Deducibility constraints and blind signatures

Sergiu Bursuc[a], Hubert Comon-Lundh[b], Stéphanie Delaune[b]

[a] *Queen's University of Belfast, UK*
[b] *LSV, CNRS & ENS de Cachan & INRIA Saclay, France*

Abstract

Deducibility constraints represent in a symbolic way the infinite set of possible executions of a finite protocol. Solving a deducibility constraint amounts to finding all possible ways of filling the gaps in a proof. For finite *local* inference systems, there is an algorithm that reduces any deducibility constraint to a finite set of solved forms. This allows one to decide any trace security property of cryptographic protocols.

We investigate here the case of infinite local inference systems, through the case study of blind signatures. We show that, in this case again, any deducibility constraint can be reduced to finitely many solved forms (hence we can decide trace security properties). We sketch also another example to which the same method can be applied.

Keywords: formal methods, verification, security protocol, constraint system

1. Introduction

This paper is concerned with the formal verification of security protocols. The formal models of security protocols are (infinite) transition systems, that are infinitely branching, because of the unbounded number of possible fake messages that can be sent by an attacker. In numerous cases, however, only finitely many such messages are relevant for mounting an attack. This is essentially what is proved in [19, 20]: if we assume a fixed number of protocol sessions and the classical public key encryption and pairing primitives, then there is an attack if, and only if, there is an attack in which the messages sent by the attacker are taken out of a fixed set of messages. We refer to this result as *the small attack property*.

More practical algorithms, that do not need to enumerate all possible relevant attacker's behaviour, rely on *deducibility constraints*, introduced in [17]. The idea is to represent symbolically all messages that can be forged by an attacker at a given stage of the protocol execution. An atomic deducibility constraint is an expression $T \stackrel{?}{\vdash} u$ where T is a finite set of terms and u is a term, both of which may contain variables. The deduction relation is interpreted according to the attacker capabilities and the variable instances correspond to the attacker choices of message forging. Deciding the satisfiability of such constraints then allows one to decide whether an attacker, after interacting with the protocol, may get a message that was supposed to remain secret.

These works have two limitations: they are restricted to some basic cryptographic primitives and they only consider the property of being able to get a supposedly secret message. They also consider some authentication properties, through an appropriate encoding. Concerning the first limitation, there are numerous extensions to other cryptographic primitives, for instance exclusive-or [11, 6], modular exponentiation [7, 21, 18], any monoidal theory [13]

and blind signatures [3]. Concerning the second limitation, the idea is to transform the deducibility constraints that have been mentioned above into finitely many solved forms, that represent in a convenient way all possible traces in presence of an active attacker. Then, whether a security property ϕ holds, can be checked by deciding the satisfiability of $\neg\phi$, together with each solved constraint. This is typically what is proposed in [10] (and also more recently in [22]), where it is shown how to decide trace properties, using the solved deducibility constraints, in case of public key and symmetric key encryption and signatures. This raises the problem of systematically designing deducibility constraint solving techniques, that would be applicable for both several primitives and any trace property.

In [4], we show that a locality property of the deduction system is sufficient for designing a deducibility constraint solving algorithm. Locality is a syntactic subformula property of normal proofs [16]: if there is a proof of t, then there is a proof of t whose every intermediate step is either a subterm of t or subterm of some hypothesis. Locality yields a tractable Entscheidungsproblem [16]. As shown in [2], this is also equivalent to a saturation property of the set of inference rules w.r.t. the subterm ordering. Therefore, if a set of inference rules modeling the attacker's capabilities on a given set of primitives can be saturated, yielding a finite set of inference rules, then, according to [4], we can simplify the deducibility constraints into finitely many solved forms and decide any trace security property. It turns out that some relevant proof systems cannot be finitely saturated w.r.t. the subterm ordering. This is the case of blind signatures, as modeled in [15]: saturating the inference rules does not terminate.

Yