2$  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  $R_{\text{CCA}_2^a}^{\mathcal{K}}$  a algorithm  $\mathcal{B}$  that samples from  $\llbracket \phi \rrbracket$  or  $\llbracket \psi \rrbracket$  (depending on the oracles internal bit). The algorithm  $\mathcal{B}$  uses a memoisation technique: it builds a store whose keys are subterms of  $\phi, \psi$  already encountered and variable in  $\mathcal{X}_{\text{enc}} \cup \mathcal{X}_{\text{dec}}$ , and values are elements of the  $\mathcal{M}_c$  domain.

(1)  $(\phi, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)R_{\text{CCA}_2^a}^{\mathcal{K}}(\phi, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$ : for every term  $t$  in the vector  $\phi$ ,  $\mathcal{B}$  samples from  $\llbracket t \rrbracket$  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  $\text{nodec}(\mathcal{K}, \phi)$  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  $sk \in \mathcal{K}$  and the challenger sent us a random sample from  $\llbracket pk \rrbracket$ , or  $sk$  does not appear in  $\mathcal{K}$  and then  $\mathcal{B}$  can draw the corresponding key pair itself.

- $f(t_1, \dots, t_n)$ , then  $\mathcal{B}$  inductively samples the function arguments  $(\llbracket t_1 \rrbracket, \dots, \llbracket t_n \rrbracket)$  and then samples from  $\llbracket f \rrbracket(\llbracket t_1 \rrbracket, \dots, \llbracket t_n \rrbracket)$ .  $\mathcal{B}$  stores the value at the key  $f(t_1, \dots, t_n)$ .

(2) **Encryption Case:**

$((\phi, \{u\}_{pk}^{n_r}), \mathcal{X}_{\text{enc}} \cup \{x\}, \mathcal{X}_{\text{dec}}, \sigma_{\text{rand}} \cup \{x \mapsto n_r\}, \theta_{\text{enc}} \cup \{x \mapsto \{u\}_{pk}^{n_r}\}, \lambda_{\text{dec}})$

$R_{\text{CCA}_2^a}^{\mathcal{K}}((\psi, \{v\}_{pk}^{n'_r}), \mathcal{X}_{\text{enc}} \cup \{x\}, \mathcal{X}_{\text{dec}}, \sigma'_{\text{rand}} \cup \{x \mapsto n'_r\}, \theta'_{\text{enc}} \cup \{x \mapsto \{v\}_{pk}^{n'_r}\}, \lambda'_{\text{dec}})$

Since we have  $\text{fresh}(n_r, n'_r; \phi, u, \psi, v)$  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  $\mathcal{X}_{\text{enc}} \cup \mathcal{X}_{\text{dec}}$  and that, if  $x \in \mathcal{X}_{\text{enc}}$ , then the store value of  $x$  is either sampled from  $\llbracket x\theta_{\text{enc}} \rrbracket$  or from  $\llbracket x\theta'_{\text{enc}} \rrbracket$  (depending on the challenger internal bit), and that if  $x \in \mathcal{X}_{\text{dec}}$  then the store value of  $x$  is either sampled from  $\llbracket x\lambda_{\text{dec}} \rrbracket$  or from  $\llbracket x\lambda'_{\text{dec}} \rrbracket$  (depending on the challenger internal bit). We also observe that if the challenger internal bit is 0 then for all  $w$ :

$$\mathcal{O}_{\text{LR}}(\text{pk}, b)(\llbracket u \rrbracket, \llbracket v \rrbracket) = \mathcal{O}_{\text{LR}}(\text{pk}, b)(\llbracket u \rrbracket, w)$$

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

$$\mathcal{O}_{\text{LR}}(\text{pk}, b)(\llbracket u \rrbracket, \llbracket v \rrbracket) = \mathcal{O}_{\text{LR}}(\text{pk}, b)(w, \llbracket v \rrbracket)$$

$\mathcal{B}$  samples two values  $\alpha, \beta$  such that if the challenger internal bit is 0 then  $\alpha$  is sampled from  $\llbracket u \rrbracket$  and if the challenger internal bit is 1 then  $\beta$  is sampled from  $\llbracket v \rrbracket$ . Therefore whatever the challenger internal is bit,  $\mathcal{O}_{\text{LR}}(\text{pk}, b)(\alpha, \beta)$  is sampled from  $\mathcal{O}_{\text{LR}}(\text{pk}, b)(\llbracket u \rrbracket, \llbracket v \rrbracket)$ :

- $\alpha$  is sampled from  $\llbracket u \rrbracket$  using the case 1 algorithm. Remark that when we encounter a decryption under  $sk' \in \mathcal{K}$ , we know that it was already sampled and can therefore retrieve it from the store.
- similarly,  $\beta$  is sampled from  $\llbracket v \rrbracket$  using the case 1 algorithm.

The condition  $\text{nodec}(\mathcal{K}; t, t')$  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  $\text{fresh}(n_r, n'_r; \phi, u, \psi, v)$  and  $\text{hidden-rand}(\mathcal{X}_{\text{enc}}\sigma_{\text{rand}} \cup \mathcal{X}_{\text{enc}}\sigma'_{\text{rand}}; \phi, u, \psi, v)$  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:

$((\phi, \text{else}^*(l, \text{dec}(u, \text{sk}))), \mathcal{X}_{\text{enc}}, \mathcal{X}_{\text{dec}} \cup \{z\}, \sigma_{\text{rand}}, \theta_{\text{enc}}, \lambda_{\text{dec}} \cup \{z \mapsto \text{else}^*(l, \text{dec}(u, \text{sk}))\})$

$R_{\text{CCA}_2^a}^{\mathcal{K}}((\psi, \text{else}^*(l', \text{dec}(v, \text{sk}))), \mathcal{X}_{\text{enc}}, \mathcal{X}_{\text{dec}} \cup \{z\}, \sigma'_{\text{rand}}, \theta'_{\text{enc}}, \lambda'_{\text{dec}} \cup \{z \mapsto \text{else}^*(l', \text{dec}(v, \text{sk}))\})$

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

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

As we observed in Remark 5, if the challenger internal bit is 0, checking whether  $\gamma$  is different from the values sampled from  $\llbracket x_1\theta_{\text{enc}} \rrbracket, \dots, \llbracket x_p\theta_{\text{enc}} \rrbracket$  amounts to checking whether  $\gamma$  is different from the values sampled from  $\llbracket y_1 \rrbracket, \dots, \llbracket y_m \rrbracket$ , 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  $x \in \mathcal{X}_{\text{enc}}$  such that the encryption binded to  $x$  in  $\theta_{\text{enc}}$  appears directly in the *left decryption*  $u$ :

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

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

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

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

We store the result at key  $z$ .

The attacker against the multi-user IND-CCA<sub>2</sub> game simply returns  $\mathcal{A}(\mathcal{B})$ . Since  $\mathcal{B}$  samples either from  $\llbracket \phi \rrbracket$  if  $b = 0$  or from  $\llbracket \psi \rrbracket$  if  $b = 1$  (up to a negligible number of samplings), and since  $\mathcal{A}$  has a non-negligible advantage of distinguishing  $\llbracket \phi \rrbracket$  from  $\llbracket \psi \rrbracket$  we know that the attacker has a non-negligible advantage against the multi-user IND-CCA<sub>2</sub> game.  $\square$

## B.1 Closure Under Restr

To close our logic under Restr, we need the atomic axioms to be closed. Therefore, we let CCA<sub>2</sub> be the closure of CCA<sub>2</sub><sup>a</sup> under Restr.

*Definition 29.* CCA<sub>2</sub> is the set of formula  $\phi \sim \psi$  such that we have the derivation:

$$\frac{\phi' \sim \psi'}{\phi \sim \psi} \text{CCA}_2^a \text{ Restr}$$

The main contribution of this sub-section, given below, states that any instance  $\vec{u} \sim \vec{v}$  of CCA<sub>2</sub> can be automatically extended into an instance  $\vec{u}' \sim \vec{v}'$  of CCA<sub>2</sub><sup>a</sup> of, at most, polynomial size.

**PROPOSITION 11.** *For every instance  $\vec{u} \sim \vec{v}$  of  $\text{CCA}_2$ , 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  $\text{CCA}_2^a$  (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  $\text{CCA}_2$  into an instance of  $\text{CCA}_2^a$ . Let  $(u_i)_{i \in I} \sim (v_i)_{i \in I}$  be an instance of  $\text{CCA}_2$ . Let  $I' \subseteq I$ , we want to extend  $(u_i)_{i \in I'} \sim (v_i)_{i \in I'}$  into an instance of  $\text{CCA}_2^a$ . 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  $\text{CCA}_2^a$  we have:

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

For all  $x \in \mathcal{X}_{\text{enc}} \cup \mathcal{X}_{\text{dec}}$ , we let  $i_x \in I$  be the index corresponding to  $x\theta_{\text{enc}}\lambda_{\text{dec}} \sim x\theta'_{\text{enc}}\lambda'_{\text{dec}}$ . Moreover, for all  $x \in \mathcal{X}_{\text{dec}}$ , we let  $t_{i_x}$  be the context used for the decryption in the definition of  $R_{\text{CCA}_2^a}^{\mathcal{K}}$  (hence we have  $x\lambda_{\text{dec}} \equiv \text{else}^*(l, \text{dec}(t_{i_x}, \theta_{\text{enc}}\lambda_{\text{dec}}), \text{sk}))$ ).

*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 disjoint and  $I' \subseteq I^{lr}$ .
- $(\tilde{u}_i)_{i \in J} \sim (\tilde{v}_i)_{i \in J}$  is an instance of  $\text{CCA}_2^a$  of polynomial size with respect to  $\sum_{i \in I'} |u_i| + |v_i|$ .

Intuitively,  $I^{lr}$  is the subset of indices of  $I \setminus I'$  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  $i \in I^{lr}$ ,  $u_i \in \text{st}((u_i)_{i \in I'})$  and  $v_i \in \text{st}((v_i)_{i \in I'})$ . 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  $I \setminus I'$  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  $u_i \in \text{st}((u_i)_{i \in I'})$ ). 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  $I \setminus I'$  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  $\text{CCA}_2^a$  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  $R_{\text{CCA}_2^a}^{\mathcal{K}}$ :

- **No Call to the Oracles:** In that case we take  $I_i^l = I_i^r = \{i\}$ .
- **Encryption Case:** let  $t, t' \in \mathcal{T}(\mathcal{F}_0, \mathcal{N}, \mathcal{X}_{\text{dec}})$  such that  $u_i \equiv \{t\lambda_{\text{dec}}\}_-$  and  $v_i \equiv \{t'\lambda'_{\text{dec}}\}_-$ . To have  $u_i$  well-defined, we need all the decryptions in  $u_i$  to be well-defined (same for  $v_i$ ). Hence let:

$$I_i^l = \{i\} \cup \bigcup_{x \in \mathcal{X}_{\text{dec}} \cap \text{st}(t)} I_{i_x}^l \quad I_i^r = \{i\} \cup \bigcup_{x \in \mathcal{X}_{\text{dec}} \cap \text{st}(t')} I_{i_x}^r$$

- **Decryption Case:** recall that  $u_i \equiv \text{else}^*(l, \text{dec}(u, \text{sk}))$  where  $u \equiv t_i\theta_{\text{enc}}\lambda_{\text{dec}}$ . Therefore we need all encryption in  $\mathcal{X}_{\text{enc}} \cap \text{st}(t_i)$  and decryption in  $\mathcal{X}_{\text{dec}} \cap \text{st}(t_i)$  to be defined, on the left and on the right. Hence we let:

$$I_i^l = \{i\} \cup \bigcup_{x \in (\mathcal{X}_{\text{dec}} \cup \mathcal{X}_{\text{enc}}) \cap \text{st}(t_i)} I_{i_x}^l \quad I_i^r = \{i\} \cup \bigcup_{x \in (\mathcal{X}_{\text{dec}} \cup \mathcal{X}_{\text{enc}}) \cap \text{st}(t_i)} I_{i_x}^r$$

We let:

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

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

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  $\text{cca}_2^a$  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:

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

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:

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

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

## B.2 Length in the $\text{cca}_2$ Axioms

If we want the formula  $\{t\}_{\text{pk}}^r \sim \{t'\}_{\text{pk}}^{r'}$  to be a valid application of the  $\text{cca}_2$  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  $\text{EQL}(\_, \_)$  in the logic, together with some derivation rules  $\mathcal{D}_L$  (supplied by the user), and we require that they verify the following properties:

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

We add to all  $\text{cca}_2$  instances the side condition  $\text{EQL}(m_l, m_r)$  for every encryption oracle call on  $(m_l, m_r)$ . Then, we know that our  $\text{cca}_2$  instances are valid in any computational model  $\mathcal{M}_c$  where

$$\begin{aligned}
& \text{Length}(n) = l_\eta & \text{Length}(0_{l_e}) = l_e \\
& \text{Length}(u) = \text{Length}(u') \text{ if } u =_R u' \text{ and } \text{Length}(u), \text{Length}(u') \text{ are not undefined} \\
& \text{Length}(\langle u, v \rangle) = \text{Length}(u) + \text{Length}(v) + l_{\langle, \rangle} & \forall l_e. \text{Length}(\text{pad}_{l_e}(u)) = l_e \\
& \forall k. \text{Length}(\{u\}_{\text{pk}}^n) = k.l_{\{\text{block}\}} + l_{\{\}} \text{ if } \text{Length}(u) = k.l_{\{\text{block}\}} \\
& \forall k. \text{Length}(\text{dec}(u, \text{sk})) = k.l_{\{\text{block}\}} \text{ if } \text{Length}(u) = k.l_{\{\text{block}\}} + l_{\{\}} \\
& \text{Length}(\text{if } b \text{ then } u \text{ else } v) = \begin{cases} \text{Length}(u) & \text{if } \text{Length}(u) = \text{Length}(v) \\ \text{undefined} & \text{otherwise} \end{cases}
\end{aligned}$$

Fig. 11. Definition of the Length partial function.

the encryption is interpreted as a IND-CCA<sub>2</sub> encryption scheme, and where the following property holds: for every ground terms  $u, v$ , if  $\text{EQL}(u, v)$  is derivable using  $\mathcal{D}_L$ , then:

$$\llbracket \text{length}(u) \rrbracket_{\mathcal{M}_c} = \llbracket \text{length}(v) \rrbracket_{\mathcal{M}_c}$$

*Example: Block Cipher.* We give here an example of derivation rules  $\mathcal{D}_L$  that axiomatize the fact that the encryption function is built upon a block cipher, taking blocks of length  $l_{\{\text{block}\}}$  and returning blocks of length  $l_{\{\text{block}\}}$ . 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  $\text{pad}_{l_e} \in \mathcal{F}$ . Intuitively  $\text{pad}_{l_e}$  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:  $l_{\langle, \rangle}, l_{\text{enc}}, l_{\{\text{block}\}}, l_\eta$ .

We define the Length (partial) function on terms in Figure 11. Then, we let  $\mathcal{D}_L$  be the (recursive) set of atomic axioms:

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

**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  $\llbracket \text{Length} \rrbracket_{\mathcal{M}_c}$  is just an under-approximation of  $\llbracket \text{length} \rrbracket_{\mathcal{M}_c}$  in every computational model  $\mathcal{M}_c$  where the encryption is interpreted as a block cipher, the padding functions are interpreted as expected etc.

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

*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  $\text{Length}(A) = \text{Length}(B) = \text{Length}(C) = l_{\text{agent}}$ .  $\diamond$

## C SHAPE OF THE TERMS

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

### C.1 Definitions

*Definition 30.* A  $\text{CCA}_2$  trace  $\mathcal{S}$  is a tuple  $(\mathcal{K}, \mathcal{R}, \mathcal{E}, \mathcal{D})$  where:

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

We can associate to every  $\text{CCA}_2$  instance a left and a right  $\text{CCA}_2$  trace.

*Definition 31.* Given a  $\text{CCA}_2$  instance  $\phi \sim \psi$  and its corresponding  $\text{CCA}_2^a$  application:

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

we define the left  $\text{CCA}_2$  trace  $\mathcal{S} = \text{l-trace}(\phi \sim \psi)$  by:

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

We define similarly its right  $\text{CCA}_2$  trace  $\mathcal{S}' = \text{r-trace}(\phi \sim \psi)$ .

Let  $\phi \sim \psi$  be a  $\text{CCA}_2$  instance and  $\mathcal{S} = \text{l-trace}(\phi \sim \psi)$  be its left  $\text{CCA}_2$  trace. We use  $\mathcal{S}$  to define the normal form of the terms appearing, on the left, in branch using the  $\text{CCA}_2$  instance  $\phi \sim \psi$ . This is done through four mutually inductive definitions:

- *S-encryption oracle calls* are well-formed encryptions.
- *S-decryption oracle calls* are well-formed decryptions.
- *S-normalized basic terms* are terms built using function symbols in  $\mathcal{F}_{\setminus \text{if}, \mathbf{0}}$  and well-formed encryptions and decryptions.
- *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 *S-simple terms* to be terms having a particular form, and *S-normalized simple terms* to be *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  $(\text{pk}(n), \text{sk}(n))$  where  $n$  is a name.

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

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

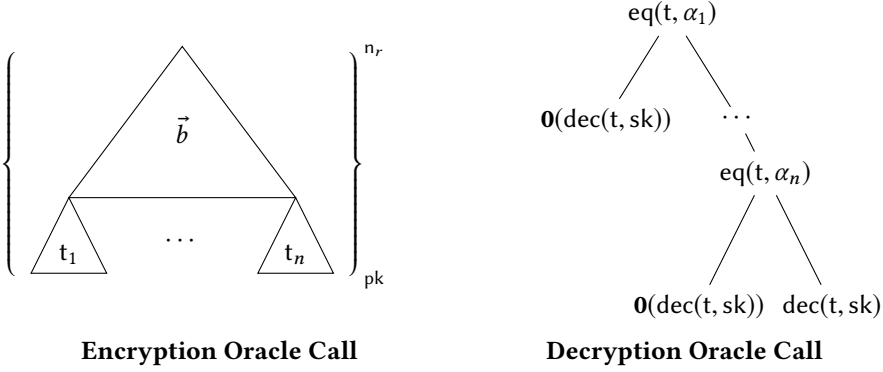
- $\{u\}_{\text{pk}}^{\text{n}_e} \in \mathcal{E}$ ,  $\text{n}_e \in \mathcal{R}$ ,  $(\text{pk}, \text{sk})$  is a valid public/private key pair and with  $\text{sk} \in \mathcal{K}$ .
- $u$  is a *S-normalized simple terms*.

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

*Definition 34.* A *S-decryption oracle call* is a term of the form  $C[\vec{g} \diamond (s_i)_{i \leq p}]$  in  $\mathcal{D}$  where:

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





**Convention:**  $\alpha_1, \dots, \alpha_n$  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:

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

- For all  $j$ ,  $\alpha_j$  is a  $\mathcal{S}$ -encryption oracle call.
- For all  $k$ ,  $\text{dec}_k$  is a  $\mathcal{S}$ -decryption oracle call.

$(\alpha_j)_j$  are called  $u$ 's encryptions. We often write  $(\text{dec}_k)_k$  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  $\mathcal{F}_{\text{if}, \mathbf{0}}$  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  $\text{CCA}_2$  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  $U[\vec{w}, (\alpha_j)_j, (\text{dec}_k)_k]$  where:

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

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

A  $\mathcal{S}$ -normalized basic term is 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  $U[\vec{w}, (\alpha_j)_j, (\text{dec}_k)_k]$  where:

- $(\alpha_j)_j$  are encryptions under  $(pk_j, sk_j)_j$ , and  $(\text{dec}_k)_k$  are decryptions under  $(pk_k, sk_k)_k$ .
- $U[\vec{w}, (\{\square_j\}_{pk_j}^0)_j, (\text{dec}(\square_k, sk_k))_k]$  is in  $R$ -normal form.

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

Finally, a  $\mathcal{S}$ -simple term is a term build using only  $\mathcal{S}$ -basic term and the  $\text{if\_then\_else\_}$  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  $\text{nodec}(\mathcal{K}, \cdot)$ .  $\diamond$

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

*Definition 38.*  $<_{\text{ind}}^{\mathcal{S}}$  is the reflexive and transitive closure of the relation  $<^{\mathcal{S}}$  defined as:

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

$$t \equiv C [\vec{g} \diamond (s_i[(\alpha_j)_j, (\text{dec}_k)_k])_{i \leq p}]$$

for all  $j$ ,  $\alpha_j <^{\mathcal{S}} t$  and for all  $k$ ,  $\text{dec}_k <^{\mathcal{S}} t$ .

- For all  $\mathcal{S}$ -normalized basic term  $t \equiv U[\vec{w}, (\alpha_j)_j, (\text{dec}_k)_k]$ , for all  $j$ ,  $\alpha_j <^{\mathcal{S}} t$  and for all  $k$ ,  $\text{dec}_k <^{\mathcal{S}} t$ .
- 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  $\leq_{\text{bt}}^{\mathcal{S}}$  be union of the restriction of  $<_{\text{ind}}^{\mathcal{S}}$  to the instances where the left term is a  $\mathcal{S}$ -normalized basic term, and the set of guards appearing in the right-term. Formally:

*Definition 39.* Let  $<_{\text{ind}}^{\prime\mathcal{S}}$  be the reflexive and transitive closure of the order  $<^{\prime\mathcal{S}}$ , which has the same definition than  $<^{\mathcal{S}}$ , apart for the  $\mathcal{S}$ -decryption oracle call:

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

$$t \equiv C [\vec{g} \diamond (s_i[(\alpha_j)_j, (\text{dec}_k)_k])_{i \leq p}]$$

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

We finally define  $\leq_{\text{bt}}^{\mathcal{S}}$  by requiring that for every terms  $u, v$ :

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

## C.2 Eager Reduction for $\mathcal{A}_{\text{FA}_s}$

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

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  $\text{FA}_s$  and  $\text{Dup}$  can be easily re-combined, this is not the case for  $\text{CCA}_2$ . Therefore, given a finite family of  $\text{CCA}_2$  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  $\mathcal{A}_{\text{FA}_s}$ , we let  $\text{instance}(P)$  be the unique  $\text{CCA}_2$  instance used in  $P$ .

*Example 18.* If  $P$  is the proof:

$$\frac{\frac{\vec{w}, (\alpha_i)_{i \in I}, (\text{dec}_j)_{j \in J} \sim \vec{w}, (\alpha'_i)_{i \in I}, (\text{dec}'_j)_{j \in J} \quad \text{CCA}_2}{\vdots}}{C[\vec{w}, (\alpha_i)_{i \in I}, (\text{dec}_j)_{j \in J}] \sim C[\vec{w}, (\alpha'_i)_{i \in I}, (\text{dec}'_j)_{j \in J}]} \quad \text{FA}_s^* \cdot \text{Dup}^*$$

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

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

*Definition 41.* A  $\text{CCA}_2$  instance:

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

is a sub-instance of a  $\text{CCA}_2$  instance:

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

if and only if  $\text{st}(\vec{w}_0) \subseteq \text{st}(\vec{w})$ ,  $I_0 \subseteq I$  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  $\text{CCA}_2$  instance  $\vec{u} \sim \vec{v}$  such that for every  $i$ ,  $\text{instance}(P_i)$  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  $\text{CCA}_2$  instance  $\vec{u} \sim \vec{v}$  such that for every  $n$ ,  $\text{instance}(P_n)$  is a sub-instance of  $\vec{u} \sim \vec{v}$ , then there exists  $P$  such that:*

- $P \vdash_{\mathcal{A}_{FA_s}} (\beta_n)_{n \in N} \sim (\beta'_n)_{n \in N}$
- $\text{instance}(P)$  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:

$$\text{if } \frac{\vec{u}' \sim \vec{v}'}{\vec{u} \sim \vec{v}} Ax \quad \text{then for every } \vec{w}_l, \vec{w}_r \text{ of the same length } \frac{\vec{w}_l, \vec{u}' \sim \vec{w}_r, \vec{v}'}{\vec{w}_l, \vec{u} \sim \vec{w}_r, \vec{v}} Ax$$

Therefore we can easily combine all proofs  $(P_n)_{n \in N}$ . For every  $n \in N$ , we let  $\text{instance}(P_n) \equiv \vec{u}_n \sim \vec{u}'_n$ . 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:

$$\frac{\frac{(\vec{v}_n)_{n \in N} \sim (\vec{v}'_n)_{n \in N}}{(\vec{u}_n)_{n \in N} \sim (\vec{u}'_n)_{n \in N}} \text{Dup}^* \quad \vdots}{(\beta_n)_{n \in N} \sim (\beta'_n)_{n \in N}}$$

Now, we want to conclude by applying the  $\text{CCA}_2$  axiom. The problem is that  $\text{CCA}_2$  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  $\mathcal{A}_{FA_s}$ .  $\square$

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  $S, S'$  be the, respectively, left and right  $\text{CCA}_2$  trace corresponding to  $\text{instance}(P)$ . If  $\beta$  and  $\beta'$  are, respectively,  $S$ -basic term and  $S'$ -basic term then there exist  $\gamma =_R \beta$  and  $\gamma' =_R \beta'$  such that:*

- $\gamma$  and  $\gamma'$  are, respectively,  $S$ -normalized basic term and  $S'$ -normalized basic term.
- There exists  $P'$  such that  $P' \vdash_{\mathcal{A}_{FA_s}} \gamma \sim \gamma'$ ,  $\text{instance}(P')$  is a sub-instance of  $\text{instance}(P)$  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  $\text{FA}_{\mathcal{S}}$  rules in  $P$ . If  $P$  has no  $\text{FA}_{\mathcal{S}}$  application, then we have three cases:

- $\beta$  and  $\beta'$  are identical, up to  $\alpha$ -renaming. In that case, we can check that  $\gamma \equiv \beta \downarrow_R$  and  $\gamma' \equiv \beta' \downarrow_R$  satisfy the wanted properties.
- $\beta$  and  $\beta'$  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  $\gamma' \equiv \beta'$ .
- $\beta$  and  $\beta'$  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,  $\beta$  and  $\beta'$  must start with the same function symbol. Hence:

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

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

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

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

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

$$P_1 \vdash_{\mathcal{A}_{\text{FA}_{\mathcal{S}}}} \gamma_1 \sim \gamma'_1, \quad \dots, \quad P_n \vdash_{\mathcal{A}_{\text{FA}_{\mathcal{S}}}} \gamma_n \sim \gamma'_n$$

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

$$P' \vdash_{\mathcal{A}_{\text{FA}_{\mathcal{S}}}} (\gamma_n)_{n \in N} \sim (\gamma'_n)_{n \in N}$$

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

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

$$\frac{\begin{array}{c} \vdots \\ (P') \\ \gamma_1, \dots, \gamma_n \sim \gamma'_1, \dots, \gamma'_n \end{array}}{f(\gamma_1, \dots, \gamma_n) \sim f(\gamma'_1, \dots, \gamma'_n)} \text{FA}_f$$

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

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

$$\frac{\langle \gamma_1, \gamma_2 \rangle \sim \langle \gamma'_1, \gamma'_2 \rangle}{\pi_1(\langle \gamma_1, \gamma_2 \rangle) \sim \pi_1(\langle \gamma'_1, \gamma'_2 \rangle)} \text{FA}_{\pi_1}$$

We look at the next rule in  $P'$ :

- If it is  $\text{CCA}_2$ , then  $\langle \gamma_1, \gamma_2 \rangle$  and  $\langle \gamma'_1, \gamma'_2 \rangle$  are the same terms, up to  $\alpha$ -renaming. We conclude by taking  $\gamma \equiv \gamma_1$  and  $\gamma' \equiv \gamma'_1$ .
- Or it is a function application:

$$\frac{\begin{array}{c} \vdots \\ (Q) \\ \hline \gamma_1, \gamma_2 \sim \gamma'_1, \gamma'_2 \end{array} \text{FA}_{\langle \cdot, \cdot \rangle}}{\langle \gamma_1, \gamma_2 \rangle \sim \langle \gamma'_1, \gamma'_2 \rangle} \text{FA}_{\pi_1} \\ \pi_1(\langle \gamma_1, \gamma_2 \rangle) \sim \pi_1(\langle \gamma'_1, \gamma'_2 \rangle)$$

Using Lemma 1, we extract from  $Q$  a proof  $Q' \vdash_{\mathcal{A}_{\text{FA}_3}} \gamma_1 \sim \gamma'_1$  no larger than  $Q$ . We conclude by taking  $\gamma \equiv \gamma_1$  and  $\gamma' \equiv \gamma'_1$ :

$$\frac{\begin{array}{c} \vdots \\ (Q') \\ \hline \gamma_1 \sim \gamma'_1 \end{array} R}{\pi_1(\langle \gamma_1, \gamma_2 \rangle) \sim \pi_1(\langle \gamma'_1, \gamma'_2 \rangle)} \quad (8)$$

- Only one side can be reduced by  $\pi_i(\langle x_1, x_2 \rangle) \rightarrow x_i$ . Therefore the next rule applied in  $(P')$  must be  $\text{CCA}_2$  (since the head function symbols differ). But in a  $\text{CCA}_2$  application, we cannot have  $\langle \_ , \_ \rangle \sim f'(\_)$  with  $f' \neq \langle \cdot, \cdot \rangle$ . Contradiction.
- Both sides can be reduced by  $\text{dec}(\{x\}_{\text{pk}(n)}, \text{sk}(n)) \rightarrow x$ . Hence  $n = 2$ ,  $\gamma_1, \gamma_2 \equiv \{u\}_{\text{pk}(n)}^r, \text{sk}(n)$ ,  $\gamma'_1, \gamma'_2 \equiv \{u'\}_{\text{pk}(n')}^{r'}$ ,  $\text{sk}(n')$  and  $P'$  is of the form:

$$\frac{\{u\}_{\text{pk}(n)}^r, \text{sk}(n) \sim \{u'\}_{\text{pk}(n')}^{r'}, \text{sk}(n')}{\text{dec}(\{u\}_{\text{pk}(n)}^r, \text{sk}(n)) \sim \text{dec}(\{u'\}_{\text{pk}(n')}^{r'}, \text{sk}(n'))} \text{FA}_{\text{dec}}$$

We look at the next rule applied on  $\{u\}_{\text{pk}(n)}^r, \_ \sim \{u'\}_{\text{pk}(n')}^{r'}, \_$ . If it is a function application then we have a shortcut using Lemma 1, as we did for (8). If it is  $\text{CCA}_2$ , we have two cases:

- $\{u\}_{\text{pk}(n)}^r$  and  $\{u'\}_{\text{pk}(n')}^{r'}$  are the same terms, up to  $\alpha$ -renaming. We conclude by taking  $\gamma \equiv u$  and  $\gamma' \equiv u'$ .
- $\{u\}_{\text{pk}(n)}^r$  and  $\{u'\}_{\text{pk}(n')}^{r'}$  are, respectively, a  $\mathcal{S}$ -encryption oracle call and a  $\mathcal{S}'$ -encryption oracle call. Then  $\text{sk}(n) \in \mathcal{K}$ . Since  $\gamma_2 \equiv \text{sk}(n)$  and  $\gamma_2$  is a  $\mathcal{S}$ -normalized basic term, we know that  $\text{nodec}(\mathcal{K}, \text{sk}(n))$ . Contradiction.
- Only one side can be reduced by  $\text{dec}(\{x\}_{\text{pk}(n)}^r, \text{sk}(n)) \rightarrow x$ . Then  $(P')$  is necessarily of the form:

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

We look at the next rule applied to  $\{t\}_{\text{pk}(n)}^r$  and  $\{t'\}_{p'}^{r'}$ :

- If it is  $\text{CCA}_2$ , then  $p' \equiv \text{pk}(n')$ . Therefore the right side can be reduced by  $\text{dec}(\{x\}_{\text{pk}(n')}^r, \text{sk}(n')) \rightarrow x$ . Contradiction.
- If it is  $\text{FA}_{\langle \_ , \_ \rangle}$  then there is a proof of  $\_ \text{pk}(n), \text{sk}(n) \sim \_ , p', \text{sk}(n')$ , which implies that  $p' \equiv \text{pk}(n')$ . Therefore the right side can be reduced by  $\text{dec}(\{x\}_{\text{pk}(n')}^r, \text{sk}(n')) \rightarrow x$ . Contradiction.
- Both side can be reduced by  $\text{eq}(x, x) \rightarrow \text{true}$ . In this case the proof cut elimination is trivial.
- Only one side can be reduced by  $\text{eq}(x, x) \rightarrow \text{true}$ . Therefore we have a proof of the form:

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

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

## D PROOF FORM

### D.1 Early Proof Form

We showed in Lemma 8 that:

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

is complete for  $\mathfrak{F}((\text{CS} + \text{FA} + \text{R} + \text{Dup} + \text{CCA}_2)^*)$ . Let us consider a proof  $P$  following this ordering. The only branching rule in  $\mathcal{A}_{>}$  is the  $\text{CS}_{\square}$  rule, which has two premises. Hence after having completed all the  $\text{CS}_{\square}$  applications we know that the proof will be non-branching and in  $\mathcal{A}_{\overline{\text{BFA}}}$ . We want to name each branch of the proof tree, and its corresponding instance of the  $\text{CCA}_2$  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  $\vdash^b$  be the proof system  $\vdash$  with branch annotations. When  $P \vdash^b t \sim t'$ , we let  $\text{label}(P)$  be the set of labels annotating the branches in  $P$ , and for all  $l \in \text{label}(P)$ , we let  $\text{instance}(P, l)$  be the  $\text{CCA}_2$  instance used in branch  $l$ .

When applying the  $\text{CS}_{\square}$  rule on two boxed conditions  $\boxed{b_1} \boxed{b_2}_b$  and  $\boxed{b'_1} \boxed{b'_2}_{b'}$ , we know that the sub-proofs of  $b_1 \sim b'_1$  and  $b_2 \sim b'_2$  lie in the fragment  $\mathcal{A}_{\text{CS}_{\square}}$ . This gives us useful information on the shape of the terms. To use this, we define the  $\text{extract}_l$  and  $\text{extract}_r$  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:

$$P|_h = \frac{\vec{w}, b_1, (u_i)_i \sim \vec{w}', b'_1, (u'_i)_i \quad \vec{w}, b_2, (v_i)_i \sim \vec{w}', b'_2, (v'_i)_i}{\vec{w}, (\text{if } \boxed{b_1} \boxed{b_2}_b \text{ then } u_i \text{ else } v_i)_i \sim \vec{w}', (\text{if } \boxed{b'_1} \boxed{b'_2}_{b'} \text{ then } u'_i \text{ else } v'_i)_i} \text{CS}_{\square}$$

We let  $\text{extract}_l(h, P)$  be proof of  $b_1 \sim b'_1$  extracted from  $P|_h$ , and  $\text{extract}_r(h, P)$  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  $P \vdash^b_{\mathcal{A}_{\text{CS}_{\square}}} t \sim t'$ , we say that  $P$  proof in *early proof form* if  $t$  and  $t'$  are of the following form:

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

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  $\text{CS}_{\square}$  rule on the conditions:

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

- Let  $P^{h_l} = \text{extract}_l(h, P)$  and  $P^{h_r} = \text{extract}_r(h, P)$ , then:

$$P^{h_l} \vdash^b_{\mathcal{A}_{\text{CS}_{\square}}} b^{h_l} \sim b'^{h_l} \quad \text{and} \quad P^{h_r} \vdash^b_{\mathcal{A}_{\text{CS}_{\square}}} b^{h_r} \sim b'^{h_r}$$

and these two proofs are in *early proof form*.

- $\text{label}(P^{h_l}) \subseteq \text{label}(P)$ , and for all  $l \in \text{label}(P^{h_l})$ ,  $\text{instance}(P^{h_l}, l)$  is a sub-instance of  $\text{instance}(P, l)$  (same for  $\text{label}(P^{h_r})$ ).
- For all  $l \in \text{label}(P)$ , the proof of  $u_l \sim u'_l$  extracted from  $P$  is in the fragment  $\mathcal{A}_{\overline{\text{BFA}}}$ .

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

**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  $P' \vdash_{\mathcal{A}_{CS_{\square}}}^b t \sim t'$  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  $x \in \{l, r\}$ ,  $\text{extract}_x(h, P)$  is of size strictly smaller than  $P$ . We only need to do some  $\alpha$ -renaming to have the labelling of the sub-proofs coincide.

Finally we can check that the resulting proof  $Q$  is such that for all  $h \in \text{cs-pos}(Q)$ ,  $x \in \{l, r\}$ , for all  $l \in \text{label}(\text{extract}_x(h, P))$ , the  $\text{CCA}_2$  instance  $\text{instance}(\text{extract}_x(h, P), l)$  is a sub-instance of  $\text{instance}(P, l)$ . This follows from the fact that  $\text{extract}_x(h, P)$  is obtained through the Restr elimination procedure from  $P$ .  $\square$

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  $P \vdash_{\mathcal{A}_{CS_{\square}}}^b t \sim t'$  in early proof form.

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

$$\text{index}(P) = \text{cs-pos}(P) \cup \bigcup_{h \in \text{cs-pos}(P)} \text{index}(\text{extract}_l(h, P)) \cup \text{index}(\text{extract}_r(h, P))$$

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

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  $\epsilon$  is the index of the toplevel proof  $P$ .

*Definition 46.* Let  $P \vdash_{\mathcal{A}_{CS_{\square}}}^b t \sim t'$  in early proof form. For all  $l \in \text{label}(P)$ , we define:

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

We abuse the notation and say that  $h \in \text{h-branch}_P(l)$  if there exists  $x \in \{l, r\}$  such that  $h_x \in \text{h-branch}_P(l)$ . 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  $\mathcal{A}_{>}$ , 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  $P \vdash_{\mathcal{A}_{CS_{\square}}}^b t \sim t'$  in early proof form, we have:

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

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  $\mathcal{A}_{\text{BFA}}$ . Therefore, for every  $l$ ,  $u_l$  and  $u'_l$  are of the form:

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

where  $D_l$  is an if-context and:

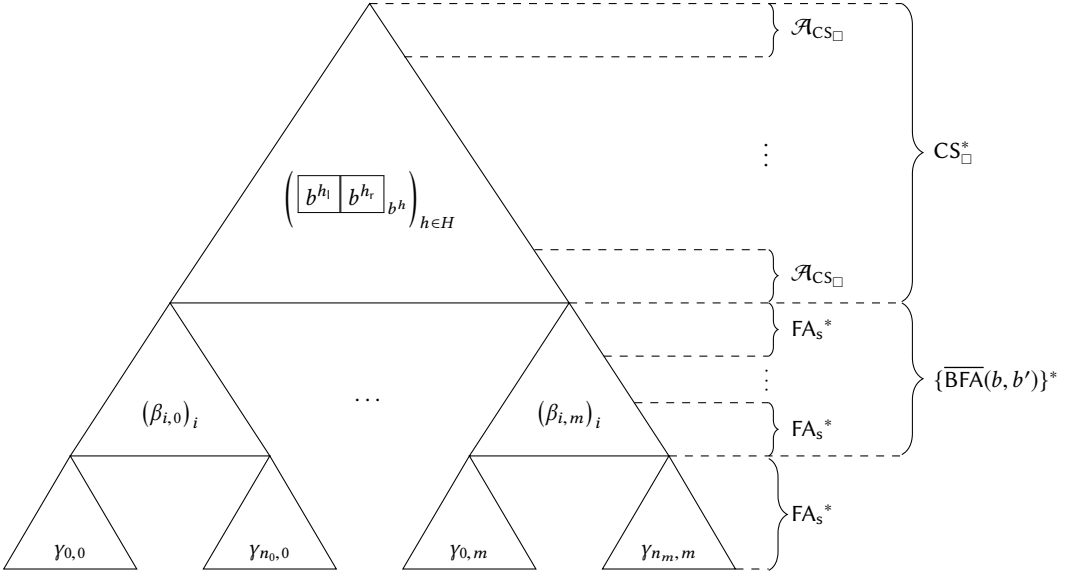


Fig. 13. The shape of the term is determined by the proof.

- $(\beta_{i,l})_{i \in I_l}$  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  $\mathcal{A}_{\text{FA}_s}$ .
- $(\gamma_{j,l})_{j \in M_l}$  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  $\mathcal{A}_{\text{FA}_s}$ .

Using these notation, we give some definitions:

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

- $(b, b') \leq_{CS \sim CS}^{\epsilon, l} (t \sim t', P)$  if and only if there exists  $h_0 \in \text{cs-pos}(P)$  such that  $b \equiv b^{h_0}$  and  $b' \equiv b'^{h_0}$ .
- $(\beta, \beta') \leq_{c \sim c}^{\epsilon, l} (t \sim t', P)$  if and only if there exists  $i \in I_l$  such that  $\beta \equiv \beta_{i,l}$  and  $\beta' \equiv \beta'_{i,l}$ .
- $(\gamma, \gamma') \leq_{l \sim l}^{\epsilon, l} (t \sim t', P)$  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  $P \vdash_{\mathcal{A}_{CS_\square}}^b t \sim t'$  in early proof form and  $L = \text{label}(P)$ . Then:

$$t \equiv C \left[ \_ \diamond (D_l [ (\beta)_{\beta \leq_c^{\epsilon, l} (t, P)} \diamond (\gamma)_{\gamma \leq_l^{\epsilon, l} (t, P)} ] )_{l \in L} \right]$$

$$\text{and } t' \equiv C \left[ \_ \diamond (D_l [ (\beta')_{\beta' \leq_c^{\epsilon, l} (t', P)} \diamond (\gamma')_{\gamma' \leq_l^{\epsilon, l} (t', P)} ] )_{l \in L} \right] \diamond$$

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

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

$$\forall s. s \leq_x^{\epsilon, l} (t, P) \quad \text{if and only if} \quad (s, \_) \leq_{x \sim x}^{\epsilon, l} (t \sim t', P)$$

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



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

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

$$\forall s, s'. (s, s') \leq_{\Delta}^{h_x, l} (t \sim t', P) \quad \text{if and only if} \quad (s, s') \leq_{\Delta}^{\epsilon, l} (b \sim b', \text{extract}_x(h, P))$$

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

$$\forall s. s \leq_{\Delta}^{h_x, l} (t, P) \quad \text{if and only if} \quad s \leq_{\Delta}^{\epsilon, l} (b, \text{extract}_x(h, P))$$

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

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

and  $P' \vdash_{\mathcal{A}_{CS\Box}}^b t_0 \sim t'_0$ , then  $(s, s') \leq_{\Delta}^{h, l} (t \sim t', P)$  if and only if  $(s, s') \leq_{\Delta}^{h, l} (t_0 \sim t'_0, P')$  for any  $\Delta \in \{c \sim c, l \sim l, cs \sim cs\}$ . We have a similar definition for  $\Delta \in \{c, l, cs\}$ .  $\diamond$

### D.3 Proof Form and Normalized Proof Form

*Definition 50.* Let  $P \vdash_{\mathcal{A}_{CS\Box}}^b t \sim t'$  in early proof form and  $L = \text{label}(P)$ . Let  $\mathcal{S}_l$  be the left trace of the  $\text{CCA}_2$  instance used in branch  $l$ , and  $\mathcal{S}'_l$  be the right trace of  $\text{instance}(P, l)$ :

$$\mathcal{S}_l^P = \text{l-trace}(\text{instance}(P, l)) \quad \mathcal{S}'_l^P = \text{r-trace}(\text{instance}(P, l))$$

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

- for every  $h \in \text{cs-pos}(P)$  and  $x \in \{l, r\}$ , the proof  $\text{extract}_x(h, P)$  is in *proof forms*.
- $(\beta, \beta') \leq_{c \sim c}^{\epsilon, l} (t \sim t', P)$ ,  $\beta$  is a  $\mathcal{S}$ -basic term and  $\beta'$  is a  $\mathcal{S}'$ -basic term.
- $(\gamma, \gamma') \leq_{l \sim l}^{\epsilon, l} (t \sim t', P)$ ,  $\gamma$  is a  $\mathcal{S}$ -basic term and  $\gamma'$  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  $P \vdash^{\text{npf}} t \sim t'$  whenever  $P$  is a proof of  $t \sim t'$  in normalized proof form.

Let  $P \vdash^{\text{npf}} t \sim t'$ , we already defined the set of conditions  $\leq_c^{h, l} (t, P)$  used in the  $\overline{\text{BFA}}$  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  $\leq_l^{h, l} (t, P)$  in the sub-proof of  $P$  of at index  $h$  and branch  $l$  is a set of normalized basic terms. Recall that a basic term may contain other basic terms in its subterm. Hence we can define the set of all normalized basic terms appearing in the subterms of  $\leq_c^{h, l} (t, P) \cup \leq_l^{h, l} (t, P)$ .

*Definition 51.* For every  $P \vdash^{\text{npf}} t \sim t'$ , for every term  $s, s' \leq_{\text{bt}}^{h, l} (t, P)$  if and only if there exists  $u (\leq_c^{h, l} \cup \leq_l^{h, l})(t, P)$  such that  $s \leq_{\text{bt}}^{S_l} u$ .

### D.4 Restriction to Proofs in Normalized Proof Form

*Definition 52.* We let  $\overline{\text{CCA}_2}$  be the restriction of  $\text{CCA}_2$  to cases  $\vec{w}, (\alpha_i)_i, (\text{dec}_j)_j \sim \vec{w}', (\alpha'_i)_i, (\text{dec}'_j)_j$  where:

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

LEMMA 15. *The following strategy is complete for  $\mathfrak{F}((CS + FA + R + Dup + \text{CCA}_2)^*)$ :*

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

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

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

For every proof  $P \vdash^b t \sim t'$  in this fragment, we let  $L^P = \text{label}(P)$  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  $\text{CCA}_2$  instance of branch  $l$ , i.e.  $\mathcal{S}_l^P = \text{l-trace}(\text{instance}(P, l))$ . Finally, we define the order  $<_l^P$  as follows: for all  $u, u' \in \mathcal{E}_l^P \cup \mathcal{D}_l^P$ , we let  $u <_l^P u'$  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  $l \in \text{label}(Q)$ ,  $\mathcal{E}_l^Q$  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  $<_l^P$  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  $u \equiv \{m\}_{\text{pk}}^{\text{nr}}$  where the corresponding secret key  $\text{sk}$  is in  $\mathcal{K}_l^P$ . Let  $(\alpha_k)_k$  be  $\mathcal{E}_l^P \cap \text{st}(m)$ , and  $(\text{dec}_n)_n$  be  $\mathcal{D}_l^P \cap \text{st}(m)$ . Let  $C$  be the smallest context such that:

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

From the definition of  $\text{CCA}_2$ , 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:

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

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  $(\text{dec}_n)_n$  are  $\mathcal{S}_l^P$ -decryption oracle calls. For every  $k$ ,  $\alpha_k$  is of the shape  $\alpha_k \equiv \{\_ \}_{\text{pk}_k}^{\text{nk}}$ . For every  $n$ , we let  $\text{sk}_n$  be the secret key used in  $\text{dec}_n$ . Assume that there is some  $i$  such that:

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

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  $\text{dec}(\{[\ ]_k\}_{\text{pk}_k}^{\text{nk}}, \text{sk}_k)$  is a subterm of  $\tilde{m}$ . By consequence,  $\text{sk}_k$  is a subterm of  $B_i[\ ]$ . But since  $\text{sk}_k \in \mathcal{K}_l^P$ , we know that  $\text{st}(B_i)$  does not contain  $\text{sk}_k$  ( $\text{sk}_k$  can only appear in  $\mathcal{D}_l^P$ ). Contradiction. Hence  $\tilde{m}$  is in  $R$ -normal form, which implies that  $(B_i[(\alpha_k)_k, (\text{dec}_n)_n])_i$  are  $\mathcal{S}_l^P$ -normalized basic terms. Similarly we prove that  $(U_m[(\alpha_k)_k, (\text{dec}_n)_n])_m$  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  $\{m\}_{\text{sk}}^{\text{nr}}$  by  $\{m_0\}_{\text{sk}}^{\text{nr}}$  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  $\text{CCA}_2$  application for the decryptions still holds. Their is one subtlety here: an encryption  $\alpha \equiv \{m_\alpha\}_{\text{pk}}^{\text{nr}}$  must be guarded in some  $\text{dec}(u_0, \text{sk})$  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  $\{m\}_{\text{sk}}^{\text{nr}}$  into  $\{m_0\}_{\text{sk}}^{\text{nr}}$ ).

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  $l' \in L_P \setminus \{l\}$  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.  $\square$

We are now ready to prove that  $\vdash^{n\text{pf}}$  is complete.

LEMMA 16. *The restriction of the fragment  $\mathcal{A}_>$  to formulas provable in  $\vdash^{n\text{pf}}$  is complete for:*

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

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

$$\mathfrak{F}((2\text{Box} + R_{\square})^* \cdot CS_{\square}^* \cdot \{\overline{\text{BFA}}(b, b')\}^* \cdot \text{UnF} \cdot \text{FA}_s^* \cdot \text{Dup}^* \cdot \overline{CCA_2})$$

First we show that this strategy remains complete even if with restrict it to proofs such that the terms after  $(2\text{Box} + R_{\square})^*$  are in proof form. Let  $\vdash_{\mathcal{A}_{CS_{\square}}} t \sim t'$ , we want to find  $t_0 =_R t, t'_0 =_R t'$  and  $P'$  such that  $P' \vdash^{n\text{pf}} t \sim t'$ .

By Proposition 14, we know that there exists  $P$  such that  $P \vdash_{\mathcal{A}_{CS_{\square}}}^b t \sim t'$ . Let  $h \in \text{index}(P), x \in \{l, r\}, h = h_x$ , and let  $b^h, b'^h$  be such that  $\text{extract}_x(h, P) \vdash_{\mathcal{A}_{CS_{\square}}}^b b^h \sim b'^h$ . First, we prove that we can ensure that for every  $(\beta, \beta') (\leq_{\overline{c} \sim c}^{h, l} \cup \leq_{[-, -]}^{h, l})(t \sim t', P)$ , the terms  $\beta$  and  $\beta'$  are, respectively,  $\mathcal{S}_l^P$ -basic term and  $\mathcal{S}'_l^P$ -basic terms. We know that:

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

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

Since this is a sub-instance, we know that  $\text{fresh}(\mathcal{R}_l^P; \vec{w})$  and  $\text{nodedc}(\mathcal{K}_l^P, \vec{w})$ . Moreover, using the fact that instance  $(P, l)$  is a  $\overline{CCA_2}$  instance, we know that  $(\alpha_j)_j$  and  $(\text{dec}_k)_k$  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  $\beta$  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:

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

Let  $t_0$  be the term obtained from  $t$  by replacing this occurrence of  $\beta$  by:

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

Similarly we define  $t'_0$  by replacing  $\beta'$  by the corresponding term. Then  $t_0 =_R t$  and  $t'_0 =_R t'$ . Moreover it is easy to check that the formula  $t_0 \sim t'_0$  is provable in  $\vdash_{\mathcal{A}_{CS_{\square}}}^b$ , as we replaced one  $\overline{\text{BFA}}$  application by three  $\overline{\text{BFA}}$  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.  $\square$

## 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):

$$C \equiv \begin{array}{c} a \\ \swarrow \quad \searrow \\ \left( \begin{array}{c} []_x \\ \swarrow \quad \searrow \\ c \quad d \end{array} \right) \quad t_3 \\ \swarrow \quad \searrow \\ t_0 \quad []_y \\ \swarrow \quad \searrow \\ t_1 \quad t_2 \end{array} \quad \text{and} \quad C' \equiv \begin{array}{c} a \\ \swarrow \quad \searrow \\ \left( \begin{array}{c} []_x \\ \swarrow \quad \searrow \\ c \quad d \end{array} \right) \quad t_3 \\ \swarrow \quad \searrow \\ t_0 \quad b \\ \swarrow \quad \searrow \\ []_y \quad []_z \end{array} \quad \diamond$$

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:*

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

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

$$\begin{aligned} \beta_0 &\equiv \text{eq}(\{\text{if } b \text{ then } s \text{ else } t\}_{pk(n)}^{nr}, 0) & \beta_1 &\equiv \text{eq}(\{\text{if } \beta_0^0 \text{ then } u \text{ else } u\}_{pk(n)}^{nr}, 0) \\ &=_R \text{if } b \text{ then } \underbrace{\text{eq}(\{s\}_{pk(n)}^{nr}, 0)}_{\beta_0^0} \text{ else } \underbrace{\text{eq}(\{t\}_{pk(n)}^{nr}, 0)}_{\beta_0^1} \end{aligned}$$

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  $\beta_0^0$  disappear using the rule  $\text{if } x \text{ then } y \text{ else } y \rightarrow y$  in  $R$ . Hence, Lemma 17 could choose  $\tilde{\beta}'_1 \equiv \beta_1$ . Of course this situation cannot occur, as we cannot have  $\beta_0^0$  be a subterm of  $\beta_1$  (this contradicts the freshness side-condition of encryptions' randomnesses in the  $\text{CCA}_2$  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:

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

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

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

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

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

*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:

$$\text{leave-st}\left(\left(u \begin{array}{c} \diagup \\ b \\ \diagdown \\ v \end{array} \right) \downarrow_R\right) = \{u, w\} \quad \overline{\text{leave-st}}\left(u \begin{array}{c} \diagup \\ b \\ \diagdown \\ v \end{array} \right) = \{u, v, w\}$$

There in another over-approximation in the  $f$  case:

$$\text{leave-st}\left(f\left(u \begin{array}{c} \diagup \\ b \\ \diagdown \\ v \end{array}, s \begin{array}{c} \diagup \\ b \\ \diagdown \\ t \end{array}\right) \downarrow_R\right) = \{f(u, s), f(v, t)\}$$

$$\overline{\text{leave-st}}\left(f\left(u \begin{array}{c} \diagup \\ b \\ \diagdown \\ v \end{array}, s \begin{array}{c} \diagup \\ b \\ \diagdown \\ t \end{array}\right)\right) = \{f(u, s), f(u, t), f(v, s), f(v, t)\}$$

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

$$\text{cond-st}\left(u \begin{array}{c} \diagup \\ b \\ \diagdown \\ u \end{array} \downarrow_R\right) = \emptyset \quad \overline{\text{cond-st}}\left(u \begin{array}{c} \diagup \\ b \\ \diagdown \\ u \end{array} \right) = \{b\} \quad \diamond$$

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

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

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

**PROOF.** The facts that  $\overline{\text{leave-st}}(u \downarrow_R) = \text{leave-st}(u \downarrow_R)$  and  $\overline{\text{cond-st}}(u \downarrow_R) = \text{cond-st}(u \downarrow_R)$  are straightforward to show. Let us prove by induction on  $\rightarrow_R^*$  that for all  $u \rightarrow_R^* u'$ ,  $\overline{\text{leave-st}}(u) \supseteq \overline{\text{leave-st}}(u')$ . If  $u \equiv u'$  this is immediate, assume that  $u \rightarrow_R v \rightarrow_R^* u'$ . By induction hypothesis we know that  $\overline{\text{leave-st}}(v) \supseteq \overline{\text{leave-st}}(u')$ . Therefore, we only need to show that  $\overline{\text{leave-st}}(u) \supseteq \overline{\text{leave-st}}(v)$ . 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:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Let us show the following helpful propositions:

**PROPOSITION 16.** For all  $\mathcal{S}_1$ -normalized basic terms  $\beta, \beta'$  if:

$$\overline{\text{leave-st}}(\beta) \cap \overline{\text{leave-st}}(\beta') \neq \emptyset$$

then we have  $\mathcal{S}_1$ -normalized basic terms  $B[\vec{w}, (\alpha^j)_j, (\delta^k)_k]$  and  $B[\vec{w}, (\alpha'^j)_j, (\delta'^k)_k]$  such that:

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

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

**PROOF.** We have  $\mathcal{S}_1$ -normalized basic terms  $B[\vec{w}, (\alpha^j)_j, (\delta^k)_k]$  and  $D[\vec{w}', (\alpha'^j)_j, (\delta'^k)_k]$  such that:

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

Since  $\beta, \beta'$  are  $\mathcal{S}_1$ -normalized basic terms, we know that:

$$B[\vec{w}, (\{\square_j\}_-)_j, (\text{dec}(\square_k, \_))_k] \quad D[\vec{w}', (\{\square_j\}_-)_j, (\text{dec}(\square_k, \_))_k]$$

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:

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

Similarly to what we did in the proof of Lemma 2, we prove that we can assume that  $B[] \equiv D[]$  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:

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

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:

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

and:

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

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  $\vec{w}'$  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  $\{m\}_{\text{pk}(n_p)}^{n_r}$ . Since  $\{m\}_{\text{pk}(n_p)}^{n_r} \in \mathcal{E}_I$  is an encryption in an instance of a  $\text{CCA}_2$  application, we know from the freshness side-condition that  $n_r$  does not appear in  $\vec{w}$  and that  $n_r \in \mathcal{R}_I$ . Moreover since  $\beta'$  is a  $\mathcal{S}_I$ -normalized basic term, we know that  $\text{fresh}(\mathcal{R}_I; \vec{w}')$ . But since  $p$  is a valid non-hole position in  $D$ , we have  $n_r \in \vec{w}'$ . Absurd.
- Similarly if  $B$  contains a hole  $[\ ]_x$  at position  $p$  such that  $\beta|_p$  is a decryption oracle call  $\text{dec}(m, \text{sk}(n))$ . Since  $\text{dec}(m, \text{sk}(n))$  is a decryption oracle call we know that  $\text{sk}(n) \in \mathcal{K}_I$ . Moreover since  $\beta'$  is a  $\mathcal{S}_I$ -normalized basic term, we know that  $\text{nodec}(\mathcal{K}_I, \vec{w}')$ . But since  $p$  is a valid non-hole position in  $D$ , we know that either  $\text{sk}(n) \in \vec{w}'$  or  $n \in \vec{w}'$ . Absurd.  $\square$

We can now state the following proposition, which subsumes Proposition 8.

**PROPOSITION 17.** *For all  $\mathcal{S}_I$ -normalized basic terms  $\beta, \beta'$ , we have  $\beta \equiv \beta'$  whenever:*

$$\overline{\text{leave-st}}(\beta) \cap \overline{\text{leave-st}}(\beta') \neq \emptyset$$

**PROOF.** We show this by induction on  $|\beta| + |\beta'|$ . Using Proposition 16 we know that we have  $\mathcal{S}_I$ -normalized basic terms  $B[\vec{w}, (a^j)_j, (d^k)_k], B[\vec{w}, (a'^j)_j, (d'^k)_k]$  such that:

$$\beta \equiv B[\vec{w}, (a^j)_j, (d^k)_k] \quad \beta' \equiv B[\vec{w}, (a'^j)_j, (d'^k)_k]$$

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

To conclude we only need to show that for all  $j$ ,  $\overline{\text{leave-st}}(\alpha^j) \cap \overline{\text{leave-st}}(\alpha'^j) \neq \emptyset$  implies that  $\alpha^j \equiv \alpha'^j$  and that  $\overline{\text{leave-st}}(\delta^k) \cap \overline{\text{leave-st}}(\delta'^k) \neq \emptyset$  implies that  $\delta^k \equiv \delta'^k$ . The former is immediate, as  $\overline{\text{leave-st}}(\alpha^j) \cap \overline{\text{leave-st}}(\alpha'^j) \neq \emptyset$  implies that  $\alpha^j \equiv \{m\}_{\text{pk}(n)}^{n_r}$  and  $\alpha'^j \equiv \{m'\}_{\text{pk}(n)}$ . Since  $\alpha^j, \alpha'^j \in \mathcal{E}_I$

and since there is *as most one*  $\mathcal{S}_I$ -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  $\delta^k, \delta'^k$  are  $\mathcal{S}_I$ -decryption oracle calls we know that

$$\delta^k \equiv C [\vec{g} \diamond (s_i)_{i \leq p}] \quad \delta'^k \equiv C' [\vec{g}' \diamond (s'_i)_{i \leq p'}]$$

where:

- There exists contexts  $u, u'$ , if-free and in  $R$ -normal form, such that:

$$\forall i < p, s_i \equiv \mathbf{0}(\text{dec}(u[(\alpha_j)_j], (\text{dec}_k)_k], \text{sk})) \quad s_p \equiv \text{dec}(u[(\alpha_j)_j], (\text{dec}_k)_k], \text{sk})$$

$$\forall g \in \vec{g}, g \equiv \text{eq}(u[(\alpha_j)_j], (\text{dec}_k)_k], \alpha_g) \text{ where } \alpha_g \in (\alpha_j)_j$$

$$\forall i < p', s'_i \equiv \mathbf{0}(\text{dec}(u'[(\alpha'_j)_j], (\text{dec}'_k)_k], \text{sk}')) \quad s'_p \equiv \text{dec}(u'[(\alpha'_j)_j], (\text{dec}'_k)_k], \text{sk}')$$

$$\forall g \in \vec{g}', g \equiv \text{eq}(u'[(\alpha'_j)_j], (\text{dec}'_k)_k], \alpha'_g) \text{ where } \alpha'_g \in (\alpha'_j)_j$$

- $(\alpha_j)_j, (\alpha'_j)_j$  are  $\mathcal{S}_I$ -encryption oracle calls.
- $(\text{dec}_k)_k, (\text{dec}'_k)_k$  are  $\mathcal{S}_I$ -decryption oracle call.

Since  $\overline{\text{leave-st}}(\delta^k) \cap \overline{\text{leave-st}}(\delta'^k) \neq \emptyset$ , and since  $u, u'$  are if-free and in  $R$ -normal form we know that  $u \equiv u'$ , for all  $j$ ,  $\overline{\text{leave-st}}(\alpha_j) \cap \overline{\text{leave-st}}(\alpha'_j)$  and for all  $k$ ,  $\overline{\text{leave-st}}(\text{dec}_k) \cap \overline{\text{leave-st}}(\text{dec}'_k)$ . It follows, by induction hypothesis, that for all  $j$ ,  $\alpha_j \equiv \alpha'_j$  and for all  $k$ ,  $\text{dec}_k \equiv \text{dec}'_k$ . We only have to check that the guards are the same. Since  $\delta^k, \delta'^k \in \mathcal{D}_I$ , we know from the definition of the  $\text{CCA}_2$  axioms that  $\delta^k$  (resp.  $\delta'^k$ ) has one guard for every encryption  $\alpha \in \mathcal{E}_I$  such that  $\alpha \equiv \{\_ \}_{\text{pk}}^n$  and  $n$  appear directly in  $s_p$  (resp.  $s'_p$ ). Since we showed that  $s_p \equiv s'_p$ , we deduce that  $\delta^k, \delta'^k$  have the same guards. Since guards are sorted according to an arbitrary but fixed order (the sort function in the definition of  $R_{\text{CCA}_2}^K$ ), we know that  $\delta^k \equiv \delta'^k$ .  $\square$

COROLLARY 2. For all  $P \vdash^{\text{npf}} t \sim t'$ , for all  $h, l$ :

- for all  $\beta, \beta' \leq_c^{h,l}(t, P)$  if  $\text{leave-st}(\beta \downarrow_R) \cap \text{leave-st}(\beta' \downarrow_R) \neq \emptyset$  then  $\beta \equiv \beta'$ .
- for all  $\gamma, \gamma' \leq_l^{h,l}(t, P)$  if  $\text{leave-st}(\gamma \downarrow_R) \cap \text{leave-st}(\gamma' \downarrow_R) \neq \emptyset$  then  $\gamma \equiv \gamma'$ .
- for all  $\beta \leq_c^{h,l}(t, P), \gamma \leq_l^{h,l}(t, P)$  if  $\text{leave-st}(\beta \downarrow_R) \cap \text{leave-st}(\gamma \downarrow_R) \neq \emptyset$  then  $\beta \equiv \gamma$ .

We can now show the following lemma, which subsumes Lemma 17:

LEMMA 18. For all  $P \vdash^{\text{npf}} t \sim t'$ , for all  $h, l$  and  $\beta, \beta' \leq_{bt}^{h,l}(t, P)$ , there exists an almost condition context  $\tilde{\beta}'[]$  such that:

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

PROOF. Let  $l \in \text{label}(P)$ . We prove by mutual induction on the definition of  $\mathcal{S}_I$ -normalized simple terms,  $\mathcal{S}_I$ -normalized basic terms,  $\mathcal{S}_I$ -encryption oracle calls and  $\mathcal{S}_I$ -decryption oracle calls that for every  $u \in \text{st}(\beta')$  such that  $u$  is in one of the four above cases, there exists a condition context  $u_c[]$  such that:

$$u \equiv u_c[\beta] \quad \text{leave-st}(\beta \downarrow_R) \cap \overline{\text{cond-st}}(u_c[]) = \emptyset \quad \overline{\text{leave-st}}(\vec{u}_c) = \overline{\text{leave-st}}(\vec{u})$$

Moreover if  $u$  is a  $\mathcal{S}_I$ -normalized basic term then there exists an almost condition context  $u_d[]$  such that:

$$u \equiv u_d[\beta] \quad \text{leave-st}(\beta \downarrow_R) \cap \left( \overline{\text{cond-st}}(u_d[]) \cup \overline{\text{leave-st}}(u_d[]) \right) = \emptyset$$



- **Normalized Simple Term:** Let  $u \equiv C[\vec{b} \diamond \vec{s}]$ , where  $\vec{b}$  are  $\mathcal{S}_I$ -normalized basic conditions and  $\vec{s}$  are  $\mathcal{S}_I$ -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:

$$\overline{\text{leave-st}(\vec{s}_c[]) = \overline{\text{leave-st}(\vec{s})} \quad \text{leave-st}(\beta \downarrow_R) \cap \left( \overline{\text{cond-st}(\vec{b}_d[], \vec{s}_c[]) \cup \overline{\text{leave-st}(\vec{b}_d[])}} \right) = \emptyset$$

Moreover:

$$\begin{aligned} \overline{\text{cond-st}(C[\vec{b}_d[] \diamond \vec{s}_c[]])} &= \overline{\text{cond-st}(\vec{b}_d[], \vec{s}_c[]) \cup \overline{\text{leave-st}(\vec{b}_d[])}} \\ \overline{\text{leave-st}(C[\vec{b}_d[] \diamond \vec{s}_c[]])} &= \overline{\text{leave-st}(\vec{s}_c[]) = \overline{\text{leave-st}(\vec{s})} = \overline{\text{leave-st}(C[\vec{b} \diamond \vec{s}])} \end{aligned}$$

Hence we can take  $\vec{u}_c \equiv C[\vec{b}_d[] \diamond \vec{s}_c[]]$ .

- **Normalized Basic Term:** Let  $u \equiv B[\vec{w}, (\alpha^i)_i, (\text{dec}^j)_j]$  be a  $\mathcal{S}_I$ -normalized basic term. Let  $(\alpha^i)_i, (\alpha^i)_i$  and  $(\text{dec}^j)_j, (\text{dec}^j)_j$  be the contexts obtained by applying the induction hypothesis to  $(\alpha^i)_i$  and  $(\text{dec}^j)_j$ . Using the fact that:

$$\overline{\text{leave-st}((\alpha^i)_i, (\text{dec}^j)_j)} = \overline{\text{leave-st}((\alpha^i)_i, (\text{dec}^j)_j)}$$

and since  $B$  and  $\vec{w}$  are if-free, one can check that:

$$\overline{\text{leave-st}(B[\vec{w}, (\alpha^i)_i, (\text{dec}^j)_j])} = \overline{\text{leave-st}(B[\vec{w}, (\alpha^i)_i, (\text{dec}^j)_j])}$$

It is then immediate to check that  $u_c \equiv B[\vec{w}, (\alpha^i)_i, (\text{dec}^j)_j]$  satisfies the wanted properties. It remains to construct the context  $u_d[]$ . If  $\text{leave-st}(\beta \downarrow_R) \cap \overline{\text{leave-st}(u)} = \emptyset$  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.  $\square$

## 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  $(C, \mathcal{D})$  of flat terms such that:

- (a)  $(C, \mathcal{D})$  is well-nested if for every  $C_0[\vec{a}_0 \diamond \vec{b}_0] \in C$ :

$$\forall C[\vec{a} \diamond \vec{b}] \in C \cup \mathcal{D}, \quad \vec{b}_0 \cap \vec{a} = \emptyset$$

- (b)  $(C, \mathcal{D})$  is well-nested if for every  $\beta_0 \equiv C_0[\vec{a}_0 \diamond \vec{b}_0] \in C$ :

- (i) For all  $\beta \equiv C[\vec{a} \diamond \vec{b}] \in C \cup \mathcal{D}$ , there exist two if-contexts  $C'_\beta, C''_\beta$  such that:

$$\beta =_R \text{ if } \beta_0 \text{ then } C'_\beta[\vec{a}'_\beta \diamond \vec{b}'_\beta] \text{ else } C''_\beta[\vec{a}''_\beta \diamond \vec{b}''_\beta]$$

where  $\vec{a}'_\beta, \vec{a}''_\beta \subseteq \vec{a} \setminus \vec{b}_0$  and  $\vec{b}'_\beta, \vec{b}''_\beta \subseteq \vec{b}$ .

- (ii) The following couples of sets are well-nested:

$$\begin{aligned} &\left\{ C'_\beta[\vec{a}'_\beta \diamond \vec{b}'_\beta] \mid C[\vec{a} \diamond \vec{b}] \in C \right\}, \left\{ C'_\beta[\vec{a}'_\beta \diamond \vec{b}'_\beta] \mid C[\vec{a} \diamond \vec{b}] \in \mathcal{D} \right\} \\ &\left\{ C''_\beta[\vec{a}''_\beta \diamond \vec{b}''_\beta] \mid C[\vec{a} \diamond \vec{b}] \in C \right\}, \left\{ C''_\beta[\vec{a}''_\beta \diamond \vec{b}''_\beta] \mid C[\vec{a} \diamond \vec{b}] \in \mathcal{D} \right\} \end{aligned}$$

**PROPOSITION 18.** *If  $(C, \mathcal{D})$  verifies the property (a) above, then it satisfies properties (i) and (ii).*

**PROOF.** Trivial by taking  $C'_\beta \equiv C''_\beta \equiv C$ .  $\square$

*Definition 57.* We let  $\text{head}$  be the partial function defined on terms such that for all  $f \in \mathcal{F}$ , for all terms  $\vec{t}$ ,  $\text{head}(f(\vec{t})) \equiv f$ .

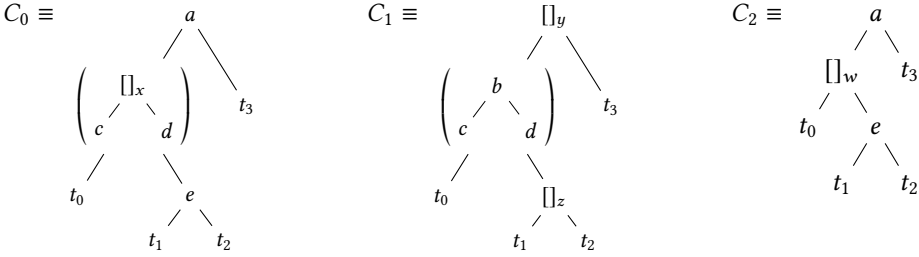
*Definition 58.* For all condition contexts  $C_0, C_1$ , we let  $C_0 \sqcup_c C_1$  be the condition context, if it exists, defined as follows:  $\text{pos}(C_0 \sqcup_c C_1) = \text{pos}(C_0) \cap \text{pos}(C_1)$  and for all position  $p$  in  $\text{pos}(C_0 \sqcup_c C_1)$ :

$$(C_0 \sqcup_c C_1)|_p \equiv \begin{cases} a & \text{if } \text{head}((C_0)|_p) \equiv \text{head}((C_1)|_p) \equiv a \quad (a \in \mathcal{F} \cup \mathcal{N}) \\ \square_x & \text{if } (C_0)|_p \equiv \square_x \wedge (\text{head}((C_1)|_p) \equiv \square_x \vee \text{head}((C_1)|_p) \equiv a) \quad (a \in \mathcal{F} \cup \mathcal{N}) \\ \square_x & \text{if } (C_1)|_p \equiv \square_x \wedge (\text{head}((C_0)|_p) \equiv \square_x \vee \text{head}((C_0)|_p) \equiv a) \quad (a \in \mathcal{F} \cup \mathcal{N}) \end{cases}$$

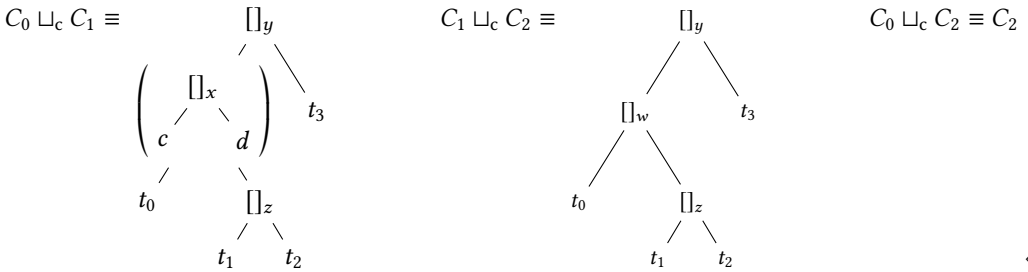
If such a condition context does not exist, then  $C_0 \sqcup_c C_1$  is the special element undefined. We also let:

$$\text{undefined} \sqcup_c C_0 \equiv C_0 \sqcup_c \text{undefined} \equiv \text{undefined}$$

*Example 20.* For all conditions  $a, b, c, d, e, f$  and terms  $t_0, \dots, t_3$ , if we let:



Then we have:

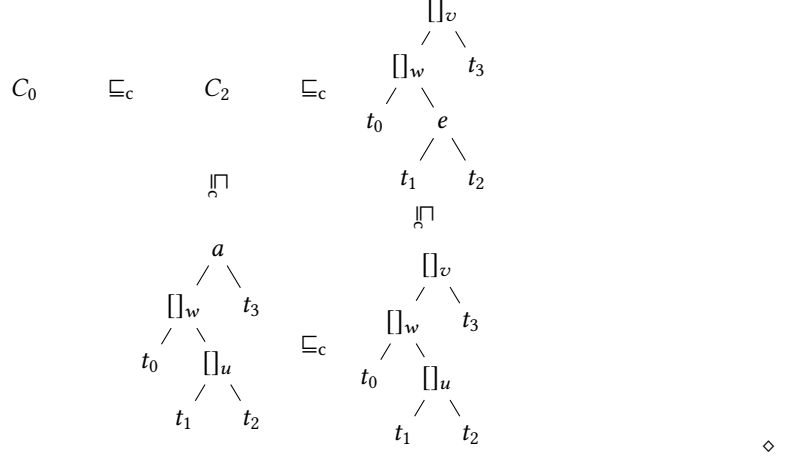


*Definition 59.* We let  $\sqsubseteq_c$  be the relation on condition contexts defined as follows: for all condition contexts  $C_0, C_1$ , we let  $C_0 \sqsubseteq_c C_1$  hold if  $\text{pos}(C_1) \subseteq \text{pos}(C_0)$  and for all position  $p$  in  $\text{pos}(C_1)$ :

$$\text{if } \text{head}((C_1)|_p) \equiv \begin{cases} a & \text{then } \text{head}((C_0)|_p) \equiv a \quad \text{where } (a \in \mathcal{F} \cup \mathcal{N}) \\ \square_x & \text{then } \text{head}((C_0)|_p) \in \mathcal{F} \cup \mathcal{N} \cup \{\square_x\} \end{cases}$$

Moreover we let  $C_0 \sqsubseteq_c \text{undefined}$  for all condition context  $C_0$  (and  $\text{undefined} \sqsubseteq_c \text{undefined}$ ).

*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:

$$C_0 \sqsubseteq_c C \sqsubseteq_c C_0 \sqcup_c C_1 \qquad C_1 \sqsubseteq_c C \sqsubseteq_c C_0 \sqcup_c C_1$$

If  $C_0 \sqcup_c C_1 \equiv \text{undefined}$  then one can check that  $C \equiv \text{undefined}$ . Otherwise we know that  $\text{pos}(C_0 \sqcup_c C_1) = \text{pos}(C_0) \cap \text{pos}(C_1)$ , and that:

$$\text{pos}(C_0) \supseteq \text{pos}(C) \supseteq \text{pos}(C_0 \sqcup_c C_1) \qquad \text{pos}(C_1) \supseteq \text{pos}(C) \supseteq \text{pos}(C_0 \sqcup_c C_1)$$

Hence  $\text{pos}(C) = \text{pos}(C_0 \sqcup_c C_1)$ . Using the fact that  $C \sqsubseteq_c C_0 \sqcup_c C_1$  we know that for all position  $p \in \text{pos}(C)$ , if  $\text{head}((C_0 \sqcup_c C_1)|_p) = a$  (with  $a \in \mathcal{F} \cup \mathcal{N}$ ) then  $\text{head}(C|_p) = a$ . If  $\text{head}((C_0 \sqcup_c C_1)|_p) = []_x$  then either  $\text{head}(C|_p) = []_x$  or  $\text{head}(C|_p) = a$  (with  $a \in \mathcal{F} \cup \mathcal{N}$ ). In the former case there is nothing to show, in the latter case observe that  $\text{head}((C_0 \sqcup_c C_1)|_p) = []_x$  implies that either  $\text{head}((C_0)|_p) = []_x$  or  $\text{head}((C_1)|_p) = []_x$ . W.l.o.g. assume  $\text{head}((C_0)|_p) = []_x$ . Then using the fact that  $C_0 \sqsubseteq_c C$ , we know that  $\text{head}((C_0)|_p) = []_x$  implies that  $\text{head}((C)|_p) = []_x$ . Absurd.

Therefore  $\forall p \in \text{pos}(C)$ ,  $\text{head}(C|_p) = \text{head}((C_0 \sqcup_c C_1)|_p)$ . Moreover  $\text{pos}(C) = \text{pos}(C_0 \sqcup_c C_1)$ . Hence  $C \equiv C_0 \sqcup_c C_1$ .  $\square$

Let  $C_0, C_1 \in \mathcal{S}_{cc}$  such that  $C_0 \sqsubseteq_c C_1$ . Moreover, assume that:

$$\forall p, p' \in \text{pos}(C_1), (C_1)|_p \equiv (C_1)|_{p'} \equiv []_x \Rightarrow (C_0)|_p \equiv (C_0)|_{p'}$$

Then, we know that  $C_0$  and  $C_1$  coincides on  $\text{pos}(C_1)$ . Therefore, any  $\rightarrow_R$  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.  $\text{cond-st}(C'_1 \downarrow_R) \cap \mathcal{T}(\mathcal{F}_{\text{if}}, \mathcal{N})$ ) all appear directly as subterm of  $C'_0$ , hence are in  $\text{cond-st}(C'_0)$ .

PROPOSITION 20. For every  $C_0, C_1 \in \mathcal{S}_{cc}$ , if  $C_0 \sqsubseteq_c C_1$  and if:

$$\forall p, p' \in \text{pos}(C_1), (C_1)|_p \equiv (C_1)|_{p'} \equiv \prod_x \Rightarrow (C_0)|_p \equiv (C_0)|_{p'}$$

then  $\text{cond-st}(C_1 \downarrow_R) \cap \mathcal{T}(\mathcal{F}_{\setminus \text{if}}, \mathcal{N}) \subseteq \overline{\text{cond-st}(C_0)}$ .

PROOF. Assume that  $C_0 \sqsubseteq_c C_1$ , with  $C_0, C_1 \neq \text{undefined}$  (the cases  $C_0 = \text{undefined}$  and  $C_1 = \text{undefined}$  are easy to handle, with the convention that  $\text{cond-st}(\text{undefined}) = \emptyset$ ), and that:

$$\forall p, p' \in \text{pos}(C_1), (C_1)|_p \equiv (C_1)|_{p'} \equiv \prod_x \Rightarrow (C_0)|_p \equiv (C_0)|_{p'} \quad (9)$$

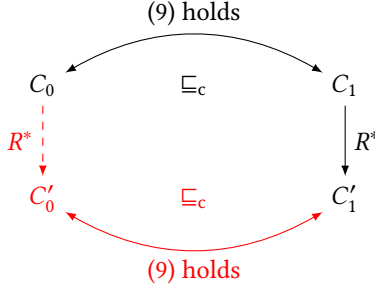
First we show that we can extend this property as follows:

$$\forall p, p' \in \text{pos}(C_1), (C_1)|_p \equiv (C_1)|_{p'} \Rightarrow (C_0)|_p \equiv (C_0)|_{p'} \quad (10)$$

Let  $q = p \cdot q_0$  and  $q' = p' \cdot q_0$  be positions in  $\text{pos}(C_1)$ . Since  $(C_1)|_p \equiv (C_1)|_{p'}$ , we know that  $\text{head}((C_1)|_q) \equiv \text{head}((C_1)|_{q'})$ .

- If  $\text{head}((C_1)|_q) \equiv a$  (with  $a \in \mathcal{F} \cup \mathcal{N}$ ) then, from the fact that  $C_0 \sqsubseteq_c C_1$  we get that  $\text{head}((C_0)|_q) \equiv a$ , and that  $\text{head}((C_0)|_{q'}) \equiv a$ .
- If  $\text{head}((C_1)|_q) \equiv \prod_x$  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  $C_1 \xrightarrow*_R C'_1$ , there exists  $C'_0$  such that  $C'_0 \sqsubseteq_c C'_1$ , (9) holds for  $C'_0, C'_1$  and  $C_0 \xrightarrow*_R C'_0$ . Graphically (hypothesis are in black, goals are in red):



Let  $\rightarrow_{R'}$  be  $\rightarrow_R$  without the non left-linear rules, which are:

$$\text{if } x \text{ then } y \text{ else } y \rightarrow y \quad \text{dec}(\{x\}_{\text{pk}(y)}^r, \text{sk}(y)) \rightarrow x$$

$$\text{if } w \text{ then (if } w \text{ then } x \text{ else } y) \text{ else } z \rightarrow \text{if } w \text{ then } x \text{ else } z$$

$$\text{if } w \text{ then } x \text{ else (if } w \text{ then } y \text{ else } z) \rightarrow \text{if } w \text{ then } x \text{ else } z$$

We mimic all reduction  $\rightarrow_R$  on  $C_1$  by a reduction on  $C_0$ , while maintaining  $\sqsubseteq_c$  and the invariant of (9). Mimicking rules in  $\rightarrow_R$  is easy as they are left-linear. To mimic rules in  $(\rightarrow_R \setminus \rightarrow_{R'})$ , we use (10).

Therefore let  $C'_1$  be in  $R$ -normal form such that  $C_1 \xrightarrow*_R C'_1$ . Let  $C'_0$  be such that  $C'_0 \sqsubseteq_c C'_1$ , (9) holds for  $C'_0, C'_1$  and  $C_0 \xrightarrow*_R C'_0$ .  $C'_1$  is of the form  $D[\vec{b}, \vec{b}_\square \diamond \vec{u}]$  where  $\vec{b}, \vec{u}$  are if-free and in  $R$ -normal form,  $\vec{b}$  does not contain any hole variable and  $\vec{b}_\square$  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.  $\square$

LEMMA 19. For all  $P \vdash^{npf} t \sim t'$ , for all  $h, l$ , the following couple of sets is well-nested:

$$\left( \{\beta \downarrow_R \mid \beta \leq_c^{h,l} (t, P)\}, \{\gamma \downarrow_R \mid \gamma \leq_l^{h,l} (t, P)\} \right)$$

PROOF. We do this proof in the case  $h = \epsilon$ . The other cases are identical.

Part 1. We consider an arbitrary ordering  $(\beta_i)_{1 \leq i \leq i_{\max}}$  of:

$$\{\beta \mid \beta \leq_c^{h,l} (t, P)\}$$

Using Lemma 18, we know that all  $i \neq i_0$ , there exists a condition context  $\tilde{\beta}_i$  such that:

$$\beta_i \equiv \tilde{\beta}_i [\beta_{i_0}] \quad \text{and} \quad \text{leave-st}(\beta_{i_0} \downarrow_R) \cap \overline{\text{cond-st}(\tilde{\beta}_i)} = \emptyset$$

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  $l_0, \dots, l_n$  be a sequence of distinct indices in  $\{0, \dots, i_{\max}\}$  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:

$$\beta_i^{(l_{j_0})} \sqsubseteq_c \sqcup_{c_j \leq n} \beta_i^{(l_j)}$$

Using Proposition 20, we deduce that:

$$\overline{\text{cond-st}(\beta_i^{(l_{j_0})})} \supseteq \text{cond-st}(\sqcup_{c_j \leq n} \beta_i^{(l_j)} \downarrow_R) \cap \mathcal{T}(\mathcal{F}_{\text{if}}, \mathcal{N})$$

Which implies that:

$$\text{leave-st}(\beta_{l_{j_0}} \downarrow_R) \cap \text{cond-st}(\sqcup_{c_j \leq n} \beta_i^{(l_j)} \downarrow_R) = \emptyset \quad (11)$$

Moreover, if  $n = n_0 + 1$  and  $i \neq l_{n_0+1}$ , we can check that:

$$\begin{aligned} \sqcup_{c_j \leq n_0} \beta_i^{(l_j)} &\equiv (\sqcup_{c_j \leq n_0+1} \beta_i^{(l_j)}) \{ \sqcup_{c_j \leq n_0} \beta_{l_{n_0+1}}^{(l_j)} / []_{l_{n_0+1}} \} \\ &=_R \text{ if } (\sqcup_{c_j \leq n_0} \beta_{l_{n_0+1}}^{(l_j)}) \text{ then } (\sqcup_{c_j \leq n_0+1} \beta_i^{(l_j)}) \{ \text{true} / []_{l_{n_0+1}} \} \\ &\quad \text{else } (\sqcup_{c_j \leq n_0+1} \beta_i^{(l_j)}) \{ \text{false} / []_{l_{n_0+1}} \} \end{aligned} \quad (12)$$

Part 2. Similarly, let  $(\gamma_m)_{1 \leq m \leq m_{\max}}$  be an arbitrary ordering of:

$$\{\gamma \mid \gamma \leq_l^{h,l} (t, P)\}$$

Let  $1 \leq i_0 \leq i_{\max}$ . For every  $m$ , we have  $\gamma_m^{(i_0)}$  such that:

$$\gamma_m \equiv \gamma_m^{(i_0)} [\beta_{i_0}] \quad \text{and} \quad \text{leave-st}(\beta_{i_0} \downarrow_R) \cap \overline{\text{cond-st}(\gamma_m^{(i_0)})} = \emptyset$$

Moreover, we let  $\gamma_m^{(0)} \equiv \gamma_m$ . Let  $1 \leq m \leq m_{\max}$ , and let  $l_0, \dots, l_n$  be a sequence of distinct indices in  $\{0, \dots, i_{\max}\}$  such that  $l_0 = 0$ . We have equations corresponding to (11) and (12), with  $\sqcup_{c_j \leq n} \gamma_m^{(l_j)}$  instead of  $\sqcup_{c_j \leq n} \beta_i^{(l_j)}$ .

Part 3. Consider the following family of couples of sets:

$$\left\{ \left( \left( \sqcup_{c_j \leq n} \beta_i^{(l_j)} \{ e_j / []_{l_j} \mid j \leq n \} \downarrow_R \right)_{1 \leq i \leq i_{\max}}, \left( \sqcup_{c_j \leq n} \gamma_m^{(l_j)} \{ e_j / []_{l_j} \mid j \leq n \} \downarrow_R \right)_{1 \leq m \leq m_{\max}} \right) \mid \right. \\ \left. l_0, \dots, l_n \text{ distinct indices in } \{0, \dots, i_{\max}\} \text{ s.t. } l_0 = 0 \text{ and } (e_j)_{1 \leq j \leq n} \in \{\text{true}, \text{false}\}^n \right\} \quad (13)$$

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  $l_0, \dots, l_n$  be distinct indices in  $\{0, \dots, i_{\max}\}$  such that  $l_0 = 0$ , and let  $(e_j)_{1 \leq j \leq n} \in \{\text{true}, \text{false}\}^n$ .

*Base case.* If  $n = n_{\max} + 1$  then from (11) we get that for every  $l \neq i$  in  $\{1, \dots, i_{\max}\}$ :

$$\text{leave-st}(\beta_l \downarrow_R) \cap \text{cond-st}((\sqcup_{c_j \leq n} \beta_i^{(j)})\{e_j/\prod_j \mid j \leq n\} \downarrow_R) = \emptyset$$

Moreover:

$$\text{leave-st}((\sqcup_{c_j \leq n} \beta_i^{(j)})\{e_j/\prod_j \mid j \leq n\} \downarrow_R) \subseteq \text{leave-st}(\beta_l \downarrow_R)$$

Hence:

$$\text{leave-st}((\sqcup_{c_j \leq n} \beta_i^{(j)})\{e_j/\prod_j \mid j \leq n\} \downarrow_R) \cap \text{cond-st}((\sqcup_{c_j \leq n} \beta_i^{(j)})\{e_j/\prod_j \mid j \leq n\} \downarrow_R) = \emptyset \quad (14)$$

Similarly, for every  $1 \leq m \leq m_{\max}$ :

$$\text{leave-st}((\sqcup_{c_j \leq n} \beta_i^{(j)})\{e_j/\prod_j \mid j \leq n\} \downarrow_R) \cap \text{cond-st}((\sqcup_{c_j \leq n} \gamma_m^{(j)})\{e_j/\prod_j \mid j \leq n\} \downarrow_R) = \emptyset$$

By consequence, the following set is well-nested:

$$((\sqcup_{c_j \leq n} \beta_i^{(j)})\{e_j/\prod_j\} \downarrow_R)_{1 \leq i \leq i_{\max}}, ((\sqcup_{c_j \leq n} \gamma_m^{(j)})\{e_j/\prod_j\} \downarrow_R)_{1 \leq m \leq m_{\max}}$$

*Inductive Case.* If  $n \leq n_{\max} \neq 0$ , then from (12) we get that for every  $l \neq i$  in  $\{1, \dots, i_{\max}\}$ :

$$\begin{aligned} (\sqcup_{c_j \leq n} \beta_i^{(l_j)})\{e_{l_j}/\prod_{l_j} \mid j \leq n\} =_R \\ \text{if } ((\sqcup_{c_j \leq n} \beta_{l_{n+1}}^{(l_j)})\{e_{l_j}/\prod_{l_j} \mid j \leq n\}) \text{ then } (\sqcup_{c_j \leq n+1} \beta_i^{(l_j)})\{e_{l_j}/\prod_{l_j} \mid j \leq n\}\{\text{true}/\prod_{l_{n+1}}\} \\ \text{else } (\sqcup_{c_j \leq n+1} \beta_i^{(l_j)})\{e_{l_j}/\prod_{l_j} \mid j \leq n\}\{\text{false}/\prod_{l_{n+1}}\} \end{aligned}$$

As we did for (14), we can show that for every  $i \neq l_{n+1}$ :

$$\text{leave-st}(\sqcup_{c_j \leq n} \beta_{l_{n+1}}^{(l_j)} \downarrow_R) \cap \text{cond-st}((\sqcup_{c_j \leq n+1} \beta_i^{(l_j)})\{e_{l_j}/\prod_{l_j} \mid j \leq n+1\} \downarrow_R) = \emptyset$$

Where  $e_{l_{n+1}}$  is either true or false. Similarly, for every  $m$ :

$$\text{leave-st}(\sqcup_{c_j \leq n} \beta_{l_{n+1}}^{(l_j)} \downarrow_R) \cap \text{cond-st}((\sqcup_{c_j \leq n+1} \gamma_m^{(l_j)})\{e_{l_j}/\prod_{l_j} \mid j \leq n+1\} \downarrow_R) = \emptyset$$

Moreover, by induction hypothesis, we know that:

$$((\sqcup_{c_j \leq n+1} \beta_i^{(l_j)})\{e_{l_j}/\prod_{l_j} \mid j \leq n+1\} \downarrow_R)_i, ((\sqcup_{c_j \leq n+1} \gamma_i^{(l_j)})\{e_{l_j}/\prod_{l_j} \mid j \leq n+1\} \downarrow_R)_i$$

is well-nested for  $e_{l_{n+1}} \equiv \text{true}$  and for  $e_{l_{n+1}} \equiv \text{false}$ . We deduce that the following set is well nested:

$$((\sqcup_{c_j \leq n} \beta_i^{(l_j)})\{e_{l_j}/\prod_{l_j} \mid j \leq n\} \downarrow_R)_i, ((\sqcup_{c_j \leq n} \gamma_i^{(l_j)})\{e_{l_j}/\prod_{l_j} \mid j \leq n\} \downarrow_R)_i$$

*Conclusion.* Recall that  $\beta_i^{(l_0)} \equiv \beta_i^{(0)} \equiv \beta_i$ . Hence:

$$\left( \{\beta \downarrow_R \mid \beta \leq_c^{e,l} (t, P)\}, \{\gamma \downarrow_R \mid \gamma \leq_l^{e,l} (t, P)\} \right)$$

is the couple of sets:

$$((\sqcup_{c_j \leq 0} \beta_i^{(l_j)}) \downarrow_R)_{1 \leq i \leq i_{\max}}, ((\sqcup_{c_j \leq 0} \gamma_m^{(l_j)}) \downarrow_R)_{1 \leq m \leq m_{\max}}$$

which is the family of well-nested sets in (13), and is therefore well-nested.  $\square$



*Second Objective.* Consider a proof  $P \vdash^{\text{npf}} t \sim t'$ , 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  $\text{CCA}_2$  trace of  $P$ . Remember that we showed in Corollary 2.(ii) that for every  $j, j' \in J$ :

$$\text{leave-st}(\gamma_j \downarrow_R) \cap \text{leave-st}(\gamma_{j'} \downarrow_R) \neq \emptyset \quad \text{implies that} \quad \gamma_j \equiv \gamma_{j'}$$

This followed from the fact that given a leaf  $u \in \text{leave-st}(\gamma_j \downarrow_R)$ , 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  $j$ ,  $\text{leave-st}(\gamma_j \downarrow_R)$  contains a persistent term in  $t$ , i.e. that  $\text{leave-st}(\gamma_j \downarrow_R) \cap \text{leave-st}(t \downarrow_R)$  is non-empty. Since  $\text{leave-st}(t \downarrow_R)$  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  $j$ :

$$\text{leave-st}(\gamma_j \downarrow_R) \cap \text{leave-st}(t \downarrow_R) \neq \emptyset$$

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:*

$$\tau[] \equiv B \left[ \vec{c} \diamond \left( \vec{w}, \text{if } C[\vec{b} \diamond \vec{a}, []] \text{ then } u \text{ else } v \right) \right]$$

Let  $t$  be the term  $\tau[a]$ , and assume that  $a$  is spurious in  $t$ , and that:

- $a \notin \vec{a} \cup \vec{b} \cup \{\text{true}, \text{false}\} \cup \text{cond-st}(u \downarrow_R) \cup \text{cond-st}(v \downarrow_R)$ .
- $a \notin \rho$  where  $\rho$  is the set of conditions appearing on the path from the root to  $\text{if } C[\vec{b} \diamond \vec{a}, a] \text{ then } u \text{ else } v$ .

Then  $t \equiv \tau[a] =_R \tau[\text{true}]$ .

PROOF. We start by observing that:

$$\begin{aligned} \text{if } C[\vec{b} \diamond \vec{a}, a] \text{ then } u \text{ else } v &= _R \text{if } a \text{ then if } C[\vec{b} \diamond \vec{a}, \text{true}] \text{ then } u \text{ else } v \\ &\quad \text{else if } C[\vec{b} \diamond \vec{a}, \text{false}] \text{ then } u \text{ else } v \end{aligned}$$

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 :

$$\text{if } C[\vec{b} \diamond \vec{a}, \text{true}] \text{ then } u \text{ else } v =_R C_l[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v]$$

$$\text{if } C[\vec{b} \diamond \vec{a}, \text{false}] \text{ then } u \text{ else } v =_R C_r[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v]$$

Since  $a \notin \text{cond-st}(u \downarrow_R), \text{cond-st}(v \downarrow_R)$  we know that  $a \notin \vec{b}_u, \vec{b}_v$ . Moreover since  $a \notin \vec{a} \cup \vec{b}$  we know that  $a \notin \vec{b}_u, \vec{b}_v, \vec{b}, \vec{a}$ . Therefore:

$$a \notin \text{cond-st}(C_l[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v]) \quad a \notin \text{cond-st}(C_r[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v])$$

We get rid in  $C_l$  and  $C_r$  of all the conditions appearing in  $\rho$ . We let  $\vec{a}^l$  and  $\vec{a}^r$  be such that:

$$\vec{a}^l \subseteq \vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \setminus \rho \quad \vec{a}^r \subseteq \vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \setminus \rho$$



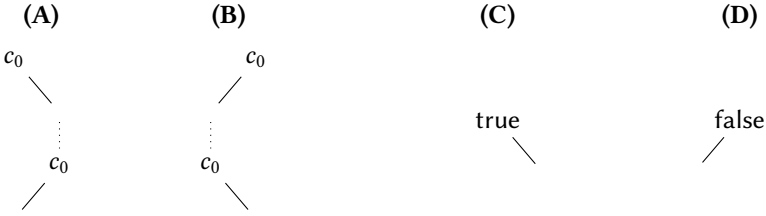
and  $C'_l, C'_r$  such that:

$$B \left[ \vec{c} \diamond \left( \vec{w}, C_l[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v] \right) \right] =_R B \left[ \vec{c} \diamond \left( \vec{w}, C'_l[\vec{a}^l \diamond \vec{t}_u, \vec{t}_v] \right) \right] \quad (15)$$

$$B \left[ \vec{c} \diamond \left( \vec{w}, C_r[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v] \right) \right] =_R B \left[ \vec{c} \diamond \left( \vec{w}, C'_r[\vec{a}^r \diamond \vec{t}_u, \vec{t}_v] \right) \right] \quad (16)$$

We know that  $a \notin \vec{a}^l$  and  $a \notin \vec{a}^r$ .

*Case 1.* If there exists  $c_0 \in \vec{c}$  such that the path  $\rho$  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  $\rho$ , and dotted edges represent a summary of a part of the path  $\rho$ .



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:

$$\text{if } x \text{ then } y \text{ else } ( \text{if } x \text{ then } v \text{ else } z ) \rightarrow_R^* \text{if } x \text{ then } y \text{ else } ( \text{if } x \text{ then } v' \text{ else } z ) \quad (\text{for all term } v')$$

to rewrite  $(\text{if } C[\vec{b} \diamond \vec{a}, a] \text{ then } u \text{ else } v)$  into the term  $(\text{if } a \text{ then if } C[\vec{b} \diamond \vec{a}, \text{true}] \text{ then } u \text{ else } v \text{ else } )$ .

*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  $\vec{w}$  such that  $s[\ ] =_R B' [\vec{c} \diamond (\vec{w}, [\ ])]$  and the path  $\rho'$  from the root of  $s$  to  $[\ ]$  is a subset of  $\rho$  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  $\rho'$ : indeed since the shape (A) and (B) (resp. (C) and (D)) are forbidden in  $\rho$ , we know that if we have a duplicate (resp. true or false) then we can always rewrite  $B$  such that the hole containing  $s$  does not disappear.

Let  $\rho' = c_1, \dots, c_n$ , we take  $B'$  minimal, i.e. only a branch  $c_1, \dots, c_n$ . We give an example of such an if-context in Figure 14.

We let  $\vec{w} = w_1, \dots, w_n$ , and we have:

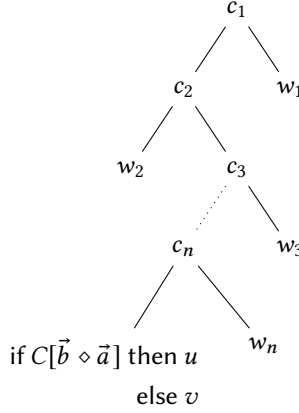
$$s =_R B' [c_1, \dots, c_n \diamond w_1, \dots, w_n, [\ ]]$$

We let  $>_u$  be a total ordering on if-free condition in  $R$ -normal form such that the  $n + 1$  maximum elements are  $c_1 >_u \dots >_u c_n >_u a$ . For every  $y$ , we let  $W_i[\vec{d}_i \diamond \vec{e}_i]$  be the  $R_{>_u}$ -normal form of  $w_i$ . Then:

$$s =_R B' \left[ c_1, \dots, c_n \diamond (W_i[\vec{d}_i \diamond \vec{e}_i])_{i \leq n}, [\ ] \right]$$

We get rid of any occurrence of  $c_1, \dots, c_n$  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  $\vec{d}'_i \cap \{c_j \mid j \leq i\} = \emptyset$  and:

$$s =_R B' \left[ c_1, \dots, c_n \diamond (W'_i[\vec{d}'_i \diamond \vec{e}'_i])_{i \leq n}, [\ ] \right]$$

Fig. 14. Example of if-context  $B'$ 

Using (15) and (16) we get:

$$t =_R B' \left[ c_1, \dots, c_n \diamond (W'_i[\vec{d}'_i \diamond \vec{e}'_i])_{i \leq n}, \begin{array}{l} \text{if } a \text{ then } C'_l[\vec{a}^l \diamond \vec{t}_u, \vec{t}_v] \\ \text{else } C'_r[\vec{a}^r \diamond \vec{t}_u, \vec{t}_v] \end{array} \right]$$

It is then quite easy to show by induction on the length of the reduction sequence that there exists a sequence  $1 \leq i_1 < \dots < i_k \leq n$  and an if-context  $B''$  such that:

$$\begin{aligned} t \downarrow_{R>_u} &\equiv \left( B' \left[ c_1, \dots, c_n \diamond (W'_i[\vec{d}'_i \diamond \vec{e}'_i])_{i \leq n}, \begin{array}{l} \text{if } a \text{ then } C'_l[\vec{a}^l \diamond \vec{t}_u, \vec{t}_v] \\ \text{else } C'_r[\vec{a}^r \diamond \vec{t}_u, \vec{t}_v] \end{array} \right] \right) \downarrow_{R>_u} \\ &\equiv B'' \left[ c_{i_1}, \dots, c_{i_k} \diamond (W'_{i_j}[\vec{d}'_{i_j} \diamond \vec{e}'_{i_j}])_{j \leq k}, \begin{array}{l} \text{if } a \text{ then } C'_l[\vec{a}^l \diamond \vec{t}_u, \vec{t}_v] \\ \text{else } C'_r[\vec{a}^r \diamond \vec{t}_u, \vec{t}_v] \end{array} \right] \downarrow_{R>_u} \end{aligned}$$

We deduce from this that  $a$  is spurious in:

$$\text{if } a \text{ then } C'_l[\vec{a}^l \diamond \vec{t}_u, \vec{t}_v] \text{ else } C'_r[\vec{a}^r \diamond \vec{t}_u, \vec{t}_v]$$

Since  $a$  will stay the top-most condition in the  $R$ -normal form of this term (because of the order  $>_u$  we chose), and since  $a \neq \text{true}$ ,  $a \neq \text{false}$  and  $a \notin \vec{a}^l, \vec{a}^r$ , there is only one rule that can be applied:  $\text{if } a \text{ then } x \text{ else } x \rightarrow x$ . Consequently:

$$C'_l[\vec{a}^l \diamond \vec{t}_u, \vec{t}_v] =_R C'_r[\vec{a}^r \diamond \vec{t}_u, \vec{t}_v]$$

Hence:

$$\begin{aligned} t &=_R B' \left[ c_1, \dots, c_n \diamond (W'_i[\vec{d}'_i \diamond \vec{e}'_i])_{i \leq n}, C'_l[\vec{a}^l \diamond \vec{t}_u, \vec{t}_v] \right] \\ &=_R s \left[ C'_l[\vec{a}^l \diamond \vec{t}_u, \vec{t}_v] \right] \\ &=_R B \left[ \vec{c} \diamond \left( \vec{w}, C'_l[\vec{a}^l \diamond \vec{t}_u, \vec{t}_v] \right) \right] \end{aligned}$$

Hence using (15) we get:

$$t =_R B \left[ \vec{c} \diamond \left( \vec{w}, C_l[\vec{b}_u, \vec{b}_v, \vec{b}, \vec{a} \diamond \vec{t}_u, \vec{t}_v] \right) \right] =_R B \left[ \vec{c} \diamond \left( \vec{w}, \text{if } C[\vec{b} \diamond \vec{a}, \text{true}] \text{ then } u \text{ else } v \right) \right] \quad \square$$

## 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  $s[\text{true}]$  and  $s[\text{false}]$ .

**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:*

$$s[\ ] \equiv B \left[ (C_i[\vec{a}_i, \ ] \diamond \vec{b}_i, \ ])_i \diamond (D_j[\vec{c}_j, \ ] \diamond \vec{t}_j)_j \right]$$

Then  $\text{leave-st}(s[a] \downarrow_R) = \text{leave-st}(s[\text{true}] \downarrow_R) \cup \text{leave-st}(s[\text{false}] \downarrow_R)$ .

**PROOF.** We know that  $s[a] =_R$  if  $a$  then  $s[\text{true}]$  else  $s[\text{false}]$ . Let  $>_u$  be a total order on if-free conditions in  $R$ -normal form such that  $a$  is minimal. It is straightforward to check that:

$$\begin{aligned} s[a] \downarrow_{R>u} &\equiv (\text{if } a \text{ then } s[\text{true}] \text{ else } s[\text{false}]) \downarrow_{R>u} \\ &\equiv \begin{cases} (s[\text{true}]) \downarrow_{R>u} & \text{if } s[\text{true}] =_{R>u} s[\text{false}] \\ \text{if } a \text{ then } (s[\text{true}]) \downarrow_{R>u} \text{ else } (s[\text{false}]) \downarrow_{R>u} & \text{otherwise} \end{cases} \end{aligned}$$

Therefore:

$$\text{leave-st}(s[a] \downarrow_{R>u}) = \text{leave-st}(s[\text{true}] \downarrow_{R>u}) \cup \text{leave-st}(s[\text{false}] \downarrow_{R>u})$$

The wanted result follows from Proposition 5.  $\square$

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:*

$$C[\vec{a} \diamond \vec{b}] \quad t_l \equiv B^l \left[ (C_i^l[\vec{a}_i^l \diamond \vec{b}_i^l])_i \diamond (D_j^l[\vec{c}_j^l \diamond \vec{t}_j^l])_j \right] \quad t_r \equiv B^r \left[ (C_i^r[\vec{a}_i^r \diamond \vec{b}_i^r])_i \diamond (D_j^r[\vec{c}_j^r \diamond \vec{t}_j^r])_j \right]$$

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  $(\vec{a} \cup \vec{b}) \cap \{\text{true}, \text{false}\} = \emptyset$ .
- $\vec{b} \cap (\bigcup_{x \in \{l, r\}, i} \vec{a}_i^x, \vec{b}_i^x) = \emptyset$  and  $\vec{b} \cap (\bigcup_{x \in \{l, r\}, j} \vec{c}_j^x, \vec{t}_j^x) = \emptyset$ .
- $\vec{a} \cap \vec{b} = \emptyset$ .

Then:

$$\text{leave-st}(t_l \downarrow_R) \cup \text{leave-st}(t_r \downarrow_R) \subseteq \text{leave-st} \left( (\text{if } C[\vec{a} \diamond \vec{b}] \text{ then } t_l \text{ else } t_r) \downarrow_R \right)$$

**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  $>_u$  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  $R_{>_u}$ -normal form of  $t_l$  and  $t_r$ . By Proposition 5, we know that:

$$\text{leave-st}(t_l \downarrow_R) = \text{leave-st}(t_l \downarrow_{R_{>_u}}) = \text{leave-st} \left( (D_l[\vec{a}_l \diamond \vec{t}_l]) \downarrow_{R_{>_u}} \right) \quad (17)$$

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  $b \notin (\bigcup_{x \in \{l, r\}, j} \vec{c}_j^x, \vec{t}_j^x)$ , 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  $b \notin \vec{a}_r$ .

$$(\text{if } b \text{ then } t_l \text{ else } t_r) \downarrow_{R>u} \equiv (\text{if } b \text{ then } D_l[\vec{a}_l \diamond \vec{t}_l] \text{ else } D_r[\vec{a}_r \diamond \vec{t}_r]) \downarrow_{R>u}$$

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  $b \notin \{\text{true}, \text{false}\}$  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  $x \rightarrow x$ . 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:

$$\begin{aligned} \text{if } C[\vec{a}, a \diamond \vec{b}] \text{ then } t_l \text{ else } t_r =_R & \text{if } a \text{ then if } C[\vec{a}, \text{true} \diamond \vec{b}] \text{ then } t_l \text{ else } t_r \\ & \text{else if } C[\vec{a}, \text{false} \diamond \vec{b}] \text{ then } t_l \text{ else } t_r \end{aligned}$$

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  $C[\vec{a}, \text{true} \diamond \vec{b}]$  and  $C[\vec{a}, \text{false} \diamond \vec{b}]$ . 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  $\vec{a}' \cup \vec{a}'' \subseteq \vec{a} \setminus \{a\}$  and  $\vec{b}' \cup \vec{b}'' \subseteq \vec{b} \setminus \{a\}$ . Then:

$$\text{if } C[\vec{a}, \text{true} \diamond \vec{b}] \text{ then } t_l \text{ else } t_r =_R \boxed{\text{if } C'[\vec{a}' \diamond \vec{b}'] \text{ then } s_l[\text{true}] \text{ else } s_r[\text{true}]}$$

$$\text{if } C[\vec{a}, \text{false} \diamond \vec{b}] \text{ then } t_l \text{ else } t_r =_R \boxed{\text{if } C''[\vec{a}'' \diamond \vec{b}''] \text{ then } s_l[\text{false}] \text{ else } s_r[\text{false}]}$$

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  $(\vec{a} \cup \vec{b}) \cap \{\text{true}, \text{false}\} = \emptyset$ , we know that  $(\vec{a}' \cup \vec{b}') \cap \{\text{true}, \text{false}\} = \emptyset$ .
- The term  $s_l[a]$  is obtained from  $t_l$  by replacing every occurrence of  $a$  by true. Hence, since  $\text{true} \notin \vec{b}$ ,  $\vec{b}' \subseteq \vec{b}$  and:

$$\vec{b} \cap (\bigcup_{x \in \{l,r\}, i} \vec{a}_i^x, \vec{b}_i^x) = \emptyset \quad \vec{b} \cap (\bigcup_{x \in \{l,r\}, j} \vec{c}_j^x) = \emptyset$$

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:

$$\begin{aligned} & \text{leave-st}(s_l[\text{true}] \downarrow_R) \cup \text{leave-st}(s_r[\text{true}] \downarrow_R) \\ \subseteq & \boxed{\text{leave-st}(\text{if } C'[\vec{a}' \diamond \vec{b}'] \text{ then } s_l[\text{true}] \text{ else } s_r[\text{true}] \downarrow_R)} \end{aligned}$$

Similarly, by applying the induction hypothesis on the rewriting of the blue framed term, we get:

$$\begin{aligned} & \text{leave-st}(s_l[\text{false}] \downarrow_R) \cup \text{leave-st}(s_r[\text{false}] \downarrow_R) \\ \subseteq & \boxed{\text{leave-st}(\text{if } C''[\vec{a}'' \diamond \vec{b}''] \text{ then } s_l[\text{false}] \text{ else } s_r[\text{false}] \downarrow_R)} \end{aligned}$$

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  $\gamma$  of a term  $t$  has a persistent leaf.

LEMMA 21. *Let  $s$  be a term of the form:*

$$s \equiv A \left[ \vec{d} \diamond (B_l[(\beta_{i,l})_i \diamond (\gamma_{j,l})_j])_l \right]$$

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)  $(\vec{d} \cup \bigcup_{i,l} \text{leave-st}(\beta_{i,l} \downarrow_R)) \cap \{\text{true}, \text{false}\} = \emptyset$ .
- (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:

$$\{\beta_{i,l} \downarrow_R \mid i\}, \{\gamma_{j,l} \downarrow_R \mid j\}$$

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:

$$\vec{a}_i \equiv \text{cond-st}(\beta_i \downarrow_R) \quad \vec{b}_i \equiv \text{leave-st}(\beta_i \downarrow_R) \quad \vec{c}_i \equiv \text{cond-st}(\gamma_j \downarrow_R) \quad \vec{t}_i \equiv \text{leave-st}(\gamma_j \downarrow_R) \\ \beta_i \downarrow_R \equiv C_i[\vec{a}_i \diamond \vec{b}_i] \quad \gamma_j \downarrow_R \equiv D_j[\vec{c}_j \diamond \vec{t}_j]$$

We know that:

$$s =_R B \left[ (C_i[\vec{a}_i \diamond \vec{b}_i])_i \diamond (D_j[\vec{c}_j \diamond \vec{t}_j])_j \right]$$

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  $\vec{a}'_j, \vec{a}''_j \subseteq \vec{a}_j \setminus \vec{b}_0$  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:

$$D_j[\vec{c}_j \diamond \vec{t}_j] =_R \text{if } C_0[\vec{a}_0 \diamond \vec{b}_0] \text{ then } D'_j[\vec{c}'_j \diamond \vec{t}'_j] \text{ else } D''_j[\vec{c}''_j \diamond \vec{t}''_j]$$

where  $\vec{c}'_j, \vec{c}''_j \subseteq \vec{a}_j \setminus \vec{b}_0$  and  $\vec{t}'_j, \vec{t}''_j \subseteq \vec{t}_j$ . Then:

$$s \equiv \text{if } C_0[\vec{a}_0 \diamond \vec{b}_0] \text{ then } B^l \left[ \left( C'_i[\vec{a}'_i \diamond \vec{b}'_i] \right)_{i \in I^l} \diamond \left( D'_j[\vec{c}'_j \diamond \vec{t}'_j] \right)_{j \in J^l} \right] \text{ } \color{red}{\text{Strue}} \\ \text{else } B^r \left[ \left( C''_i[\vec{a}''_i \diamond \vec{b}''_i] \right)_{i \in I^r} \diamond \left( D''_j[\vec{c}''_j \diamond \vec{t}''_j] \right)_{j \in J^r} \right] \color{red}{\text{Sfalse}}$$

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  $s_{\text{true}}$  (for  $j \in J^r$ , we apply the induction hypothesis to  $s_{\text{false}}$ ). Lets check that the premises hold for  $s_{\text{true}}$ :

- (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:

$$C'_i[\vec{a}'_i \diamond \vec{b}'_i] \equiv C'_j[\vec{a}'_j \diamond \vec{b}'_j] \quad C''_i[\vec{a}''_i \diamond \vec{b}''_i] \equiv C''_j[\vec{a}''_j \diamond \vec{b}''_j]$$

- Using the inductive property of well-nested couples (item (iv)) we know that the following couple of sets is well-nested:

$$\left\{ \left\{ C'_i[\vec{a}'_i \diamond \vec{b}'_i] \mid i \in I^l \cup I^r \cup \{0\} \right\}, \left\{ D'_j[\vec{c}'_j \diamond \vec{t}'_j] \mid j \in J^l \cup J^r \right\} \right\}$$

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:

$$\left\{ \left\{ C'_i[\vec{a}'_i \diamond \vec{b}'_i] \mid i \in I^l \cup \{0\} \right\}, \left\{ D'_j[\vec{c}'_j \diamond \vec{t}'_j] \mid j \in J^l \right\} \right\}$$

We can apply the induction hypothesis to  $s_{\text{true}}$ , 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  $S' = S \setminus \{0\}$ . 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  $\vec{a}'_i \subseteq \vec{a}_i \setminus \vec{b}_0, \dots$ ), so it only remains to show that:

$$\vec{b}_0 \cap \bigcup_{i \in S'} \vec{b}'_i, \vec{b}''_i = \emptyset \quad (18)$$

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:

$$B[] \equiv A[([])_{d \in \vec{d}} \diamond (B_l[])_l]$$

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.  $\square$

## 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  $\gamma \leq_c^{\epsilon, l}(t, P)$  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  $\beta, \beta' \leq_c^{\epsilon, l}(t, P)$ , we have  $\text{cond-st}(\beta \downarrow_R) \cap \text{leave-st}(\beta \downarrow_R) = \emptyset$ .
- (ii)  $\left( \bigcup_{\beta \leq_c^{\epsilon, l}(t, P)} \text{leave-st}(\beta \downarrow_R) \right) \cap \{\text{true}, \text{false}\} = \emptyset$ .
- (iii) For every  $\beta, \beta' \leq_c^{\epsilon, l}(t, P)$  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  $\beta, \beta' \leq_c^{\epsilon, l}(t, P)$ , 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:

$$\left( \left\{ \beta \downarrow_R \mid \beta \leq_c^{\epsilon, l}(t, P) \right\}, \left\{ \gamma \downarrow_R \mid \gamma \leq_c^{\epsilon, l}(t, P) \right\} \right)$$

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 disjoint, 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:

$$\text{if true then } v \text{ else } w \rightarrow w \qquad \text{if false then } v \text{ else } w \rightarrow w$$

**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  $\text{false} \notin \vec{a}_i \cup \vec{b}_i$  (resp.  $\text{true} \notin \vec{a}_i \cup \vec{b}_i$ ).

Then the left-most (resp. right-most) element of  $\vec{t}$  is  $s$ .

**PROOF.** It suffices to show that the existence of a decomposition satisfying these two properties is preserved by  $\rightarrow_R$ , which is simple. We conclude by observing that since  $s$  is if-free, the only decomposition of  $s \downarrow_R$  into  $C \left[ \left( D_i [\vec{a}_i \diamond \vec{b}_i] \right)_i \diamond \vec{t} \right]$  is such that  $C \equiv []$ . Hence  $\vec{t}$  is a single element  $u$ , and  $u \equiv s \downarrow_R \equiv s$ .  $\square$

We would like to prove that for every  $b$ , there exists no derivation of  $b \sim \text{true}$  or  $b \sim \text{false}$ . 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  $b \sim \text{true}$  or false, as long as  $b$  is if-free and the proof is in the fragment  $\mathcal{A}_>$ .

**PROPOSITION (9).** *Let  $b$  an if-free condition in  $R$ -normal, with  $b \not\equiv \text{false}$  (resp.  $b \not\equiv \text{true}$ ). Then there exists no derivation of  $b \sim \text{false}$  (resp.  $b \sim \text{true}$ ) in  $\mathcal{A}_>$ .*

**PROOF.** We prove only that there is no derivation of  $b \sim \text{false}$  in  $\mathcal{A}_>$  (the proof that there is no derivation of  $b \sim \text{true}$  in  $\mathcal{A}_>$  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  $P \vdash^{\text{mpf}} b \sim \text{false}$ . We choose the condition  $b$  such that its proof  $P$  is of minimal size.

First the minimality of the derivation implies that for all  $h \in \text{index}(P)$ , for all  $b_0$  such that  $b_0 \leq_{\text{cs}}^h (b, P)$  or  $b_0 \leq_{\text{cs}}^h (\text{false}, P)$ ,  $b_0 \neq_R \text{false}$ . Let  $H = \text{cs-pos}(P)$ . We now focus on the left-most branch of the proof.

*First Part.* Let  $l \in \text{label}(P)$ . First we show that for all  $\beta \leq_c^{\epsilon, l} (b, P)$ ,  $\beta \neq_R \text{false}$ . Assume that this is not the case, let  $\beta =_R \text{false}$  and  $\beta'$  be such that  $(\beta, \beta') \leq_{c \rightarrow c}^{\epsilon, l} (b \sim \text{false}, P)$ . If  $\beta =_R \beta' =_R \text{false}$  then there is an easy proof cut elimination which yields a smaller proof  $P'$  of  $b \sim \text{false}$ .

Hence assume  $\beta' \neq_R \text{false}$ . If  $\beta =_R \text{false}$  then  $\text{leave-st}(\beta \downarrow_R) = \text{leave-st}(\text{false} \downarrow_R) = \{\text{false}\}$ . Since  $\beta$  is a normalized basic condition (for the  $\text{CCA}_2$  trace  $\mathcal{S}$  of its branch in  $P$ ), and since false is a normalized basic condition, using Proposition 17 we have  $\beta \equiv \text{false}$ .



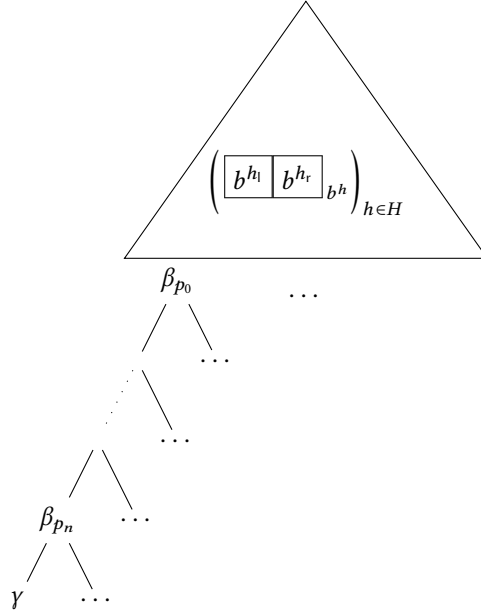


Fig. 15. Shape of the Term in the Proof of Proposition 9

There exists a derivation of  $\beta \sim \beta'$  in  $\mathfrak{F}(\text{FA}_s^* \cdot \text{Dup}^* \cdot \text{CCA}_2)$ . Since  $\beta \equiv \text{false}$ , no rules in  $\text{FA}_s$  are applied. Therefore the derivation is only an application of  $\text{CCA}_2$ , which is not possible. Similarly we do not have  $\beta \neq_R \text{false}$  and  $\beta' =_R \text{false}$ .

*Part 2.* Using Proposition 17 we know that  $\beta \neq_R \text{false}$  implies that for all  $u \in \text{leave-st}(\beta \downarrow_R)$ ,  $u \neq \text{false}$ . Moreover, for any term  $w$ ,  $w \downarrow_R$  does not contain false in its conditions (or we could apply if false then  $x$  else  $y \rightarrow y$ ). Hence for every  $a \in \text{cond-st}(\beta \downarrow_R)$ ,  $a \neq \text{false}$ .

We let  $(\gamma, \gamma') \leq_1^{\epsilon, l} (b \sim \text{false}, P)$  be the left-most elements (as shown in Figure 15). For all  $a \in \text{cond-st}(\gamma \downarrow_R)$ ,  $a \neq \text{false}$ . Hence if we let  $u_0 \in \text{leave-st}(\gamma \downarrow_R)$  be the left-most leaf element of  $\gamma \downarrow_R$ , then by Proposition 23 we know that  $u_0 \equiv b$ . Recall that  $b \neq_R \text{false}$ .

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  $\gamma' \equiv \text{false}$ . Since there exists a derivation of  $\gamma \sim \gamma'$  in  $\mathfrak{F}(\text{FA}_s^* \cdot \text{Dup}^* \cdot \text{CCA}_2)$ , no rule in  $\text{FA}_s$  is applied. Therefore the derivation is only an application of  $\text{CCA}_2$ . Contradiction.  $\square$

Thanks to this proposition, we can ensure that any proof  $P$  of  $t \sim t'$  does not contain a  $\text{CS}_\square$  or  $\text{BFA}$  application on true or false: if we have a  $\text{CS}_\square$  or  $\text{BFA}$  application on (true, true) or (false, false) then there is an easy proof cut elimination, and the previous proposition ensures that there are no  $\text{CS}_\square$  or  $\text{BFA}$  applications on (true,  $b$ ), ( $b$ , true), (false,  $b$ ) or ( $b$ , false) (with  $b \neq_R \text{false}$ , 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  $l \in \text{label}(P')$ ,  $h \in \text{index}(P')$ ,  $x \in \{l, r\}$  we have:

$$\forall \beta \in \left( (\leq_c^{h_x, l} \cup \leq_{cs}^{h_x})(t, P') \right) \cup \left( (\leq_c^{h_x, l} \cup \leq_{cs}^{h_x})(t', P') \right), \quad \{\text{false}, \text{true}\} \cap \text{leave-st}(\beta \downarrow_R) = \emptyset$$

PROOF. Through simple proof cut eliminations, We can construct a proof  $P'$  from  $P$  such that:

$$\{(\text{true}, \text{true}), (\text{false}, \text{false})\} \cap (\leq_{c \sim c}^{h_x, l} (t \sim t', P) \cup \leq_{cs \sim cs}^{h_x} (t \sim t', P)) = \emptyset$$

Then using Proposition 9 we know that for all:

$$(\beta, \beta') \in (\leq_{c \sim c}^{h_x, l} (t \sim t', P) \cup \leq_{cs \sim cs}^{h_x} (t \sim t', P))$$

the conditions  $\beta$  and  $\beta'$  are such that  $\beta \neq_R \text{false}$  and  $\beta' \neq_R \text{false}$  (same with true). Finally if  $\beta \neq_R \text{false}$  then using Proposition 17 we know that for every  $u \in \text{leave-st}(\beta \downarrow_R)$ ,  $u \neq \text{false}$  (idem with true).  $\square$

## G.2 Basic Terms have Disjoints Conditions and Leaves

We now prove that every normalized basic terms has disjoint conditions and leaves. Let  $\beta$  be a normalized basic terms. First, we show that every condition term  $b$  in  $\text{cond-st}(\beta \downarrow_R)$  is the leaf of another normalized basic term  $\beta'$ , which is a strict subterm of  $\beta$ . Therefore, if  $\text{cond-st}(\beta \downarrow_R) \cap \text{leave-st}(\beta \downarrow_R) \neq \emptyset$  then there exists  $\beta'$  such that  $\text{leave-st}(\beta \downarrow_R) \cap \text{leave-st}(\beta' \downarrow_R) \neq \emptyset$ . Using Proposition 17, we deduce that  $\beta \equiv \beta'$ , which contradicts the fact that  $\beta'$  is a strict sub-term of  $\beta$ .

First, we define the set of normalized basic conditions appearing in a term  $t$ .

*Definition 63.* For all term  $t$ , we let  $\langle_{bc}^S t$  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:

$$\langle_{bc}^S t = \vec{b} \cup (\langle_{bc}^S \vec{b}) \cup (\langle_{bc}^S \vec{u})$$

- If  $t$  is a  $\mathcal{S}$ -normalized basic term  $B[\vec{w}, (\alpha_i)_i, (\text{dec}_j)_j]$ , then:

$$\langle_{bc}^S t = \bigcup_i \langle_{bc}^S \alpha_i \cup \bigcup_j \langle_{bc}^S \text{dec}_j$$

- For every  $\mathcal{S}$ -encryption oracle call  $t \equiv \{u\}_{pk}^r$ , then:

$$\langle_{bc}^S t = \langle_{bc}^S u$$

- 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  $\mathbf{0}(\text{dec}(s[(\alpha_i), (\text{dec}_j)_j], sk))$  or  $\text{dec}(s[(\alpha_i), (\text{dec}_j)_j], sk)$ . Then:

$$\langle_{bc}^S t = \vec{b} \cup (\langle_{bc}^S \vec{b}) \cup \bigcup_i \langle_{bc}^S \alpha_i \cup \bigcup_j \langle_{bc}^S \text{dec}_j$$

We show that the over-approximated set of conditions  $\overline{\text{cond-st}(\beta)}$  is exactly the over-approximated set of leaves of the normalized basic conditions that are subterm of  $\beta$ .

PROPOSITION 25. For every term  $\beta$  such that  $\beta$  is a  $\mathcal{S}$ -normalized basic term,  $\mathcal{S}$ -normalized simple term,  $\mathcal{S}$ -decryption oracle call or  $\mathcal{S}$ -encryption oracle call:

$$\overline{\text{cond-st}(\beta)} = \bigcup_{u \langle_{bc}^S \beta} \overline{\text{leave-st}(u)}$$

PROOF. We prove this by induction on the order  $\langle_{ind}^S$ , 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  $\beta$  is minimal for  $<_{\text{ind}}^S$ , then we have the following cases:

- $\mathcal{S}$ -decryption oracle call:  $\beta$  is of the form  $C[\vec{b} \diamond \vec{u}]$ , and there exists  $s, \text{sk}$  such that terms in  $\vec{u}$  are of the form  $\mathbf{0}(\text{dec}(s, \text{sk}))$  or  $\text{dec}(s, \text{sk})$ , and  $s$  is if-free. Moreover by minimality of  $\beta$  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  $\overline{\text{cond-st}}(\beta) = \emptyset$ . Finally since  $\beta$  is minimal there are no  $u$  such that  $u <_{\text{bc}}^S \beta$ .
- $\mathcal{S}$ -encryption oracle call case cannot happen, as  $\beta$  would not be minimal.
- $\mathcal{S}$ -normalized basic term:  $\beta$  contains no if then else symbol, hence  $\overline{\text{cond-st}}(\beta) = \emptyset$ . Moreover since  $\beta$  is minimal there are no  $u$  such that  $u <_{\text{bc}}^S \beta$ .
- $\mathcal{S}$ -normalized simple term case cannot happen, as  $\beta$  would not be minimal.

*Inductive Case.* Let  $\beta$  be such that for all  $\beta' \neq \beta$ , if  $\beta' <_{\text{ind}}^S \beta$  then the property holds for  $\beta'$ .

- $\mathcal{S}$ -normalized basic term:  $\beta$  is of the form  $B[\vec{w}, (\alpha_i)_i, (\text{dec}_j)_j]$ . The result is then immediate by induction hypothesis and using the definition of  $\overline{\text{cond-st}}(\cdot)$  and  $<_{\text{bc}}^S$ :

$$\begin{aligned} \overline{\text{cond-st}}(\beta) &= \bigcup_i \overline{\text{cond-st}}(\alpha_i) \quad \cup \quad \bigcup_j \overline{\text{cond-st}}(\text{dec}_j) \quad (\text{By definition of } \overline{\text{cond-st}}(\cdot)) \\ &= \bigcup_i \bigcup_{u <_{\text{bc}}^S \alpha_i} \overline{\text{leave-st}}(u) \quad \cup \quad \bigcup_j \bigcup_{u <_{\text{bc}}^S \text{dec}_j} \overline{\text{leave-st}}(u) \quad (\text{By induction hypothesis}) \\ &= \bigcup_{u <_{\text{bc}}^S \beta} \overline{\text{leave-st}}(u) \quad (\text{By definition of } <_{\text{bc}}^S) \end{aligned}$$

- $\mathcal{S}$ -decryption oracle call:  $t$  is of the form  $C[\vec{g} \diamond \vec{u}]$ , where there exists  $s, \text{sk}$  such that terms in  $\vec{u}$  are of the form  $\mathbf{0}(\text{dec}(s[(\alpha_i), (\text{dec}_j)_j], \text{sk}))$  or  $\text{dec}(s[(\alpha_i), (\text{dec}_j)_j], \text{sk})$ , and  $s$  is if-free. Then:

$$\begin{aligned} \overline{\text{cond-st}}(\beta) &= \bigcup_i \overline{\text{cond-st}}(\alpha_i) \quad \cup \quad \bigcup_j \overline{\text{cond-st}}(\text{dec}_j) \quad \cup \quad \overline{\text{cond-st}}(\vec{g}) \quad \cup \quad \overline{\text{leave-st}}(\vec{g}) \\ & \quad (\text{By definition of } \overline{\text{cond-st}}(\cdot)) \\ &= \bigcup_i \bigcup_{u <_{\text{bc}}^S \alpha_i} \overline{\text{leave-st}}(u) \quad \cup \quad \bigcup_j \bigcup_{u <_{\text{bc}}^S \text{dec}_j} \overline{\text{leave-st}}(u) \quad \cup \quad \bigcup_{u <_{\text{bc}}^S \vec{g}} \overline{\text{leave-st}}(u) \quad \cup \quad \overline{\text{leave-st}}(\vec{g}) \\ (\text{By induction hypothesis: remark that guards in } \vec{g} \text{ are } \mathcal{S}\text{-normalized basic terms s.t. } \vec{g} \leq_{\text{bt}}^S \beta) \\ &= \bigcup_{u <_{\text{bc}}^S \beta} \overline{\text{leave-st}}(u) \quad (\text{By definition of } <_{\text{bc}}^S) \end{aligned}$$

- $\mathcal{S}$ -encryption oracle call:  $t$  is of the form  $\{s\}_{\text{pk}}^r$ , then:

$$\begin{aligned} \overline{\text{cond-st}}(\beta) &= \overline{\text{cond-st}}(s) \quad (\text{By definition of } \overline{\text{cond-st}}(\cdot)) \\ &= \bigcup_{u <_{\text{bc}}^S s} \overline{\text{leave-st}}(u) \quad (\text{By induction hypothesis}) \\ &= \bigcup_{u <_{\text{bc}}^S \beta} \overline{\text{leave-st}}(u) \quad (\text{By definition of } <_{\text{bc}}^S) \end{aligned}$$

- $\mathcal{S}$ -normalized simple term:  $t$  is of the form  $C[\vec{b} \diamond \vec{v}]$ . Then:

$$\overline{\text{cond-st}}(\beta) = \overline{\text{cond-st}}(\vec{b}) \quad \cup \quad \overline{\text{cond-st}}(\vec{v}) \quad \cup \quad \overline{\text{leave-st}}(\vec{b}) \quad (\text{By definition of } \overline{\text{cond-st}}(\cdot))$$

$$\begin{aligned}
&= \bigcup_{u <_{bc}^S \vec{b}} \overline{\text{leave-st}}(u) \cup \bigcup_{u <_{bc}^S \vec{c}} \overline{\text{leave-st}}(u) \cup \overline{\text{leave-st}}(\vec{b}) \quad (\text{By induction hypothesis}) \\
&= \bigcup_{u <_{bc}^S \beta} \overline{\text{leave-st}}(u) \quad (\text{By definition of } <_{bc}^S) \quad \square
\end{aligned}$$

We can now prove that every normalized basic terms has disjoint conditions and leaves, using Proposition 17 and the result above.

PROPOSITION 26. *Let  $P \vdash^{\text{npf}} t \sim t'$ . Then for all  $h, l$  for all  $\beta \leq_{bt}^{h,l}(t, P)$ :*

$$\overline{\text{cond-st}}(\beta) \cap \overline{\text{leave-st}}(\beta) = \emptyset$$

PROOF. Let  $h, l$  and  $\beta \leq_{bt}^{h,l}(t, P)$  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  $S_l^P$ -normalized basic term  $u <_{bc}^{S_l} \beta$  such that  $\overline{\text{leave-st}}(u) \cap \overline{\text{leave-st}}(\beta) \neq \emptyset$ .

By Proposition 17,  $u \equiv \beta$ . But  $u <_{bc}^{S_l} \beta$  implies that  $u$  is a strict subterm of  $\beta$ . 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<sup>11</sup>. Therefore, we need to show that if some normalized basic term  $\beta$  appears twice in the same branch, then there is a proof cut. We have three possibilities:

- The two occurrences of  $\beta$  are involved in  $\overline{\text{BFA}}$  applications.
- The two occurrences of  $\beta$  are involved in  $\overline{\text{CS}}_{\square}$  applications.
- One occurrence of  $\beta$  is with an  $\overline{\text{BFA}}$  application, the other with a  $\overline{\text{CS}}_{\square}$  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  $P \vdash^{\text{npf}} a, a' \sim b, c$ , 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  $P' \vdash^{\text{npf}} t \sim t'$  since  $P \vdash^{\text{npf}} a, a' \sim b, c$ . Using Proposition 24, we can assume that for every  $h \in \text{index}(P), l, x$ :

$$\forall \beta \in \left( (\leq_c^{h_x, l} \cup \leq_{cs}^{h_x, l})(t, P') \right) \cup \left( (\leq_c^{h_x, l} \cup \leq_{cs}^{h_x, l})(t', P') \right), \quad \{\text{false, true}\} \cap \text{leave-st}(\beta \downarrow_R) = \emptyset$$

Let  $(\gamma, \gamma') \leq_1^{\epsilon, l}(t \sim t', P)$  be the left-most elements of  $t$  and  $t'$ . By Proposition 23 we know that  $\langle a, a \rangle \downarrow_R \in \text{leave-st}(\gamma \downarrow_R)$  and  $\langle b, c \rangle \downarrow_R \in \text{leave-st}(\gamma' \downarrow_R)$ . 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  $\mathfrak{F}(\text{FA}_s^* \cdot \text{Dup}^* \cdot \text{CCA}_2)$ , we know that there exist  $S_l^P$ -normalized basic terms  $\gamma_1, \gamma_2$  and  $S_l^{P'}$ -normalized basic terms  $\gamma'_1, \gamma'_2$  such that  $\gamma =_R \langle \gamma_1, \gamma_2 \rangle$ ,  $\gamma' =_R \langle \gamma'_1, \gamma'_2 \rangle$ , and  $\gamma_1, \gamma_2 \sim \gamma'_1, \gamma'_2$  is provable in  $\mathfrak{F}(\text{FA}_s^* \cdot \text{Dup}^* \cdot \text{CCA}_2)$ .

Moreover  $a \in \text{leave-st}(\gamma_1 \downarrow_R)$  and  $a \in \text{leave-st}(\gamma_2 \downarrow_R)$ , hence  $\text{leave-st}(\gamma_1 \downarrow_R) \cap \text{leave-st}(\gamma_2 \downarrow_R) \neq \emptyset$ . 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  $\mathfrak{F}(\text{FA}_s^* \cdot \text{Dup}^* \cdot \text{CCA}_2)$ . By Lemma 2,  $\gamma'_1 \equiv \gamma'_2$ . We conclude by observing that since  $\langle b, c \rangle$  is the left-most element of  $\gamma' \downarrow_R$ ,  $b$  and  $c$  are the left-most element of, respectively,  $\gamma'_1$  and  $\gamma'_2$ . Therefore  $b \equiv c$ .  $\square$

<sup>11</sup>Indeed, we recall that Proposition 17 shows that if  $\text{leave-st}(\beta \downarrow_R) \cap \text{leave-st}(\beta' \downarrow_R) \neq \emptyset$  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  $\{\text{then, else}\}$ .

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:

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

Then  $\delta \vec{\rho} = (b_1, \text{then}), (b_2, \text{else})$  is the directed path corresponding to the branch starting at the root of  $s$  and ending at the term  $t_2$ . Moreover,  $\vec{\rho} = b_1, b_2$ .  $\diamond$

*Definition 65.* Let  $P \vdash^{\text{npf}} t \sim t'$ , we know that  $t$  is of the form:

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

For all  $l$ , we let:

- $\delta \text{cs-path}^{\epsilon, l}(t, P)$  be the directed path of condition occurring from the root of  $t$  to  $D_l[]$  in  $P$ .
- $\delta \text{cs-path}^{\epsilon, l}(t \sim t', P)$  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  $h \in \text{index}(P)$ ,  $x \in \{l, r\}$  by having:

$$\delta \text{cs-path}^{h, l}(t, P) = \delta \text{cs-path}^{\epsilon, l}(b, \text{extract}_x(h, P)) \\ \text{and } \delta \text{cs-path}^{h, l}(t \sim t', P) = \delta \text{cs-path}^{\epsilon, l}(b \sim b', \text{extract}_x(h, P))$$

where  $\text{extract}_x(h, P)$  is a proof of  $b \sim b'$ .

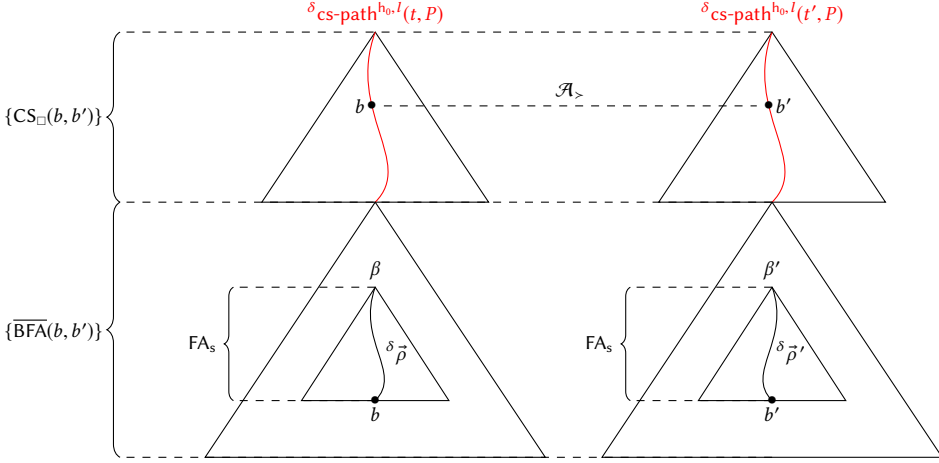
We let the depth of a position  $h$  in  $P$  to be the number of nested applications of the  $\text{CS}_{\square}$  rule to  $h$ .

*Definition 66.*  $\text{sym}] \text{if-depth}_P$  Let  $P \vdash^{\text{npf}} t \sim t'$ . For every  $h \in \text{index}(P)$ , we let  $\text{if-depth}_P(h)$  be the depth of  $h$  in  $P$ , defined by:

$$\text{if-depth}_P(h) = \begin{cases} 0 & \text{if } h \in \text{cs-pos}(P) \\ 1 + \text{if-depth}_{P^l}(h) & \text{if } \exists g \in \text{cs-pos}(P) \text{ s.t. } h \in \text{index}(P^l) \text{ where } P^l = \text{extract}_l(g, P) \\ 1 + \text{if-depth}_{P^r}(h) & \text{if } \exists g \in \text{cs-pos}(P) \text{ s.t. } h \in \text{index}(P^r) \text{ where } P^r = \text{extract}_r(g, P) \end{cases}$$

**LEMMA 22.** Let  $P \vdash^{\text{npf}} t \sim t'$ . There exists  $P'$  such that  $P' \vdash^{\text{npf}} t \sim t'$  and for all  $h \in \text{index}(P')$  with  $h \neq \epsilon$ , for all  $x \in \{l, r\}$ , if we let  $h = h_x$  and  $P^h = \text{extract}_x(h, P')$  be the proof of  $b^h \sim b'^h$  then for all  $l \in \text{label}(P^h)$ :

- The proof  $P^h$  does not use the  $\{\overline{\text{BFA}}(b, b')\}$  rules.
- $\text{cs-path}^{h, l}(t, P)$  (resp.  $\text{cs-path}^{h, l}(t', P)$ ) does not contain two occurrences of the same condition.
- For all  $\gamma \leq_l^{h, l}(t, P')$ ,  $(b^h \downarrow_R) \in \text{leave-st}(\gamma \downarrow_R)$  and for all  $\gamma' \leq_l^{h, l}(t', P')$ ,  $(b'^h \downarrow_R) \in \text{leave-st}(\gamma' \downarrow_R)$ .
- For all  $\beta \leq_c^{\epsilon, l}(t, P')$ ,  $\text{leave-st}(\beta \downarrow_R) \cap \text{cs-path}^{\epsilon, l}(t, P) = \emptyset$  (same for  $t'$ ).
- For all  $\gamma \leq_l^{\epsilon, l}(t, P')$ ,  $\text{leave-st}(t \downarrow_R) \cap \text{leave-st}(\gamma \downarrow_R) \neq \emptyset$  (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  $P \vdash^{\text{npf}} t \sim t'$  and for all  $l \in \text{label}(P)$ ,  $h \in \text{index}(P)$ ,  $x \in \{l, r\}$  we have:

$$\forall \beta \in \left( (\leq_c^{h_x, l} \cup \leq_{cs}^{h_x, l})(t, P) \right) \cup \left( (\leq_c^{h_x, l} \cup \leq_{cs}^{h_x, l})(t', P) \right), \quad \{\text{false}, \text{true}\} \cap \text{leave-st}(\beta \downarrow_R) = \emptyset \quad (19)$$

First, we rewrite the proof  $P$  so that all CS applications are of the form:

$$\frac{b, (u_i)_i \sim b', (u'_i)_i \quad b, (v_i)_i \sim b', (v'_i)_i}{(\text{if } b \text{ then } u_i \text{ else } v_i)_i \sim (\text{if } b' \text{ then } u'_i \text{ else } v'_i)_i} \text{CS} \quad (20)$$

We prove by induction on  $n$ , starting with the inner-most CS conditions, that there exists  $P$  such that  $P \vdash^{\text{npf}} t \sim t'$ , (19) is true for  $P$  and the following properties hold for all  $h, h' \in \text{index}(P)$ :

- (i) If  $\text{if-depth}_P(h) \geq n$  then the  $\text{extract}_l(h, P)$  and  $\text{extract}_r(h, P)$  do not use the  $\{\overline{BFA}(b, b')\}$  rules.
- (ii) If  $\text{if-depth}_P(h) \geq n$  then for all  $x, l$ ,  $\text{cs-path}^{h_x, l}(t, P)$  and  $\text{cs-path}^{h_x, l}(t', P)$  do not contain two occurrences of the same condition.
- (iii) If  $\text{if-depth}_P(h) \geq n$  then for all  $x$ , if  $\text{extract}_x(h, P)$  is the proof of  $b \sim b'$  then for all  $l$ , for all  $\gamma \leq_l^{h_x, l}(t, P)$ ,  $(b \downarrow_R) \in \text{leave-st}(\gamma \downarrow_R)$  and for all  $\gamma' \leq_l^{h_x, l}(t', P)$ ,  $(b' \downarrow_R) \in \text{leave-st}(\gamma' \downarrow_R)$ .
- (iv) If  $\text{if-depth}_P(h) < n$  then for all  $h, h' \in \text{index}(P)$  such that  $h \leq h'$ , if we let  $h''$  be such that  $h' = h \cdot h''$  and  $x$  be such that  $h'' \in \text{index}(\text{extract}_x(h, P))$ , then for all  $x'$ , for all  $l \in \text{label}(\text{extract}_{x'}(h', P))$ , we have

$$\delta_{\text{cs-path}}^{h_x, l}(t, P) \supseteq \delta_{\text{cs-path}}^{h_{x'}, l}(t, P)$$

Let  $n_{\max}$  be the maximal if-depth in the proof of  $t \sim t'$ :

$$n_{\max} = \max_{h \in \text{index}(P)} \text{if-depth}_P(h)$$

*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  $\text{if-depth}_P(h) \geq n_{\max} + 1$ ; 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  $l \in \text{label}(P)$  and  $h_0 \in \text{h-branch}(l)$  such that  $\text{if-depth}_P(h_0) = n$ . Let  $x_0 \in \{l, r\}$  and  $h_0 = h_{0x_0}$ . We start by showing that for all  $l$ , for all  $\beta \leq_c^{h_0, l}(t, P)$ , if there exists  $b \in \text{cs-path}^{h_0, l}(t, P)$  such that  $b \in \text{leave-st}(\beta \downarrow_R)$  then there exists  $(b, b') \in \text{cs-path}_{\sim}^{h_0, l}(t, P)$  and  $(\beta, \beta') \leq_{c \sim c}^{h_0, l}(t \sim t', P)$  s.t.:

- $b' \in \text{leave-st}(\beta' \downarrow_R)$ .
- 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  $\delta \vec{\rho} \subseteq \delta \text{cs-path}^{h_0, l}(t, P)$  (resp.  $\delta \vec{\rho}' \subseteq \delta \text{cs-path}^{h_0, l}(t, P)$ ).

This is described in Figure 16.

Let  $\beta \leq_c^{h_0, l}(t, P)$  and  $b \in \text{cs-path}^{h_0, l}(t, P)$  such that  $b \in \text{leave-st}(\beta \downarrow_R)$ . We know that there exists  $b'$  and  $\beta'$  such that  $(b, b') \in \text{cs-path}_{\sim}^{h_0, l}(t, P)$  and  $(\beta, \beta') \leq_{c \sim c}^{h_0, l}(t \sim t', P)$ .

Let  $h \in \text{cs-pos}(\text{extract}_{x_0}(h_0, P))$  and  $x$  be the direction taken in  $l$  at  $h$  be such that  $\text{extract}(h, P)$  is the rule  $\text{CS}_{\square}(b, b')$ . We know that  $\text{extract}_x(h, P)$  is a proof of  $a \sim a'$ , where  $a =_R b$  and  $a' =_R b'$ . As  $\text{if-depth}(h) = n + 1$  we know by induction hypothesis (i) that  $\text{extract}_x(h, P)$  does not use  $\{\text{BFA}(b, b')\}$ . Hence the set  $\leq_{\perp}^{\epsilon, l}(a, \text{extract}_x(h, P))$  is the singleton  $\{\gamma_l\}$  and the set  $\leq_{\perp}^{\epsilon, l}(a', \text{extract}_x(h, P))$  is the singleton  $\{\gamma'_l\}$ . Let  $H = \text{index}(\text{extract}_x(h, P))$ , we have:

$$a \equiv C[(b^g)_{g \in H} \diamond (\gamma_{l_a})_{l_a}] \quad a' \equiv C[(b'^g)_{g \in H} \diamond (\gamma'_{l'_a})_{l'_a}]$$

By induction hypothesis (iii) we know that  $b \in \text{leave-st}(\gamma_l \downarrow_R)$  and  $b' \in \text{leave-st}(\gamma'_l \downarrow_R)$ .  $\gamma_l$  and  $\beta$  are  $S_l$ -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  $\mathfrak{F}(\text{FA}_s^* \cdot \text{Dup}^* \cdot \overline{\text{CA}}_2)$ . Therefore, using Lemma 2, we get that  $\beta' \equiv \gamma'_l$ . Since  $b' \in \text{leave-st}(\gamma'_l \downarrow_R)$ , we deduce that  $b' \in \text{leave-st}(\beta' \downarrow_R)$ . This concludes the proof of the first bullet point.

We now prove the second bullet point. By induction hypothesis (iv) we know that:

$$\delta \text{cs-path}^{h_0, l}(t, P) \supseteq \delta \text{cs-path}^{h_x, l}(t, P) \quad \delta \text{cs-path}^{h_0, l}(t', P) \supseteq \delta \text{cs-path}^{h_x, l}(t', P)$$

By definition of  $\vec{\rho}$ ,  $\text{cond-st}(\gamma_l \downarrow_R) = \text{cond-st}(\beta \downarrow_R) \supseteq \vec{\rho}$ . We can do better, and obtained an inclusion in the *directed* condition path. First, we know that:

- $a \equiv C[(b^g)_{g \in H} \diamond (\gamma_{l_a})_{l_a}]$ ,  $a =_R b$  and  $b$  is if-free and in  $R$ -normal form.
- Invariant (ii) holds, hence  $\delta \text{cs-path}^{h_x, l}(t, P)$  does not contain two occurrences of the same condition.
- $\delta \text{cs-path}^{h_x, l}(t, P)$  does not contain true and false.

The existence of a decomposition as described above is invariant by (chunks of)  $\rightarrow_{R > u}$  reductions, for a well-chosen ordering  $>_u$ . At the end of the reduction, we have  $b$ . By looking at the reduction backward, we see that  $b$  is a leaf of  $\gamma_l \downarrow_{R > u}$ , such that the directed path  $\delta \vec{\rho}$  from the root of  $\gamma_l \downarrow_{R > u}$  to  $b$  is included in the path from the root of  $a$  to  $\gamma_l$ .

We deduce that  $\delta \vec{\rho} \subseteq \delta \text{cs-path}^{h_x, l}(t, P)$ . By consequence,  $\delta \vec{\rho} \subseteq \delta \text{cs-path}^{h_0, l}(t, P)$ . Similarly we show that  $\delta \vec{\rho}' \subseteq \delta \text{cs-path}^{h_0, l}(t', P)$ .

*Step 2.* By doing some proof cut elimination, we can guarantee that for all  $l$ , for all  $\beta \leq_c^{h_0, l}(t, P)$ :

$$\text{leave-st}(\beta \downarrow_R) \cap \text{cs-path}^{h_0, l}(t, P) = \emptyset$$

Assume this is not the case: using **Step 1** we have:

$$\delta \vec{\rho} \subseteq \delta \text{cs-path}^{h_0, l}(t, P) \quad \delta \vec{\rho}' \subseteq \delta \text{cs-path}^{h_0, l}(t', P)$$

Therefore we can rewrite  $\beta$  and  $\beta'$  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.  $x = l$ ), and false if we are on the else branch (i.e.  $x = r$ ). 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  $(a, a'), (b, b') \leq_{cs-cs}^{h_0} (t, P)$  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  $Q \vdash^{npf} a, a \sim a', b'$ . 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  $b^{h_0}, b'^{h_0}$  be such that  $\text{extract}_{x_0}(h, P) \vdash^{npf} b^{h_0} \sim b'^{h_0}$ . We know that:

$$b^{h_0} \equiv C \left[ \left( \boxed{b^{h_l} \ b^{h_r}} \right)_{h \in H^{h_0}} \diamond \left( D_l^{h_0} \left[ (\beta)_{\beta \leq_c^{h_0, l}(t, P)} \diamond (\gamma)_{\gamma \leq_l^{h_0, l}(t, P)} \right] \right)_{l \in L^{h_0}} \right]$$

where  $H^{h_0} = \text{cs-pos}(\text{extract}_{x_0}(h_0, P))$  and  $L^{h_0} = \text{label}(\text{extract}_{x_0}(h_0, P))$ .

To prove that for all  $l$ , for all  $\gamma \leq_l^{h_0, l}(t, P)$ , we have  $b^{h_0} \downarrow_R \in \text{leave-st}(\gamma \downarrow_R)$ , 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  $\gamma' \leq_l^{h_0, l}(t', P)$  we have  $b'^{h_0} \downarrow_R \in \text{leave-st}(\gamma' \downarrow_R)$ ):

- (21.i): the only difficulty lies in proving that for all  $\beta \leq_c^{h_0, l}(t, P)$ ,  $\text{cond-st}(\beta \downarrow_R) \cap \text{leave-st}(\beta \downarrow_R) = \emptyset$ , 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  $(\text{cs-path}^{h_0, l}(t, P))^2$  this was shown in **Step 2 b**. For couples of positions in  $D_l^{h_0} \times D_l^{h_0}$  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  $\beta_0, \beta_1 \leq_c^{h_0, l}(t, P)$  on the same branch such that  $\text{leave-st}(\beta_0) \cap \text{leave-st}(\beta_1) \neq \emptyset$ . By Proposition 17 we know that  $\beta_0 \equiv \beta_1$ . Let  $\beta'_0, \beta'_1$  be the conditions at positions, respectively,  $p$  and  $p'$  in  $b'^{h_0}$ . We know that  $(\beta_0, \beta'_0), (\beta_1, \beta'_1) \leq_c^{h_0, l}(t \sim t', P)$ . We can extract from  $P$  a proof of:

$$\beta_0, \beta_0 \sim \beta'_0, \beta'_1$$

in  $\mathfrak{F}(\text{FA}_s^* \cdot \text{Dup}^* \cdot \overline{\text{CCA}_2})$ , 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  $\beta_1$  and  $\beta'_1$  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  $\text{cs-path}^{h_0, l}(t, P) \times D_l^{h_0}$ .

- (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  $\{\overline{\text{BFA}}(b, b')\}$  applications.

Using Corollary 2.(ii) and the proof  $Q$  constructed at **Step 3**, we know that for all  $\gamma, \gamma' \leq_l^{h_0, l}(t, Q)$ ,  $\gamma \equiv \gamma'$  (and the same holds for  $(t', Q)$ ). Therefore there is a proof cut elimination that allows us to remove all  $\{\overline{\text{BFA}}(b, b')\}$  applications, by rewriting:

$$D_l \left[ - \diamond (\gamma)_{\gamma \leq_l^{h_0, l}(t, Q)} \right] \quad \text{and} \quad D_l \left[ - \diamond (\gamma')_{\gamma' \leq_l^{h_0, l}(t', Q)} \right]$$



into, respectively,  $\gamma_0$  and  $\gamma'_0$  (where  $\gamma_0 \leq_1^{h_0, l}(t, Q)$  and  $\gamma'_0 \leq_1^{h_0, l}(t', Q)$ ).

*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**).  $\square$

## H BOUNDING THE BASIC TERMS

### H.1 $\alpha$ -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  $(t, P)$ - $\alpha$ -bounded terms, where  $P \vdash^{\text{npf}} t \sim t'$ , and then prove that  $(t, P)$ - $\alpha$ -bounded terms are of bounded size w.r.t.  $|t \downarrow_R| + |t' \downarrow_R|$ . Basically, a term  $\beta$  in  $\leq_{\text{bt}}^{h, l}(t, P)$  or  $\text{cs-path}^{h, l}(t, P)$  is  $(t, P)$ - $\alpha$ -bounded if we are in one of the following case:

- $\beta$  is a normalized basic term, and  $\beta$  has a leaf term appearing in  $\text{st}(t \downarrow_R)$ . Since  $\beta$  is uniquely characterized by its leaf terms, this bound  $\beta$ .
- Let  $\beta'$  be the term matching  $\beta$  on the *right*. If  $\beta'$  shares a leaf term with  $\text{st}(t' \downarrow_R)$ , then, by the previous observation,  $\beta'$  is bounded. Since  $\beta$  and  $\beta'$  differ only by the content of their encryptions, this also bound  $\beta$ .
- If  $\beta$  is a case-study condition (i.e. in  $\text{cs-path}^{h, l}(t, P)$ ), and if there exists a  $(t, P)$ - $\alpha$ -bounded normalized basic term  $\varepsilon$  such that  $\beta$  appears in  $\varepsilon$ 's leaf terms. Indeed, since  $\varepsilon$  is bounded, it has finitely many leaf terms, which are of bounded size. Hence  $\beta$  is also of bounded size.
- If  $\beta$  is a normalized basic terms used in the sub-proof of  $b \sim b'$ , where  $b$  and  $b'$  are  $(t, P)$ - $\alpha$ -bounded case-study conditions, and if  $b$  appears in  $\beta$ 's leaf terms. Again, since  $\beta$  is uniquely characterized by any of its leaf terms, and since  $b$  is bounded, we know that  $\beta$  is bounded.
- Finally, if  $\beta$  is a decryption guard of some decryption oracle call  $d$ , where  $d$  appears in a  $(t, P)$ - $\alpha$ -bounded normalized basic term  $\zeta$ . Since  $\zeta$  is bounded, and since  $\beta$  is a sub-term of  $\zeta$ , the term  $\beta$  is also bounded.

We formally define what is a  $(t, P)$ - $\alpha$ -bounded terms.

*Definition 67.* For all  $P \vdash^{\text{npf}} t \sim t'$ , the set of  $(t, P)$ - $\alpha$ -bounded terms is the smallest subset of:

$$\{\beta \mid \exists h, l. \beta \leq_{\text{bt}}^{h, l}(t, P)\} \cup \{b \mid \exists h. b \in \text{cs-path}^{h, l}(t, P)\}$$

such that for all  $h, l$ , for all  $\beta \in \leq_{\text{bt}}^{h, l} \cup \text{cs-path}^{h, l}(t, P)$ ,  $\beta$  is  $(t, P)$ - $\alpha$ -bounded if:

- **Base case:**  $h = \epsilon$  and  $\text{leave-st}(\beta \downarrow_R) \cap \text{st}(t \downarrow_R) \neq \emptyset$ .
- **Base case:**  $h = \epsilon$  and there exists  $\beta'$  such that:

$$(\beta, \beta') \in (\leq_{\text{bt}}^{\epsilon, l} \cup \leq_{\text{c-c}}^{\epsilon, l} \cup \text{cs-path}^{\epsilon, l})(t \sim t', P)$$

and  $\text{leave-st}(\beta' \downarrow_R) \cap \text{st}(t' \downarrow_R) \neq \emptyset$ .

- **Inductive case, same label:**  $\beta \in \text{cs-path}^{h, l}(t, P)$  and there exists  $\varepsilon \leq_{\text{bt}}^{h, l}(t, P)$  such that  $\varepsilon$  is  $(t, P)$ - $\alpha$ -bounded and  $\beta \in \text{leave-st}(\varepsilon \downarrow_R)$ .
- **Inductive case, different labels:**  $\beta \leq_{\text{bt}}^{h, l}(t, P)$ , there exists  $h'$  such that  $h \in \text{cs-pos}(h')$  and  $b \in \text{cs-path}^{h', l}(t, P)$  such that  $b$  is  $(t, P)$ - $\alpha$ -bounded and  $\beta \in \text{leave-st}(\beta \downarrow_R)$ .
- **Inductive case, guard:**  $\beta \leq_{\text{bt}}^{h, l}(t, P)$ , there exists  $\varepsilon \leq_{\text{bt}}^{h, l}(t, P)$  such that:
  - $\varepsilon \equiv B[\vec{w}, (\alpha_i)_i, (\text{dec}_j)_j]$  is  $(t, P)$ - $\alpha$ -bounded.
  - $\beta$  is a guard of a  $\mathcal{S}_i^P$ -decryption oracle call  $d \in (\text{dec}_j)_j$ .

We continue our proof cut eliminations, starting from the derivations constructed in Lemma 22. We let  $P \vdash_{\alpha}^{\text{npf}} t \sim t'$  be the restriction of  $\vdash^{\text{npf}}$  to derivations satisfying the properties guaranteed by Lemma 22 which use only  $(t, P)$ - $\alpha$ -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  $P \vdash_{\alpha}^{\text{npf}} t \sim t'$  if:

- (I)  $P \vdash^{\text{npf}} t \sim t'$  and the properties (a) to (e) of Lemma 22 hold.
- (II) The following sets are sets of, respectively,  $(t, P)$ - $\alpha$ -bounded and  $(t', P)$ - $\alpha$ -bounded terms:

$$\{\beta \mid \exists h, l. \beta \leq_{\text{bt}}^{h, l}(t, P)\} \cup \{b \mid \exists h. b \leq_{\text{cs}}^h(t, P)\} \\ \{\beta' \mid \exists h, l. \beta' \leq_{\text{bt}}^{h, l}(t', P)\} \cup \{b' \mid \exists h. b' \leq_{\text{cs}}^h(t', P)\}$$

- (III) For every  $l \in \text{label}(\epsilon)$ , for every path  $\vec{\rho}$  of  $S_l^P$ -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).  $\vdash_{\alpha}^{\text{npf}}$  is complete for  $\vdash^{\text{npf}}$ .

PROOF. Let  $P$  be such that  $P \vdash^{\text{npf}} t \sim t'$ , 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  $\beta$  in:

$$\{\beta \mid \exists h, l. \beta \leq_{\text{bt}}^{h, l}(t, P)\} \cup \{b \mid \exists h. b \leq_{\text{cs}}^h(t, P)\}$$

such that  $\beta$  is not  $(t, P)$ - $\alpha$ -bounded, then there is a cut elimination removing  $\beta$  (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  $(t, P)$ - $\alpha$ -bounded, hence we will conclude by induction. First, we want to pick a term  $\beta$  maximal for a carefully chosen relation.

*Order  $<_g$ .* Let  $<_g$  be the transitive closure of the relation  $\ll_g$  on:

$$\bigcup_{h \in \text{index}(P)} \{(\beta, h) \mid \exists l. \beta \leq_{\text{bt}}^{h, l}(t, P)\} \cup \bigcup_{h \in \text{index}(P)} \{(b, h) \mid \exists l. b \in \text{cs-path}^{h, l}(t, P)\}$$

defined by:

$$(\zeta, h) \ll_g (\zeta', h') \text{ iff } \begin{cases} h = h' \wedge \zeta, \zeta' \leq_{\text{bt}}^{h, l}(t, P) \wedge \zeta \text{ is a guard of a dec. oracle call } d \in \text{st}(\zeta') \\ h = h' \wedge \zeta \in \text{cs-path}^{h, l}(t, P) \wedge \zeta' \leq_{\text{bt}}^{h, l}(t, P) \wedge \zeta \in \text{leave-st}(\zeta' \downarrow_R) \\ h > h' \wedge \zeta \leq_{\text{bt}}^{h, l}(t, P) \wedge \zeta' \in \text{cs-path}^{h', l}(t, P) \wedge \zeta' \in \text{leave-st}(\zeta \downarrow_R) \end{cases}$$

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  $<_g^h$  of  $<_g$  to:

$$\{(\beta, h) \mid \exists l. \beta \leq_{\text{bt}}^{h, l}(t, P)\} \cup \{(b, h) \mid \exists l. b \in \text{cs-path}^{h, l}(t, P)\}$$

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  $h$ ,  $<_g^h$  is a strict order and that when we go from the domain of  $<_g^h$  to the domain of  $<_g^{h'}$ , we have  $h' > h$ .

W.l.o.g. we assume that  $(\beta, h)$  is maximal for  $<_g$  among the set of terms that are not  $(t, P)$ - $\alpha$ -bounded. Consider an arbitrary  $l$  such that  $h \in \text{h-branch}(l)$ . Since  $\beta$  is not  $(t, P)$ - $\alpha$ -bounded, we know that if  $\beta$  is a guard of some decryption oracle call  $d \in \text{st}(\zeta)$  with  $\zeta \leq_{\text{bt}}^{h, l}(t, P)$ , then  $\zeta$  is not

$(t, P)$ - $\alpha$ -bounded. By maximality of  $\beta$ , it follows that if  $\beta \leq_{\text{bt}}^{h,l}(t, P)$  then  $\beta$  is not a decryption guard of any  $\zeta \leq_{\text{bt}}^{h,l}(t, P)$ .

Case  $h = \epsilon$ . First we are going to describe what to do for  $h = \epsilon$ . From Lemma 22.(e), we know that for every  $l \in \text{label}(P)$ , for all  $\gamma \leq_1^{\epsilon,l}(t, P)$ , the basic term  $\gamma$  is  $(t, P)$ - $\alpha$ -bounded. Therefore  $\beta \not\leq_1^{\epsilon,l}(t, P)$ . Moreover, from Lemma 22.(d) we get that  $\beta \leq_c^{\epsilon,l}(t, P)$  and  $\beta \in \text{cs-path}^{\epsilon,l}(t, P)$  are mutually exclusive. Putting everything together, we have three cases:

- (i) either  $\beta (\not\leq_1^{\epsilon,l} \cup \leq_c^{\epsilon,l})(t, P)$  and  $\beta \notin \text{cs-path}^{\epsilon,l}(t, P)$ .
- (ii) or  $\beta (\not\leq_1^{\epsilon,l} \cup \leq_c^{\epsilon,l})(t, P)$  and  $\beta \in \text{cs-path}^{\epsilon,l}(t, P)$ .
- (iii)  $\beta (\not\leq_1^{\epsilon,l} \cup \leq_c^{\epsilon,l})(t, P)$  and  $\beta \notin \text{cs-path}^{\epsilon,l}(t, P)$ .

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  $\beta'$  be such that  $(\beta, \beta') (\leq_{c \sim c}^{\epsilon,l})(t \sim t', P)$ . Since  $\beta$  is not  $(t, P)$ - $\alpha$ -bounded we know that for all  $u \in \text{leave-st}(\beta \downarrow_R)$ , for all  $u' \in \text{leave-st}(\beta' \downarrow_R)$ ,  $u$  and  $u'$  are spurious in, respectively,  $t$  and  $t'$ . We let:

$$\begin{aligned} t &\equiv C[\vec{b}_{cs} \diamond D_l [(\beta_i)_{i \in J} \diamond (Y_m)_{m \in M}], \Delta] \\ t' &\equiv C[\vec{b}'_{cs} \diamond D_l [(\beta'_i)_{i \in J} \diamond (Y'_m)_{m \in M}], \Delta'] \end{aligned}$$

where, for every  $i \in J$ ,  $(\beta_i, \beta'_i) \leq_{c \sim c}^{\epsilon,l}(t \sim t', P)$ , and for every  $m \in M$ ,  $(Y_m, Y'_m) \leq_{| \sim |}^{\epsilon,l}(t \sim t', P)$ . Moreover, we assume that for every  $i \in J$ , the hole  $[]_i$  (which is mapped to  $\beta_i$ ) 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:

$$I = \{i \in J \mid \text{leave-st}(\beta \downarrow_R) \cap \text{leave-st}(\beta_i \downarrow_R) \neq \emptyset\}$$

We know that we have a proof of  $(\beta_i)_{i \in I} \sim (\beta'_i)_{i \in I}$  in the fragment  $\mathfrak{F}(\text{FA}_s^* \cdot \text{Dup}^* \cdot \overline{\text{CCA}_2})$ . Therefore:

$$\forall b, b' \in \{\beta'_i \mid i \in I\}, b \equiv b' \equiv \beta' \quad (21)$$

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  $I' = \{i \in J \mid \text{leave-st}(\beta' \downarrow_R) \cap \text{leave-st}(\beta'_i \downarrow_R) \neq \emptyset\}$ . We deduce from this that:

$$\forall b, b' \in \{\beta_i \mid i \in I'\}, b \equiv b' \equiv \beta \quad (22)$$

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  $\beta$  we know that there exists  $\tilde{\beta}_i[]$  such that:

$$\tilde{\beta}_i[\beta] \equiv \beta_i \quad \text{and} \quad \text{leave-st}(\beta \downarrow_R) \cap \text{cond-st}(\tilde{\beta}_i[] \downarrow_R) = \emptyset$$

Similarly, for every  $m \in M$ , there exists  $\tilde{Y}_m[]$  such that:

$$\tilde{Y}_m[\beta] \equiv Y_m \quad \text{and} \quad \text{leave-st}(\beta \downarrow_R) \cap \text{cond-st}(\tilde{Y}_m[] \downarrow_R) = \emptyset$$

Then we have:

$$\begin{aligned} t &\equiv C[\vec{b}_{cs} \diamond (D_l [(\beta_i)_{i \in J} \diamond (Y_m)_{m \in M}], \Delta)] \\ &\equiv C[\vec{b}_{cs} \diamond (D_l [((\beta)_{i \in I}, (\tilde{\beta}_i[\beta])_{i \notin I}) \diamond (\tilde{Y}_m[\beta])_{m \in M}], \Delta)] \end{aligned}$$

Let  $C_\beta[\vec{b}_\beta \diamond \vec{u}_\beta] \equiv \beta \downarrow_R$ . We have:

$$\begin{aligned} & D_l \left[ \left( (\beta)_{i \in I}, (\tilde{\beta}_i[\beta])_{i \notin I} \right) \diamond (\tilde{y}_m[\beta])_{m \in M} \right] \\ =_R & \text{ if } C_\beta[\vec{b}_\beta \diamond \vec{u}_\beta] \text{ then } D_l \left[ \left( (\text{true})_{i \in I}, (\tilde{\beta}_i[\text{true}])_{i \notin I} \right) \diamond (\tilde{y}_m[\text{true}])_{m \in M} \right] \\ & \text{ else } D_l \left[ \left( (\text{false})_{i \in I}, (\tilde{\beta}_i[\text{false}])_{i \notin I} \right) \diamond (\tilde{y}_m[\text{false}])_{m \in M} \right] \end{aligned}$$

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  $u \notin \text{cond-st}(\tilde{y}_m[\ ] \downarrow_R)$ . 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:

$$\begin{aligned} & C \left[ \vec{b}_{cs} \diamond \left( \begin{array}{l} \text{if } C_\beta[\vec{b}_\beta \diamond \vec{u}_\beta] \text{ then } D_l \left[ \left( (\text{true})_{i \in I}, (\tilde{\beta}_i[\text{true}])_{i \notin I} \right) \diamond (\tilde{y}_i[\text{true}])_m \right] \\ \text{else } D_l \left[ \left( (\text{false})_{i \in I}, (\tilde{\beta}_i[\text{false}])_{i \notin I} \right) \diamond (\tilde{y}_i[\text{false}])_m \right] \end{array} \right), \Delta \right] \\ =_R & C \left[ \vec{b}_{cs} \diamond \left( \begin{array}{l} \text{if } C_\beta[\vec{b}_\beta \diamond (\text{true})_o] \text{ then } D_l \left[ \left( (\text{true})_{i \in I}, (\tilde{\beta}_i[\text{true}])_{i \notin I} \right) \diamond (\tilde{y}_i[\text{true}])_m \right] \\ \text{else } D_l \left[ \left( (\text{false})_{i \in I}, (\tilde{\beta}_i[\text{false}])_{i \notin I} \right) \diamond (\tilde{y}_i[\text{false}])_m \right] \end{array} \right), \Delta \right] \\ =_R & C \left[ \vec{b}_{cs} \diamond \left( D_l \left[ \left( (\text{true})_{i \in I}, (\tilde{\beta}_i[\text{true}])_{i \notin I} \right) \diamond (\tilde{y}_i[\text{true}])_m \right], \Delta \right) \right] \end{aligned} \quad (23)$$

- **(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:

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

And, for every  $m \in M$ , there exists  $\tilde{y}'_m[\ ]$  such that:

$$\tilde{y}'_m[\beta'] \equiv \gamma'_m \quad \text{and} \quad \text{leave-st}(\beta' \downarrow_R) \cap \text{cond-st}(\tilde{y}'_m[\ ] \downarrow_R) = \emptyset$$

Then by the same reasoning we have:

$$\begin{aligned} t' & \equiv C \left[ \vec{b}'_{cs} \diamond \left( D_l \left[ (\beta'_i)_i \diamond (\gamma'_m)_{m \in M} \right], \Delta' \right) \right] \\ & \equiv C \left[ \vec{b}'_{cs} \diamond \left( D_l \left[ \left( (\beta')_{i \in I}, (\tilde{\beta}'_i[\beta'])_{i \notin I} \right) \diamond (\tilde{y}'_m[\beta'])_{m \in M} \right], \Delta' \right) \right] \\ & =_R C \left[ \vec{b}'_{cs} \diamond \left( D_l \left[ \left( (\text{true})_{i \in I}, (\tilde{\beta}'_i[\text{true}])_{i \notin I} \right) \diamond (\tilde{y}'_m[\text{true}])_{m \in M} \right], \Delta' \right) \right] \end{aligned} \quad (24)$$

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  $\text{cca}_2$  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  $\beta \not\leq_c^{h,l}(t, P)$ . We have three cases:

- either  $\beta \leq_l^{h,l}(t, P)$ : using Lemma 22.(c), there exists  $h_0, b^h$  such that  $h \in \text{cs-pos}(h_0)$ ,  $b^h \in \text{cs-path}^{h_0,l}(t, P)$  and  $(b^h \downarrow_R) \in \text{leave-st}(\beta \downarrow_R)$ . Since  $h \in \text{cs-pos}(h_0)$  implies that  $h_0 < h$ , we know that  $\beta <_g b^h$ . We then have two cases. Either  $b^h$  is  $(t, P)$ - $\alpha$ -bounded, and then using the inductive case for different labels of the definition of  $(t, P)$ - $\alpha$ -bounded terms, we know that  $\beta$

is  $(t, P)$ -abounded. Absurd. Or  $b^h$  is not  $(t, P)$ - $\alpha$ -bounded, which contradicts the maximality of  $\beta$  among the set of terms which are not  $(t, P)$ -abounded. Absurd.

- (b) either  $\beta \not\leq_l^{h,l}(t, P)$  and  $\beta \in \text{cs-path}^{h,l}(t, P)$ : this case is done exactly like case (ii).
- (c) either  $\beta \not\leq_l^{h,l}(t, P)$  and  $\beta \notin \text{cs-path}^{h,l}(t, P)$ : this case is done exactly like case (iii).

*Valid Proof Rewriting.* We do the rewritings described above for every  $h$  such that  $(\beta, h)$  is maximal for  $<_g$ , and for every  $l$  such that  $\beta \leq_{bt}^{h,l}(t, P)$  or  $\beta \in \text{cs-path}^{h,l}(t, P)$ , *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  $\text{CCA}_2$  axiom hold. This is tedious, but here are the key ingredients:

- $\beta$  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  $(\beta, h)$  is maximal for  $<_g$ . Consider  $h'$  such that  $(\beta, h')$  is not maximal for  $<_g$ : then there exists  $h$  such that  $(\beta, h)$  is maximal for  $<_g$  and  $h < h'$ . 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  $\beta$ . Therefore, if an encryption used to contain  $\beta$  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  $\text{EQL}(\_, \_)$ .

*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  $S_l^P$ -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  $(\beta, \beta'_0) \leq_{c \sim c}^{\epsilon, l}(t, P)$  and  $(\beta, \beta'_1) \leq_{c \sim c}^{\epsilon, l}(t, P)$ , using Lemma 2 we have  $\beta'_0 \equiv \beta'_1$ . Since they are on the same branch, one may rewrite the lowest occurrence of  $\beta$  and  $\beta'_0$  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.  $\square$

## 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  $(t, P)$ - $\alpha$ -bounded terms, one may try to show that for every  $\beta \in (\leq_{bt}^{h,l}(t, P) \cup \text{cs-path}^{h,l}(t, P))$ , if  $\beta$  is  $(t, P)$ - $\alpha$ -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  $(\beta, \beta') \leq_c^{h,l}(t \sim t', P)$  such that  $\beta'$  has a leaf term appearing in  $t'$ , but  $\beta$  shares no leaf term with  $\beta'$  nor  $t$ :

$$\text{leave-st}(\beta \downarrow_R) \cap \text{leave-st}(\beta' \downarrow_R) = \emptyset \quad \text{leave-st}(\beta \downarrow_R) \cap \text{st}(t \downarrow_R) = \emptyset$$

$$\text{leave-st}(\beta' \downarrow_R) \cap \text{st}(t' \downarrow_R) \neq \emptyset$$

$\beta'$  is  $\alpha$ -bounded since it shares a leaf term with  $t'$ , and using the second case,  $\beta$  is  $\alpha$ -bounded too. But  $\beta$  shares no leaf term with  $t$  and  $t'$ .

Still, we can bound  $\beta$ . Since  $(\beta, \beta') \leq_c^{h,l}(t \sim t', P)$ , 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  $\beta$  is a  $S_l$ -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  $S_l$ -encryption oracle calls*. This motivates the introduction of the notion of *leaf frame*.

*Leaf frame.* Let  $\beta$  be a  $S_l$ -normalized basic term, and  $u, v \in \text{leave-st}(\beta \downarrow_R)$  be leaf terms of  $\beta$ . Then  $u$  and  $v$  only differ by their encryptions. That is, if one replaces all the zero decryptions  $\mathbf{0}(\text{dec}(\_, \text{sk}))$  by  $\text{dec}(\_, \text{sk})$ , and all the leaves of encryptions  $\{m\}_{\text{pk}}^n$  by  $\{\llbracket \alpha \rrbracket_{\text{pk}}^n$  (where  $\alpha$  is the unique term of  $\mathcal{E}_l$  such that  $\alpha \equiv \{\_\}_{\text{pk}}^n$ ) 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  $P \vdash_{\alpha}^{\text{npf}} t \sim t'$  and  $l$  be a branch label in  $\text{label}(P)$ . We define the left *leaf frame*  $\text{l-frame}_l^P$  of  $\beta \in (\leq_{\text{bt}}^{\text{h},l}(t, P) \cup \text{cs-path}^{\text{h},l}(t, P))$  inductively as follows:

$$\text{l-frame}_l^P(s) \equiv \begin{cases} \{\llbracket \alpha \rrbracket_{\text{pk}}^n & \text{if } \exists \alpha \equiv \{m\}_{\text{pk}}^n \in \mathcal{E}_l^P \wedge s \equiv \{\_\}_{\text{pk}}^n \\ \text{dec}(\text{l-frame}_l^P(s), \text{sk}) & \text{if } \text{sk} \in \mathcal{K}_l^P \wedge s \equiv \mathbf{0}(\text{dec}(s, \text{sk})) \\ \text{l-frame}_l^P(v) & \text{if } s \equiv \text{if } b \text{ then } u \text{ else } v \\ f((\text{l-frame}_l^P(u_i))_i) & \text{otherwise} \end{cases}$$

We also let  $\underline{\text{l-frame}}_l^P(\beta)$  be  $\text{l-frame}_l^P(\beta)$  where we make every hole variable appear at most once, by replacing a hole variable  $\llbracket \alpha \rrbracket$  occurring at position  $p$  in  $\beta$  by  $\llbracket \alpha, p \rrbracket$ .

We define the right *leaf frame*  $\text{r-frame}_l^P$  (and its underlined version  $\underline{\text{r-frame}}_l^P$ ) of  $\beta \in (\leq_{\text{bt}}^{\text{h},l}(t', P) \cup \text{cs-path}^{\text{h},l}(t', P))$ , using  $\mathcal{E}_l^P$  instead of  $\mathcal{E}_l$ .

*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  $\underline{\text{l-frame}}_l^P(\beta)$  are annotated by both the position  $p$  of the hole *and* the encryption  $\alpha$  that appears at  $p$  in  $\beta$ . By consequence, if two normalized basic terms  $\beta$  and  $\beta'$  are such that  $\underline{\text{l-frame}}_l^P(\beta)$  and  $\underline{\text{l-frame}}_l^P(\beta')$  share a hole variable  $\llbracket \alpha, p \rrbracket$ , it means that  $\beta$  and  $\beta'$  contain the *same encryption  $\alpha$  at the same position  $p$* . This is crucial, as we want  $\underline{\text{l-frame}}_l^P$  to uniquely characterize normalized basic terms.  $\diamond$

*Example 24.* For all  $S_l^P$ -decryption oracle call  $\text{dec}$  guarding  $\text{dec}(s[(\alpha_i)_i, (\text{dec}_j)_j], \text{sk})$ , if for all  $i$ ,  $\alpha_i \equiv \{\_\}_{\text{pk}_i}^{n_i}$  then:

$$\text{l-frame}_l^P(\text{dec}) \equiv \text{dec}\left(s\left[\left(\{\llbracket \alpha_i \rrbracket_{\text{pk}_i}^{n_i}\rrbracket\right)_i, (\text{l-frame}_l^P(\text{dec}_j))_j\right], \text{sk}\right)$$

We also give an example of  $\underline{\text{l-frame}}_l^P$ . Assuming that  $\alpha_0 \equiv \{A\}_{\text{pk}}^{n_0}$  and  $\alpha_1 \equiv \{B\}_{\text{pk}}^{n_1}$  are encryptions in  $\mathcal{E}_l^P$ :

$$\underline{\text{l-frame}}_l^P(\langle \alpha_0, \langle \alpha_1, \alpha_0 \rangle \rangle) \equiv \langle \{\llbracket \alpha_0, 00 \rrbracket_{\text{pk}}^{n_0}\rrbracket, \langle \{\llbracket \alpha_1, 100 \rrbracket_{\text{pk}}^{n_1}\rrbracket, \{\llbracket \alpha_0, 110 \rrbracket_{\text{pk}}^{n_0}\rrbracket \rangle \rangle \rangle \quad \diamond$$

**PROPOSITION 27.** Let  $P \vdash_{\alpha}^{\text{npf}} t \sim t'$  and  $l \in \text{label}(P)$ . Let  $b$  be an if-free term in  $R$ -normal form. For every  $S_l$ -normalized basic terms  $\gamma$ , if  $b \in \text{leave-st}(\gamma \downarrow_R)$  then  $\text{l-frame}_l^P(b) \equiv \text{l-frame}_l^P(\gamma)$ .

**PROOF.** This is by induction on the size of  $\gamma$ .  $\square$

**PROPOSITION 28.** Let  $P \vdash_{\alpha}^{\text{npf}} t \sim t'$  and  $l \in \text{label}(P)$ . For every  $S_l$ -normalized basic terms  $\beta, \beta'$ , if  $\text{l-frame}_l^P(\beta) \equiv \text{l-frame}_l^P(\beta')$  then  $\beta \equiv \beta'$ .

**PROOF.** The proof is exactly the same than for Proposition 17.  $\square$

PROPOSITION 29. Let  $P \vdash_{\alpha}^{npf} t \sim t'$  and  $l \in \text{label}(P)$ . For all  $h$ , if  $(b, b') \leq_{cs \sim cs}^{h,l} (t \sim t', P)$  then there exists  $h'$  and  $(\gamma, \gamma') (\leq_{c \sim c}^{h',l} \cup \leq_{l \sim l}^{h',l}) (t \sim t', P)$  such that  $b \in \text{leave-st}(\gamma \downarrow_R)$  and  $b' \in \text{leave-st}(\gamma' \downarrow_R)$ .

PROOF. Let  $h, x$  be such that  $h = h_x$ . Let  $h_0 \in \text{cs-pos}(\text{extract}_x(h, P))$  and  $x_0$  be such that  $x_0$  is the direction taken in  $l$  at position  $h_0$ , and such that  $Q = \text{extract}_{x_0}(h_0, P)$  is a proof of  $b \sim b'$ .

Using the fact that the sub-proofs of  $\text{CS}_{\square}$  conditions of  $P$  do not use the  $\text{BFA}$  rule, we know that  $Q$  lies in the fragment:

$$\mathfrak{F}(\text{CS}_{\square} \cdot \text{FA}_s^* \cdot \text{Dup}^* \cdot \overline{\text{CCA}_2})$$

Let  $(\gamma, \gamma') \leq_{l \sim l}^{\epsilon, l} (b \sim b', Q)$ . Using the property (c) of Lemma 22 (which holds thanks to  $\vdash_{\alpha}^{npf}$ ), we know that  $b \in \text{leave-st}(\gamma \downarrow_R)$  and  $b' \in \text{leave-st}(\gamma' \downarrow_R)$ .  $\square$

PROPOSITION 30. Let  $P \vdash_{\alpha}^{npf} t \sim t'$  and  $l \in \text{label}(P)$ . For all  $h$ , if  $(\beta, \beta') (\leq_{c \sim c}^{h,l} \cup \leq_{l \sim l}^{h,l} \cup \text{cs-path}_{\sim}^{h,l}) (t \sim t', P)$  then  $\text{l-frame}_l^P(\beta) \equiv \text{r-frame}_l^P(\beta')$ .

PROOF. First we deal with the case  $(\beta, \beta') (\leq_{c \sim c}^{h,l} \cup \leq_{l \sim l}^{h,l}) (t \sim t', P)$ . We know that we can extract a proof  $Q$  (from  $P$ ) such that  $Q \vdash_{\alpha}^{npf} \beta \sim \beta'$  and  $Q$  is in the fragment  $\mathfrak{F}(\text{FA}_s^* \cdot \text{Dup}^* \cdot \overline{\text{CCA}_2})$ . The result follows from the definitions of  $\text{l-frame}_l^P$  and  $\text{r-frame}_l^P$ .

Now we deal with the case  $(\beta, \beta') (\text{cs-path}_{\sim}^{h,l}) (t \sim t', P)$ . Using Proposition 29 we know that there exists  $h'$  and  $(\gamma, \gamma') (\leq_{c \sim c}^{h',l} \cup \leq_{l \sim l}^{h',l}) (t \sim t', P)$  such that  $\beta \in \text{leave-st}(\gamma \downarrow_R)$  and  $\beta' \in \text{leave-st}(\gamma' \downarrow_R)$ . Since  $\beta$  is if-free and in  $R$ -normal form, we obtain that  $\text{l-frame}_l^P(\beta) \equiv \text{l-frame}_l^P(\gamma)$  by applying Proposition 27. Similarly  $\text{r-frame}_l^P(\beta') \equiv \text{r-frame}_l^P(\gamma')$ . Moreover, from the previous case, we get that  $\text{l-frame}_l^P(\gamma) \equiv \text{r-frame}_l^P(\gamma')$ . Hence  $\text{l-frame}_l^P(\beta) \equiv \text{r-frame}_l^P(\beta')$ .  $\square$

PROPOSITION 31. Let  $P \vdash_{\alpha}^{npf} t \sim t'$  and  $l \in \text{label}(P)$ . For every  $\mathcal{S}_l$ -normalized basic terms  $\beta, \beta'$ ,  $\text{l-frame}_l^P(\beta) \equiv \text{l-frame}_l^P(\beta')$  if and only if  $\underline{\text{l-frame}}_l^P(\beta) \equiv \underline{\text{l-frame}}_l^P(\beta')$ .

PROOF. This is obvious, since the hole variable annotations in  $\underline{\text{l-frame}}_l^P$  uniquely characterize both the position of the hole and the encryption appearing at this position.  $\square$

PROPOSITION 32. Let  $P \vdash_{\alpha}^{npf} t \sim t'$  and  $l \in \text{label}(P)$ . For every  $\mathcal{S}_l$ -normalized basic terms  $\beta, \beta'$  and substitutions  $\theta, \theta'$ , if  $\underline{\text{l-frame}}_l^P(\beta)\theta \equiv \underline{\text{l-frame}}_l^P(\beta')\theta'$  then  $\underline{\text{l-frame}}_l^P(\beta) \equiv \underline{\text{l-frame}}_l^P(\beta')$ .

PROOF. We prove this by induction on the size of  $\beta$ . The base case is trivial, let's deal with the inductive case. Let  $\beta$  and  $\beta'$  be  $\mathcal{S}_l^P$ -normalized basic terms, we know that  $\beta \equiv B[\vec{w}, (\alpha_i)_i, (\text{dec}_j)_j]$  where:

- for every  $i$ ,  $\alpha_i \equiv \{m_i\}_{pk_i}^{n_i} \in \mathcal{E}_l^P$ .
- for every  $j$ ,  $\text{dec}_j$  is a decryption oracle call for  $\text{dec}(s_j, sk_j)$  in  $\mathcal{D}_l^P$ .

Similarly, we have a decomposition of  $\beta'$  into  $B'[\vec{w}', (\alpha'_i)_i, (\text{dec}'_j)_j]$ . By definition of  $\underline{\text{l-frame}}_l^P$ , and using the fact that  $\text{fresh}(\mathcal{R}_l^P; \vec{w})$ , we have:

$$\underline{\text{l-frame}}_l^P(\beta) \equiv B[\vec{w}, (\{\{\alpha_i\}_{pk_i}^{n_i}\}_i, \text{dec}(\text{l-frame}_l^P(s_j), sk_j))]$$

Similarly:

$$\underline{\text{l-frame}}_l^P(\beta') \equiv B'[\vec{w}', (\{\{\alpha'_i\}_{pk_i}^{n'_i}\}_i, \text{dec}(\text{l-frame}_l^P(s'_j), sk'_j))]$$

We have three cases:

- Either  $\beta \equiv \{m\}_{pk}^n \in \mathcal{E}_l^P$ . Then  $\underline{\text{l-frame}}_l^P(\beta) \equiv \{\{\beta, 0\}_{pk}^n\}$ . By definition of  $\underline{\text{l-frame}}$ , and using the fact that  $\underline{\text{l-frame}}_l^P(\beta)\theta \equiv \underline{\text{l-frame}}_l^P(\beta')\theta'$ , we get that  $\beta'$  is of the form  $\{m'\}_{pk}^n$ . We deduce from the freshness side condition of  $n$  that  $m' \equiv m$ .

- Or  $\beta \equiv \text{dec}$  where  $\text{dec}$  is a  $\mathcal{S}_l^P$ -decryption oracle call guarding  $\text{dec}(s, \text{sk})$ . Then  $\underline{\text{l-frame}}_l^P(\beta) \equiv \text{dec}(\underline{\text{l-frame}}_l^P(s), \text{sk})\mu$ , where  $\mu$  is the substitution that lifts positions of  $s$  into positions of  $\text{dec}(s, \text{sk})$ , i.e. for every  $\alpha \in \mathcal{E}_l^P$  and position  $p \in \text{pos}(s)$ :

$$\mu(\llbracket \alpha, p \rrbracket) \equiv \llbracket \alpha, 0 \cdot p \rrbracket$$

By definition of  $\underline{\text{l-frame}}$ , and using the fact that  $\underline{\text{l-frame}}_l^P(\beta)\theta \equiv \underline{\text{l-frame}}_l^P(\beta')\theta'$  and that  $\beta'$  is a  $\mathcal{S}_l^P$ -normalized basic term, we get that  $\beta'$  is also some  $\text{dec}'$  where  $\text{dec}'$  is a  $\mathcal{S}_l^P$ -decryption oracle call guarding  $\text{dec}(s', \text{sk})$ .

Moreover we have  $\underline{\text{l-frame}}_l^P(s)\mu\theta \equiv \underline{\text{l-frame}}_l^P(s')\mu\theta$ , and  $s, s'$  are  $\mathcal{S}_l^P$ -normalized basic terms. Hence by induction hypothesis  $\underline{\text{l-frame}}_l^P(s) \equiv \underline{\text{l-frame}}_l^P(s')$ , which concludes this case.

- Or we are not in one of the two cases above. Then, there exists  $f \in \mathcal{F}_{\text{if},0}$  s.t.  $\beta \equiv f(u_1, \dots, u_n)$  and  $\beta' \equiv f(u'_1, \dots, u'_n)$ , where  $u_1, \dots, u_n$  and  $u'_1, \dots, u'_n$  are  $\mathcal{S}_l^P$ -normalized basic term. Hence  $\underline{\text{l-frame}}_l^P(\beta)$  and  $\underline{\text{l-frame}}_l^P(\beta')$  both starts with the function symbol  $f$ .

Moreover, if we let, for every  $1 \leq i \leq n$ ,  $\mu_i$  be the lifting substitution such that, for every  $\alpha \in \mathcal{E}_l^P$  and position  $p$ ,  $\mu_i(\llbracket \alpha, p \rrbracket) \equiv \llbracket \alpha, i \cdot p \rrbracket$ , then:

$$\underline{\text{l-frame}}_l^P(\beta) \equiv f(\underline{\text{l-frame}}_l^P(u_1)\mu_1, \dots, \underline{\text{l-frame}}_l^P(u_n)\mu_n)$$

$$\underline{\text{l-frame}}_l^P(\beta') \equiv f(\underline{\text{l-frame}}_l^P(u'_1)\mu_1, \dots, \underline{\text{l-frame}}_l^P(u'_n)\mu_n)$$

We apply  $\theta$  to the equations above, and use the fact that  $\underline{\text{l-frame}}_l^P(\beta)\theta \equiv \underline{\text{l-frame}}_l^P(\beta')\theta'$ :

$$\begin{aligned} f(\underline{\text{l-frame}}_l^P(u_1)\mu_1\theta, \dots, \underline{\text{l-frame}}_l^P(u_n)\mu_n\theta) &\equiv \underline{\text{l-frame}}_l^P(\beta)\theta \\ &\equiv \underline{\text{l-frame}}_l^P(\beta')\theta' \\ &\equiv f(\underline{\text{l-frame}}_l^P(u'_1)\mu_1\theta, \dots, \underline{\text{l-frame}}_l^P(u'_n)\mu_n\theta) \end{aligned}$$

Hence, for every  $1 \leq i \leq n$ ,  $\underline{\text{l-frame}}_l^P(u_i)\mu_i\theta \equiv \underline{\text{l-frame}}_l^P(u'_i)\mu_i\theta$ . By induction hypothesis, we deduce that  $\underline{\text{l-frame}}_l^P(u_i) \equiv \underline{\text{l-frame}}_l^P(u'_i)$ . Therefore  $\underline{\text{l-frame}}_l^P(\beta) \equiv \underline{\text{l-frame}}_l^P(\beta')$ .  $\square$

*Definition 70.* We let  $<_{\text{st}}$  be the strict, well-founded, subterm ordering.

*Nested Sequences of Basic Conditions.* We want to bound the number of nested basic condition appearing in  $P \vdash_{\alpha}^{\text{npf}} t \sim t'$ . Using the contrapositive of Proposition 28, we know that when  $\beta <_{\text{st}} \beta'$  we have  $\underline{\text{l-frame}}_l^P(\beta) \not\equiv \underline{\text{l-frame}}_l^P(\beta')$ . Moreover, using Proposition 31 and Proposition 32, we know that  $\underline{\text{l-frame}}_l^P(\beta) \not\equiv \underline{\text{l-frame}}_l^P(\beta')$  implies that  $\underline{\text{l-frame}}_l^P(\beta)\theta \not\equiv \underline{\text{l-frame}}_l^P(\beta')\theta'$  (for every substitutions  $\theta, \theta'$ ).

Therefore, for any sequence of nested  $\mathcal{S}_l^P$ -normalized basic conditions:

$$\beta_1 <_{\text{st}} \dots <_{\text{st}} \beta_n$$

and for any substitutions  $\theta_1, \dots, \theta_n$ , we know that  $(\underline{\text{l-frame}}_l^P(\beta_i)\theta_i)_{1 \leq i \leq n}$  is a sequence of pair-wise distinct terms. To use this, we prove that there exists a sequence of substitutions  $\theta_1, \dots, \theta_n$  such that:

$$\{\underline{\text{l-frame}}_l^P(\beta_1)\theta_1, \dots, \underline{\text{l-frame}}_l^P(\beta_n)\theta_n\} \subseteq \mathcal{B}(t, t')$$

where  $\mathcal{B}(t, t')$  is a set of bounded size w.r.t.  $|t| + |t'|$ . Since the  $(\underline{\text{l-frame}}_l^P(\beta_i)\theta_i)_{1 \leq i \leq n}$  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  $\mathbf{0}(\text{dec}(w, \text{sk}))$  by  $\text{dec}(w, \text{sk})$  (where  $\text{sk} \in \mathcal{K}$ ), non-deterministically stopping at some encryptions. Formally:

$$\zeta_{\mathcal{K}}(u) = \begin{cases} \{\text{dec}(v, \text{sk}) \mid w \in v \in \zeta_{\mathcal{K}}(w)\} & \text{if } u \equiv \mathbf{0}(\text{dec}(w, \text{sk})) \text{ and } \text{sk} \in \mathcal{K} \\ \{u\} \cup \{\{v\}_{\text{pk}(n)}^{nr} \mid v \in \zeta_{\mathcal{K}}(m)\} & \text{if } u \equiv \{m\}_{\text{pk}(n)}^{nr} \text{ and } \text{sk}(n) \in \mathcal{K} \\ \{f(v_1, \dots, v_n) \mid \forall i, v_i \in \zeta_{\mathcal{K}}(u_i)\} & \text{otherwise, where } u \equiv f(u_1, \dots, u_n) \end{cases}$$

Moreover, given a set of ground terms  $\mathcal{S}$ , we let  $\text{guards}_{\mathcal{K}}(\mathcal{S})$  be an over-approximation of the set of guards of terms in  $\mathcal{S}$ :

$$\text{guards}_{\mathcal{K}}(\mathcal{S}) = \{\text{eq}(s, \alpha) \mid \text{dec}(s, \text{sk}(n)) \in \mathcal{S} \wedge \alpha \equiv \{\_ \}_- \in \text{st}(s) \wedge \text{sk}(n) \in \mathcal{K}\}$$

*Definition 72.* Let  $\mathcal{S}_k(t)$  be the set of private keys appearing in  $t \downarrow_R$ , i.e.  $\mathcal{S}_k(t) = \{\text{sk}(n) \mid \text{sk}(n) \in \text{st}(t \downarrow_R)\}$ . For every term  $t$ , we let  $\mathcal{B}(t)$  be the set:

$$\mathcal{B}(t) = \bigcup_{\mathcal{K} \subseteq \mathcal{S}_k(t)} \bigcup_{\substack{u \in \text{st}(\text{leave-st}(t \downarrow_R)) \\ \forall u \in \text{st}(\text{cond-st}(t \downarrow_R))}} \zeta_{\mathcal{K}}(u) \cup \text{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u))$$

Moreover, we let  $\mathcal{B}(t, t') = \mathcal{B}(t) \cup \mathcal{B}(t')$ .

**PROPOSITION 33.** *Let  $P \vdash_{\alpha}^{nrf} t \sim t'$  and  $l \in \text{label}(P)$ . Let  $\beta$  be a  $\mathcal{S}_l^P$ -normalized basic condition. Then, for every  $u \in \text{leave-st}(\beta \downarrow_R)$ , there exists  $\theta$  such that  $\underline{\text{l-frame}}_l^P(\beta)\theta \in \zeta_{\mathcal{K}}(u)$ .*

**PROOF.** We show this by induction on  $|\beta|$ .

- If  $\beta$  is an encryption  $\{m\}_{\text{pk}}^n \in \mathcal{E}_l^P$ , then  $\underline{\text{l-frame}}_l^P(\beta) \equiv \{\{\_ \}_{\beta, 0}\}_{\text{pk}}^n$  and:

$$\text{leave-st}(\beta \downarrow_R) = \{\{v\}_{\text{pk}}^n \mid v \in \text{leave-st}(m \downarrow_R)\}$$

Let  $u \in \text{leave-st}(\beta \downarrow_R)$ , there exists  $u_m \in \text{leave-st}(m \downarrow_R)$  such that  $u \equiv \{u_m\}_{\text{pk}}^n$ . Let  $\theta$  be the substitution mapping  $\{\_ \}_{\beta, 0}$  to  $u_m$ . Then:

$$\underline{\text{l-frame}}_l^P(\beta)\theta \equiv \{u_m\}_{\text{pk}}^n \equiv u \in \zeta_{\mathcal{K}_l^P}(u)$$

- If  $\beta$  is a decryption oracle call in  $\mathcal{D}_l^P$  for  $\text{dec}(s, \text{sk})$ , the:

$$\text{leave-st}(\beta \downarrow_R) \subseteq \{\text{dec}(u_s, \text{sk}) \mid u_s \in \text{leave-st}(s \downarrow_R)\} \cup \{\mathbf{0}(\text{dec}(u_s, \text{sk})) \mid u_s \in \text{leave-st}(s \downarrow_R)\}$$

Let  $u \in \text{leave-st}(\beta \downarrow_R)$ , there exists  $u_s \in \text{leave-st}(s \downarrow_R)$  such that  $u \equiv \text{dec}(u_s, \text{sk})$  or  $u \equiv \mathbf{0}(\text{dec}(u_s, \text{sk}))$ . Since  $s$  is a  $\mathcal{S}_l^P$ -normalized basic term, by induction hypothesis we have  $\theta$  such that  $\underline{\text{l-frame}}_l^P(s)\theta \in \zeta_{\mathcal{K}_l^P}(u_s)$ . Moreover:

$$\underline{\text{l-frame}}_l^P(\beta) \equiv \text{dec}(\underline{\text{l-frame}}_l^P(s)\mu, \text{sk})$$

where  $\mu$  is a renaming of hole variables. Let  $\theta' = \mu^{-1}\theta$ , then:

$$\underline{\text{l-frame}}_l^P(\beta)\theta' \equiv \text{dec}(\underline{\text{l-frame}}_l^P(s)\mu\mu^{-1}\theta, \text{sk}) \equiv \text{dec}(\underline{\text{l-frame}}_l^P(s)\theta, \text{sk}) \in \zeta_{\mathcal{K}_l^P}(u)$$

- Otherwise,  $\beta \equiv f(\beta_1, \dots, \beta_n)$  where, for every  $1 \leq i \leq n$ ,  $\beta_i$  is a  $\mathcal{S}_l^P$ -normalized basic term. Then, using the fact that  $\beta$  is a  $\mathcal{S}_l^P$ -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  $1 \leq i \leq n$   $v_i \in \text{leave-st}(\beta_i \downarrow_R)$  and  $u \equiv f(v_1, \dots, v_n)$ . By induction hypothesis, there exists  $\theta_1, \dots, \theta_n$  such that for every  $1 \leq i \leq n$ :

$$\underline{\text{l-frame}}_l^P(\beta_i)\theta_i \in \zeta_{\mathcal{K}_l^P}(v_i)$$

For very  $1 \leq i \leq n$ , let  $\mu_i$  be the lifting substitution such that, for every  $\alpha \in \mathcal{E}_l^P$  and position  $p$ ,  $\mu_i(\llbracket \alpha, p \rrbracket) \equiv \llbracket \alpha, i \cdot p \rrbracket$ . 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  $(\mu_i\theta_i)_{1 \leq i \leq n}$  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:

$$f(\underline{\text{l-frame}}_l^P(\beta_1)\mu_1\theta_1, \dots, \underline{\text{l-frame}}_l^P(\beta_n)\mu_n\theta_n) \in \zeta_{\mathcal{K}_l^P}(u) \quad \square$$

We lift the previous result to  $\alpha$ -bounded conditions.

**LEMMA 23.** *Let  $P \vdash_{\alpha}^{npf} t \sim t'$ ,  $l$  a branch label in  $\text{label}(P)$ ,  $h$  a proof index and  $\beta \in (\leq_{bt}^{h,l}(t, P) \cup \text{cs-path}^{h,l}(t, P))$ . If  $\beta$  is  $(t, P)$ - $\alpha$ -bounded then there exists a substitution  $\theta$  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  $(t, P)$ - $\alpha$ -bounded terms.

- **Base case:** Assume  $h = \epsilon$  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  $\theta$  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  $h = \epsilon$  and there exists  $\beta'$  such that:

$$(\beta, \beta') \in (\leq_{\sim}^{\epsilon, l} \cup \leq_{c \sim c}^{\epsilon, l} \cup \leq_{\text{cs} \sim \text{cs}}^{\epsilon, l})(t \sim t', P) \quad \text{and} \quad \text{leave-st}(\beta' \downarrow_R) \cap \text{st}(t' \downarrow_R) \neq \emptyset$$

By Proposition 30 we know that  $\underline{\text{l-frame}}_l^P(\beta) \equiv \underline{\text{r-frame}}_l^P(\beta')$ . 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  $\theta$  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  $\beta \in \text{cs-path}^{h,l}(t, P)$  and that there exists  $\epsilon \leq_{bt}^{h,l}(t, P)$  such that  $\epsilon$  is  $(t, P)$ - $\alpha$ -bounded and  $\beta \in \text{leave-st}(\epsilon \downarrow_R)$ . By induction hypothesis we have  $\theta$  such that  $\underline{\text{l-frame}}_l^P(\epsilon)\theta \in \mathcal{B}(t, t')$ . We know that  $\beta$  is if-free and in  $R$ -normal form and that  $\epsilon$  is a  $S_l^P$ -normalised basic term. Therefore, by Proposition 27, we have  $\underline{\text{l-frame}}_l^P(\beta) \equiv \underline{\text{l-frame}}_l^P(\epsilon)$ . 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  $\epsilon \leq_{bt}^{h,l}(t, P)$  such that:
  - $\epsilon \equiv B[\vec{w}, (\alpha_i)_i, (\text{dec}_j)_j]$  is  $(t, P)$ - $\alpha$ -bounded.
  - $\beta$  is a guard of a  $S_l^P$ -decryption oracle call  $d \in (\text{dec}_j)_j$ .

By induction hypothesis there exists  $\theta$  such that  $\underline{\text{l-frame}}_l^P(\epsilon)\theta \in \mathcal{B}(t, t')$ . Moreover let  $(\text{pk}_i)_i$  and  $(n_i)_i$  be such that  $\forall i, \alpha_i \equiv \{\_ \}_{\text{pk}_i}^{n_i}$ . Then:

$$\underline{\text{l-frame}}_l^P(\epsilon) \equiv B \left[ \vec{w}, \left( \{ \llbracket \alpha_i \rrbracket_{\text{pk}_i}^{n_i} \}_i, (\underline{\text{l-frame}}_l^P(\text{dec}_j)_j) \right) \right]$$

Therefore there exists a renaming of hole variables  $\mu$  such that  $\underline{\text{l-frame}}_l^P(d)\mu\theta \in \text{st}(\underline{\text{l-frame}}_l^P(\varepsilon)\theta)$ . Since  $\mathcal{B}(t, t')$  is closed under  $\text{st}$ , this implies that:

$$\underline{\text{l-frame}}_l^P(d)\mu\theta \in \mathcal{B}(t, t')$$

$d$  is of the form  $\text{dec}(s, \text{sk})$  where  $\text{sk} \in \mathcal{K}$ . Since members of  $\text{guards}_{\mathcal{K}}(\_)$  are of the form  $\text{eq}(\_, \_)$ , we know that there exists some  $u \in \text{st}(\text{leave-st}(t \downarrow_R) \cup \text{cond-st}(t \downarrow_R))$  such that  $\underline{\text{l-frame}}_l^P(d)\mu\theta \in \zeta_{\mathcal{K}}(u)$ . Since  $\beta$  is a guard of  $d$ ,  $\beta$  is of the form  $\text{eq}(s, \alpha)$  where  $\alpha$  is an encryption under key  $\text{pk}$  (corresponding to  $\text{sk}$ ) and randomness  $n$  appearing directly in  $s$ . It follows that:

$$\underline{\text{l-frame}}_l^P(d) \equiv \text{dec}(\underline{\text{l-frame}}_l^P(s), \text{sk}) \quad \underline{\text{l-frame}}_l^P(\beta) \equiv \text{eq}(\underline{\text{l-frame}}_l^P(s), \{[\alpha]_{\text{pk}}^n\})$$

Since  $\alpha$  appears directly in  $s$ , and since  $\underline{\text{l-frame}}_l^P(d)\mu\theta \in \zeta_{\mathcal{K}}(u)$ , there exists  $\theta'$  such that:

$$\underline{\text{l-frame}}_l^P(\beta)\theta' \in \text{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u)) \subseteq \mathcal{B}(t, t') \quad \square$$

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:*

$$|\mathcal{B}(t)| \leq |t \downarrow_R|^2 \cdot 2^{|t \downarrow_R|}$$

**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  $u \in \text{st}(\text{leave-st}(t \downarrow_R) \cup \text{cond-st}(t \downarrow_R))$ , we have  $u \in \text{st}(t \downarrow_R)$ , and therefore  $|u| \leq |t \downarrow_R|$ . Therefore the set  $\zeta_{\mathcal{K}}(u)$  contains terms of size at most  $|t \downarrow_R|$ .

Let  $\text{dec}(s, \text{sk}) \in \zeta_{\mathcal{K}}(u)$ , then  $|\text{dec}(s, \text{sk})| = |s| + 3$  and for every  $\alpha$  appearing in  $s$ :

$$|\text{eq}(s, \alpha)| = |s| + |\alpha| + 1 \leq 2|s| + 1 \leq 2|\text{dec}(s, \text{sk})| \leq 2|t \downarrow_R|$$

Hence the set  $\text{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u))$  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  $\text{dec}(s, \text{sk})$  where an encryption might be, there are at most  $|s| - 1 \leq |t \downarrow_R| - 1$  such  $\alpha$ , independently of the set of keys  $\mathcal{K}$ . It follows that:

$$\left| \bigcup_{\mathcal{K} \subseteq \mathcal{S}_{\mathcal{K}}(t)} \text{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u)) \right| \leq |\zeta_{\mathcal{K}}(u)| \cdot (|t \downarrow_R| - 1)$$

Independently of the set of keys  $\mathcal{K}$  chosen, we have at most  $|\text{st}(t \downarrow_R)| \leq |t \downarrow_R|$  choices for  $u$ , and the set  $\bigcup_{\mathcal{K} \subseteq \mathcal{S}_{\mathcal{K}}(t)} \zeta_{\mathcal{K}}(u)$  contains at most  $2^{|u|} \leq 2^{|t \downarrow_R|}$  elements (we choose the positions where we remove  $\mathbf{0}$ s). Hence:

$$\begin{aligned} \left| \bigcup_{\mathcal{K} \subseteq \mathcal{S}_{\mathcal{K}}(t)} \zeta_{\mathcal{K}}(u) \cup \text{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u)) \right| &\leq \left| \bigcup_{\mathcal{K} \subseteq \mathcal{S}_{\mathcal{K}}(t)} \zeta_{\mathcal{K}}(u) \right| + \left| \bigcup_{\mathcal{K} \subseteq \mathcal{S}_{\mathcal{K}}(t)} \text{guards}_{\mathcal{K}}(\zeta_{\mathcal{K}}(u)) \right| \\ &\leq |\zeta_{\mathcal{K}}(u)| + (|t \downarrow_R| - 1) \cdot |\zeta_{\mathcal{K}}(u)| \leq |t \downarrow_R| \cdot 2^{|t \downarrow_R|} \end{aligned}$$

By consequence:

$$|\mathcal{B}(t)| \leq |t \downarrow_R| \cdot |t \downarrow_R| \cdot 2^{|t \downarrow_R|} \leq |t \downarrow_R|^2 \cdot 2^{|t \downarrow_R|} \quad \square$$

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  $\text{label}(P)$ ,  $h$  a proof index. Let  $(\beta_i)_{i \leq n}$  such that for all  $i$ ,  $\beta_i \leq_{bt}^{h,l}(t, P)$ . If  $\beta_1 <_{st} \dots <_{st} \beta_n$  then  $n \leq |\mathcal{B}(t, t')|$ .*

PROOF. For every  $i \neq j$ , we know, using Proposition 28, that  $\text{l-frame}_i^P(\beta_i) \not\equiv \text{l-frame}_i^P(\beta_j)$ . By Proposition 31, we deduce that  $\text{l-frame}_i^P(\beta_i) \not\equiv \text{l-frame}_i^P(\beta_j)$ . Since  $P \vdash_{\alpha}^{\text{npf}} t \sim t'$ , we know that for every  $i$ ,  $\beta_i$  is  $(t, P)$ - $\alpha$ -bounded. Using Lemma 23, we deduce that for every  $i$ , there exists a substitution  $\theta_i$  such that:

$$\text{l-frame}_i^P(\beta_i)\theta_i \in \mathcal{B}(t, t')$$

Using the contrapositive of Proposition 32, we have that for every  $i \neq j$ :

$$\text{l-frame}_i^P(\beta_i)\theta_i \not\equiv \text{l-frame}_i^P(\beta_j)\theta_j$$

Therefore, by a pigeon-hole argument,  $n \leq |\mathcal{B}(t, t')|$ .  $\square$

### H.3 Candidate Sequences

Let  $P \vdash_{\alpha}^{\text{npf}} t \sim t'$ . 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  $\alpha$ -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  $p \in \text{pos}(C)$ , if  $C|_p \equiv []$  then  $p = p' \cdot 0$  and there exist two nonces  $n, n_r$  such that  $C|_{p'} \equiv \{[]\}_{\text{pk}(n)}^{n_r}$ .

Moreover, we require that every hole appears at most once.

*Remark 12.* For every  $\beta \leq_{\text{bt}}^{h,l}(t, P)$ , the context  $\text{l-frame}_i^P(\beta)$  is an  $\alpha$ -context.  $\diamond$

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  $\vdash_{\alpha}^{\text{npf}} t \sim t'$ , 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  $\alpha \in \mathcal{A}_{n+1} \setminus \mathcal{A}_n$ ,  $\alpha \equiv \{D[\vec{b} \diamond \vec{u}]\}_{\text{pk}(n_p)}^{n_r}$  where:
  - $\vec{b} \cup \vec{u}$  are in  $\mathcal{U}_{n-1}$  and there exists  $\{\_ \}_-^{n_r} \in \text{st}(t \downarrow_R) \cup \text{st}(t' \downarrow_R)$ .
  - 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  $\{\_ \}_-^{n_r}$ .
- 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  $\alpha$ -context  $C$  such that there exists  $\theta$  with  $C\theta \in \mathcal{B}(t, t')$ .
  - Let  $[]_1, \dots, []_a$  be the variables of  $C$ , and let  $\alpha_1, \dots, \alpha_a$  be encryptions in  $\mathcal{A}_n$ . For all  $1 \leq k \leq a$ , let  $s_i$  be such that  $\{s_i\}_- \equiv \alpha_i \in \mathcal{A}_n$ .
  - Let  $v_0 \equiv C[(s_i)_{1 \leq i \leq a}]$ . Then let  $v$  be the term obtained from  $v_0$  as follows: take positions  $p_1, \dots, p_o \in \text{pos}(C)$  such that for all  $1 \leq i \leq o$ ,  $C|_{p_i} \equiv \text{dec}(\_, \text{sk}_i)$  (where  $\text{sk}_i$  is a valid private key, i.e. of the form  $\text{sk}(n_i)$ ); for every  $1 \leq i \leq o$ , replace in  $v_0$  the subterm  $\text{dec}(s, \text{sk})$  at position  $p$  by  $D[\vec{g} \diamond \vec{w}]$ , where  $\vec{g}$  are terms in  $\mathcal{U}_n$  of the form  $\text{eq}(s, \alpha)$  (with  $\alpha \equiv \{\_ \}_-^{n_\alpha} \in \mathcal{A}_n$  and  $\alpha$  directly appears in  $s$ ) and  $\forall w \in \vec{w}$ ,  $w \equiv \text{dec}(s, \text{sk})$  or  $w \equiv \mathbf{0}(\text{dec}(s, \text{sk}))$ .

PROPOSITION 35. Let  $P \vdash_{\alpha}^{npf} t \sim t'$ . For  $l \in \text{label}(P)$ , there exists a valid candidate sequence  $(\mathcal{U}_n, \mathcal{A}_n)_{n \in \mathbb{N}}$  for  $t, t'$  such that:

$$\bigcup_h \leq_{bt}^{h,l}(t, P) \subseteq \bigcup_{n < |\mathcal{B}(t, t')|} \mathcal{U}_n \quad \text{and} \quad \bigcup_h \text{cs-path}^{h,l}(t, P) \subseteq \bigcup_{n < |\mathcal{B}(t, t')|} \text{leave-st}(\mathcal{U}_n \downarrow_R)$$

PROOF. First, we show that there exists a valid candidate sequence such that the inclusion holds when taking the union over  $\mathbb{N}$  on the right, and s.t. for every  $n$ ,  $\mathcal{A}_n$  contains only valid encryptions in  $\mathcal{E}_l^P$ , i.e.:

$$\mathcal{S} = \bigcup_h \leq_{bt}^{h,l}(t, P) \subseteq \bigcup_{n < +\infty} \mathcal{U}_n \quad \text{and} \quad \bigcup_{n \in \mathbb{N}} \mathcal{A}_n \subseteq \mathcal{E}_l^P \quad (25)$$

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  $\mathcal{A}_{n+1} \setminus \mathcal{A}_n$  then there exists  $v \in \vec{b} \cup \vec{u}$  such that  $v \in \mathcal{U}_n \setminus \mathcal{U}_{n-1}$  (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  $\beta$  in  $\mathcal{S} \setminus \mathcal{S}'$  such that  $\beta$  is minimal for  $<_{st}$  in  $\mathcal{S} \setminus \mathcal{S}'$ . Then we add  $\beta$  to  $\mathcal{S}'$  and update  $(\mathcal{A}_n)_{n \in \mathbb{N}}$  as follows:

*Case 1.* If  $\beta$  is minimal for  $<_{st}$  in  $\mathcal{S}$ , we have  $\beta$  of the form  $B[\vec{w}, (\alpha_i)_{i \in I}, (\text{dec}_j)_{j \in J}]$ . By minimality of  $\beta$ , we have  $I = \emptyset$  and for all  $j \in J$ ,  $\text{dec}_j$  has no encryptions in  $\mathcal{E}_l^P$ , and by consequence no guards. It follows that  $\beta$  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  $\beta$  such that for all  $\beta' <_{st} \beta$ ,  $\beta' \in \mathcal{S}'$ . Since  $\mathcal{S}' \subseteq \bigcup_{n \in \mathbb{N}} \mathcal{U}_n$ , and since  $\{\beta' \mid \beta' <_{st} \beta\}$  is finite, there exists  $n_m$  such that:

$$\{\beta' \mid \beta' <_{st} \beta\} \cap \left( \leq_{bt}^{h,l}(t, P) \cup \text{cs-path}^{h,l}(t, P) \right) \subseteq \bigcup_{0 \leq n \leq n_m} \mathcal{U}_n$$

From Lemma 23 we have a substitution  $\theta$  such that:

$$\text{l-frame}_l^P(\beta)\theta \in \mathcal{B}(t, t')$$

We then just need to show that we can obtain  $\beta$  from  $\text{l-frame}_l^P(\beta)$  using the procedure defining  $\mathcal{U}_{n_m+1}$ :

- For all encryption  $\alpha \equiv \{m\}_{pk}^n \in \text{st}(\beta) \cap \mathcal{E}_l^P$ , we know that  $m \equiv C[\vec{b} \diamond \vec{u}]$  where  $\vec{b}, \vec{u} <_{st} \beta$ . Hence  $\vec{b}, \vec{u}$  are in  $\bigcup_{0 \leq n \leq n_m} \mathcal{U}_n$ . We then have two cases:
  - either  $\bigcup_{n \in \mathbb{N}} \mathcal{A}_n$  already contains an encryption  $\alpha'$  with randomness  $n$ . Since  $\bigcup_{n \in \mathbb{N}} \mathcal{A}_n \subseteq \mathcal{E}_l^P$ , and using the side-condition of the  $\text{cca}_2$  application, we know that  $\alpha \equiv \alpha' \in \bigcup_{n \in \mathbb{N}} \mathcal{A}_n$ . By minimality of the  $(\mathcal{A}_n)_{n \in \mathbb{N}}$  we know that  $\alpha \in \mathcal{A}_{n_m+1}$ .
  - or  $\bigcup_{n \in \mathbb{N}} \mathcal{A}_n$  does not contain an encryption with randomness  $n$ . Then we simply add  $\alpha$  to  $\mathcal{A}_{n'}$ , where  $n' \leq n_m + 1$  is the smallest possible: we know that there exists such a  $n'$  since adding  $\alpha$  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  $\vdash_{\alpha}^{\text{npf}}$ .

Then we replace in  $\text{l-frame}_l^P(\beta)$  the holes  $[\ ]_{\alpha, -}$  by  $\{C[\vec{b} \diamond \vec{u}]\}_{\text{pk}}^n$ . This produce a term  $v_0$ .

- Finally we also replace in  $v_0$  every occurrence of  $\text{dec}(\_, \text{sk})$  or  $\mathbf{0}(\text{dec}(\_, \text{sk}))$  in  $\text{st}(\text{l-frame}_l^P(\beta))$  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  $\vec{g} <_{\text{st}} \beta$ , hence are in  $\bigcup_{0 \leq n \leq n_m} \mathcal{U}_n$ .

*Conclusion.* We show that when  $\mathcal{S} = \mathcal{S}'$  we have:

$$\mathcal{S} \cap \bigcup_{n < +\infty} \mathcal{U}_n = \mathcal{S} \cap \bigcup_{n < |\mathcal{B}(t, t')|} \mathcal{U}_n \quad (26)$$

Assume that  $\mathcal{S} \cap \mathcal{U}_{|\mathcal{B}(t, t')|-1} \subsetneq \mathcal{S} \cap \mathcal{U}_{|\mathcal{B}(t, t')|}$ , take  $\beta \in \mathcal{S} \cap (\mathcal{U}_{|\mathcal{B}(t, t')|} \setminus \mathcal{U}_{|\mathcal{B}(t, t')|-1})$ . We know that  $\beta \equiv B[\vec{w}, (\alpha_i)_i, (\text{dec}_j)_j]$  and that there is an encryption  $\alpha$  in  $(\alpha_i)_i$  or in the encryptions of the  $(\text{dec}_j)_j$  such that  $\alpha \in \mathcal{A}_{|\mathcal{B}(t, t')|-1} \setminus \mathcal{A}_{|\mathcal{B}(t, t')|-2}$  (otherwise  $\beta$  would be in  $\mathcal{S} \cap \mathcal{U}_{|\mathcal{B}(t, t')|-1}$ ). Let  $\alpha \equiv \{C[\vec{b} \diamond \vec{u}]\}_{\text{pk}}^n$ , 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  $v \in \mathcal{U}_{|\mathcal{B}(t, t')|-1} \setminus \mathcal{U}_{|\mathcal{B}(t, t')|-2}$ . Since  $\beta$  is in  $\mathcal{S}$  and since  $v$  is a  $\mathcal{S}_l^P$ -normalized basic term appearing in  $\beta$  we know that  $v \in \mathcal{S}$ . Let  $\beta_0 \equiv \beta$ ,  $\beta_1 \equiv v$ , we have  $v \in \mathcal{S} \cap (\mathcal{U}_{|\mathcal{B}(t, t')|-1} \setminus \mathcal{U}_{|\mathcal{B}(t, t')|-2})$ . By induction we can build a sequence of terms  $\beta_n$ , for  $n \in \{0, \dots, |\mathcal{B}(t, t')|\}$  such that for all  $0 \leq n \leq |\mathcal{B}(t, t')|$ ,  $\beta_n \in \mathcal{S} \cap (\mathcal{U}_{|\mathcal{B}(t, t')|-i} \setminus \mathcal{U}_{|\mathcal{B}(t, t')|-(i+1)})$  and  $\beta_{n+1} <_{\text{st}} \beta_n$  (with the convention  $\mathcal{U}_{-1} = \emptyset$ ). We built a sequence of terms in  $\mathcal{S}$ , strictly ordered by  $<_{\text{st}}$  and of length  $|\mathcal{B}(t, t')| + 1$ . This contradicts Lemma 24. Absurd.

To finish, it remains to show that:

$$\bigcup_h \text{cs-path}^{h,l}(t, P) \subseteq \bigcup_{n < |\mathcal{B}(t, t')|} \text{leave-st}(\mathcal{U}_n \downarrow_R)$$

Let  $b$  in  $\bigcup_h \text{cs-path}^{h,l}(t, P)$ . Using Proposition 29 we know that there exists  $\gamma \leq_{\text{bt}}^{h,l}(t, P)$  such that  $b \in \text{leave-st}(\gamma \downarrow_R)$ . Since  $\gamma \in \bigcup_{n < |\mathcal{B}(t, t')|} \mathcal{U}_n \downarrow_R$ , we have  $b \in \bigcup_{n < |\mathcal{B}(t, t')|} \text{leave-st}(\mathcal{U}_n \downarrow_R)$ .  $\square$

**PROPOSITION 36.** *For all terms  $u$ , let  $C_u$  be the set of  $\alpha$ -contexts:*

$$C_u = \{C \mid \exists \theta. C\theta \equiv u \wedge \text{every hole appears at most once}\}$$

and  $C_u^\alpha$  be  $C_u$  quotiented by the  $\alpha$ -renaming of holes relation. Then  $|C_u^\alpha| \leq 2^{|u|}$ .

**PROOF.** The set of contexts  $C_u^\alpha$  can be injected in the subsets of positions of  $u$  as follows: for every context  $C$ , associate to  $C$  the set of positions of  $u$  such that  $C|_p$  is a hole. This is invariant by  $\alpha$ -renaming and uniquely characterizes  $C$  modulo hole renaming. It follows that there are less element of  $C_u^\alpha$  than subsets of  $\text{pos}(u)$ , i.e.  $2^{|\text{pos}(u)|} = 2^{|u|}$ .  $\square$

**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  $n \in \mathbb{N}$ :*

$$|\mathcal{A}_n| \leq N \quad |\mathcal{U}_n| \leq N^2 \cdot 2^{3 \cdot N}$$

**PROOF.** For every  $n$ ,  $\mathcal{A}_n$  contains only terms of the form  $\alpha \equiv \{m\}_{\text{pk}}^{n_r}$ , where  $\{\_ \}_-^{n_r} \in \text{st}(t \downarrow_R) \cup \text{st}(t' \downarrow_R)$ . 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  $\alpha$ -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  $\alpha$ -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:

$$\left| \bigcup_{u \in \mathcal{B}(t, t')} C_u^\alpha \right| \leq \sum_{u \in \mathcal{B}(t, t')} |C_u^\alpha| \leq \sum_{u \in \mathcal{B}(t, t')} 2^{2 \cdot N} \leq N^2 \cdot 2^N \cdot 2^{2 \cdot N} = N^2 \cdot 2^{3 \cdot N} \quad \square$$

**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  $n \in \mathbb{N}$ :*

$$\forall u \in \bigcup_{n < |\mathcal{B}(t, t')|} \mathcal{U}_n, |u| \leq 2^{Q(N)} \cdot 2^{4 \cdot N}$$

Where  $Q(X)$  is a polynomial of degree 4.

**PROOF.** Even though there are at most  $|\mathcal{B}(t, t')| \cdot N^2 \cdot 2^{3 \cdot N}$  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  $\alpha \in \mathcal{A}_{n+1} \setminus \mathcal{A}_n$ ,  $\alpha$  is of the form  $\{C[\vec{b} \diamond \vec{u}]\}_{\text{pk}}^n$ , where  $\vec{b}, \vec{u}$  are in  $\mathcal{U}_n$  and  $C$  is such that no term appears twice on the same branch. Recall that we call branch the ordered list of *inner conditions*, which does not include the final leaf. It 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:

$$|C[\vec{b} \diamond \vec{u}]| \leq |C| + |C| \cdot U_n \leq 2 \cdot |C| \cdot U_n \leq 2^{|\mathcal{U}_n|+3} \cdot U_n$$

since  $U_n$  is greater than 1 (terms can not be of size 0). Therefore  $|\alpha| \leq 4 + 2^{|\mathcal{U}_n|+3} \cdot U_n$ . Using the bound from Proposition 37, we can take:

$$A_n = 4 + 2^{N^2 \cdot 2^{3 \cdot N} + 3} \cdot U_n$$

Now let  $u \equiv C[(\alpha_i)_{i \in I}, (\text{dec}_j)_{j \in J}]$  in  $\mathcal{U}_{n+1} \setminus \mathcal{U}_n$ . 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 leaf is of size at most  $|C[(\alpha_i)_{i \in I}, (\text{dec}_j)_{j \in J}]| + 1 \leq |C| + |I| \cdot A_n + 1$ . Hence every decryption's size is upper-bounded by:

$$N + N \cdot U_n + (N + 1) \cdot (|C| + |I| \cdot A_n + 1)$$

Finally  $|C|$  is such that there exists  $\theta$  such that  $C\theta \in \mathcal{B}(t, t')$ , hence  $|C| \leq 2 \cdot N$  using Proposition 34. Hence, assuming  $U_n \geq N$  (which will be the case):

$$\begin{aligned} |C[(\alpha_i)_{i \in I}, (\text{dec}_j)_{j \in J}]| &\leq |C| + |I| \cdot A_n + |J| \cdot (N + N \cdot U_n + (N + 1) \cdot (|C| + |I| \cdot A_n + 1)) \\ &\leq 2N + 2N \cdot A_n + 2N \cdot (N + N \cdot U_n + (N + 1) \cdot (2N + 2N \cdot A_n + 1)) \end{aligned}$$

Seen as a multi-variate polynomial in  $N, A_n$  and  $U_n$ , we have only monomials  $N, N \cdot A_n, N^2, N^2 \cdot U_n, N^3$  and  $N^3 \cdot A_n$ . Hence there exists a constant  $L$  such that:

$$u \leq L \cdot N^3 (A_n + U_n) \leq L \cdot N^3 (4 + 2^{N^2 \cdot 2^{3 \cdot N} + 3} \cdot U_n + U_n)$$

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.  $\square$

**COROLLARY 3.** *Let  $P \vdash_{\alpha}^{npf} t \sim t'$  and  $N = |\mathcal{B}(t,t')|$ . For  $l \in \text{label}(P)$  and for all proof index  $h$ :*

$$\forall u \in \left( \leq_{bt}^{h,l}(t,P) \cup \text{cs-path}^{h,l}(t,P) \right), |u| \leq 2^{Q(N)}.2^{4.N}$$

**PROOF.** Direct consequence of Proposition 35 and Proposition 38.  $\square$

To conclude, we only need to bound the number of nested  $\text{CS}_{\square}$  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  $1 \leq i < n$ ,  $h_{i+1} \in \text{cs-pos}_P(h_i)$  and  $h_1 = \epsilon$ . Then  $n \leq |\mathcal{B}(t,t')| + 1$ . Moreover  $|\text{label}(P)| \leq 2^{|\mathcal{B}(t,t')|}$ .*

**PROOF.** Let  $l \in \text{label}(P)$  be such that  $h_n \in \text{h-branch}(l)$ . The proof consists in building an increasing sequence of  $\mathcal{S}_l^P$ -normalized basic terms  $\beta_1 <_{\text{st}} \dots <_{\text{st}} \beta_m$  from  $(h_i)_{1 \leq i \leq n}$  of length  $m \geq n$ . We then concludes using Lemma 24.

If  $h_n \neq \epsilon$ , then  $h_n$  is of the form  $h_{x_n}^n$ . We know that  $\text{extract}_{x_n}(h^n, P)$  is a proof of  $b^n \sim b^n$  in  $\mathcal{A}_{\text{CS}_{\square}}$ . Moreover  $b^n \downarrow_R$  is in  $\text{cs-path}^{h_{n-1},l}(t,P)$  and is  $(t,P)$ - $\alpha$ -bounded. Be definition of  $(t,P)$ - $\alpha$ -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  $1 \leq j \leq k_n$ ,  $\beta_{n,j} \leq_{bt}^{h_{n-1},l}(t,P)$ .
- $b^n \downarrow_R \in \text{leave-st}(\beta_{n,1} \downarrow_R)$ .
- $\beta_{n,k_n} \leq_l^{h_{n-1},l}(t,P)$ .
- for all  $1 \leq j < k_n$ ,  $\beta_{n,j}$  is a guard of a decryption in  $\beta_{n,j+1}$ , and therefore  $\beta_{n,j} <_{\text{st}} \beta_{n,j+1}$ .

If  $h_{n-1} \neq \epsilon$ , then since  $\beta_{n,k_n} \leq_l^{h_{n-1},l}(t,P)$  is  $(t,P)$ - $\alpha$ -bounded, and since for any  $\beta \leq_{bt}^{h_{n-1},l}(t,P)$ ,  $\beta_{n,j}$  is not a guard of  $\beta$ , we know that we are in the inductive case with different labels of the definition of  $(t,P)$ - $\alpha$ -bounded terms. Therefore there exists  $b^{n-1} \in \text{cs-path}^{h_{n-2},l}(t,P)$  such that  $b^{n-1} \in \text{leave-st}(\beta_{n,k_n})$ .

We then iterate this process until we reach  $\epsilon$ , 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  $i$ ,  $b^{i-1} \in \text{leave-st}(\beta_{i,k_i} \downarrow_R)$  and  $b^{i-1} \in \text{leave-st}(\beta_{i-1,1} \downarrow_R)$  we know, using Proposition 17, that  $\beta_{i,k_i} \equiv \beta_{i-1,1}$ . Therefore we have:

$$\beta_{n,1} <_{\text{st}} \dots <_{\text{st}} \beta_{n,k_n} \equiv \beta_{n-1,1} <_{\text{st}} \dots <_{\text{st}} \beta_{n-1,k_{n-1}} \dots <_{\text{st}} \beta_{3,k_3} \equiv \beta_{2,1} <_{\text{st}} \dots <_{\text{st}} \beta_{2,k_2}$$

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  $|\text{label}(P)|$ , we only need to observe that we cannot have two  $\text{CS}_{\square}$  applications on the same condition in a given branch. Consider the binary tree associated to the  $\text{CS}_{\square}$  applications in  $P$ , labelled by the corresponding  $\text{CS}_{\square}$  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.  $\square$

**THEOREM (MAIN RESULT).** *The following problem is decidable in 3-NEXPTIME:*

**Input:** A ground formula  $\vec{u} \sim \vec{v}$ .

**Question:** Is  $Ax \wedge \vec{u} \approx \vec{v}$  unsatisfiable?

**PROOF.** Let  $\vec{u} = u_1, \dots, u_n$ ,  $\vec{v} = v_1, \dots, v_n$  and:

$$t \equiv \langle u_1, \langle \dots, \langle u_{n-1}, u_n \rangle \rangle \rangle \quad t' \equiv \langle v_1, \langle \dots, \langle v_{n-1}, v_n \rangle \rangle \rangle$$



Using the  $\text{FA}_{\langle \_, \_ \rangle}$  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  $\pi_1$  and  $\pi_2$  by:

$$\forall n > i \geq 1, \rho_i \equiv \pi_1(\pi_2^{i-1}(\square)) \qquad \rho_n \equiv \pi_2^{n-1}(\square)$$

Then:

$$\frac{\frac{t \sim t'}{(\rho_i[t])_{1 \leq i \leq n} \sim (\rho_i[t'])_{1 \leq i \leq n}} \text{FA}^*}{\vec{u} \sim \vec{v}} R$$

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  $N = |\text{st}(t \downarrow_R)| + |\text{st}(t' \downarrow_R)|$ . Using Proposition 39, we have bounded the number of branches of the proof tree (by  $2^{N^2 \cdot 2^N}$ ), and the number of nested  $\text{CS}_\square$  conditions. For every branch, we non-deterministically guesses a set of  $\alpha$ -bounded basic terms that can appear in a proof  $P$  of  $P \vdash_\alpha^{\text{npf}} t \sim t'$  using the valid candidate sequence algorithm (in polynomial time in  $O(N \cdot 2^{3 \cdot N} \cdot 2^{Q(N)} \cdot 2^{4 \cdot N})$ ), 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  $\text{CCA}_2$  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.  $\square$

#### H.4 The Pure Fragment

**PROOF OF LEMMA 13.** Consider a proof  $P \vdash_\alpha^{\text{npf}} t \sim t'$  that only uses  $\text{Refl}_\alpha$ . For every  $h, l$ , for every  $\gamma \leq_{\text{bt}}^{h,l} (t, P)$ ,  $\gamma$  is a normalized basic term. Since we are in the pure fragment,  $\gamma$  contains no encryptions and decryptions. Consequently,  $\gamma$  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  $\beta \leq_{\text{bt}}^{h,l} \cup_{\text{cs-path}}^{h,l} (t, P)$  such that:

**Inductive case, different labels:**  $\beta \leq_{\text{bt}}^{h,l} (t, P)$ , there exists  $h'$  such that  $h \in \text{cs-pos}(h')$  and  $b \in \text{cs-path}^{h',l}(t, P)$  such that  $b$  is  $(t, P)$ - $\alpha$ -bounded and  $b \in \text{leave-st}(\beta \downarrow_R)$ .

Since  $\beta$  is if-free and in  $R$ -normal form,  $\text{leave-st}(\beta \downarrow_R) = \{\beta\}$ , 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  $\beta$ . Therefore, we can consider a proof of  $t \sim t'$  where for every  $(b, b')$   $\text{cs-path}^{h,l} (t \sim t', P)$ , the proof of  $b \sim b'$  extracted from  $P$  does not use  $\text{CS}_\square$ . Hence  $P$  can be taken in the fragment with no boxed  $\text{CS}$  conditions:

$$\mathfrak{F}(R \cdot \text{CS}^* \cdot \{\text{BFA}(b, b')\}^* \cdot \text{FA}_s^* \cdot \text{Dup}^* \cdot \text{Refl}_\alpha) \quad (27)$$

Remark that if  $u \sim v$  is provable in  $\mathfrak{F}(\{\text{BFA}(b, b')\}^* \cdot \text{FA}_s^* \cdot \text{Dup}^* \cdot \text{Refl}_\alpha)$ , then  $u$  and  $v$  are alpha-equal.

*Conclusion.* We prove by induction on the  $\alpha$ -bounded inductive predicate that for every  $h, l$ , and  $(\beta, \beta') \leq_{\text{bt}}^{h,l} \cup_{\text{cs-path}}^{h,l} (t, P)$ ,  $\beta$  or  $\beta'$  is a member of  $\text{st}(t \downarrow_R) \cup \text{st}(t' \downarrow_R)$ , and  $\beta$  contains only names in  $\text{st}(t) \cup \text{st}(t')$ .

We look at all possible cases in Definition 67. Let  $h, l$ , and  $\beta \leq_{\text{bt}}^{h,l} \cup_{\text{cs-path}}^{h,l} (t, P)$ .

- **Base case:** Since  $\beta$  is if-free and in  $R$ -normal form,  $\text{leave-st}(\beta \downarrow_R) = \{\beta\}$ , hence  $\beta \in \text{st}(t \downarrow_R)$ .

- **Base case:** There exists  $\beta'$  such that:

$$(\beta, \beta') (\leq_{| \sim |}^{\epsilon, l} \cup \leq_{c \sim c}^{\epsilon, l} \cup \text{cs-path}_{\sim}^{\epsilon, l}) (t \sim t', P)$$

and  $\text{leave-st}(\beta' \downarrow_R) \cap \text{st}(t' \downarrow_R) \neq \emptyset$ . We deduce that  $\beta' \in \text{st}(t' \downarrow_R)$ .

- **Inductive case, same label:** if  $\beta \in \text{cs-path}^{h, l}(t, P)$  and there exists  $\epsilon \leq_{\text{bt}}^{h, l}(t, P)$  such that  $\epsilon$  is  $(t, P)$ - $\alpha$ -bounded and  $\beta \in \text{leave-st}(\epsilon \downarrow_R)$ .  
From (27), we now that  $h = \epsilon$ . Since  $\epsilon$  is if-free and in  $R$ -normal form,  $\text{leave-st}(\epsilon \downarrow_R) = \{\epsilon\}$ . Hence  $\beta \equiv \epsilon$ . 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.  $\square$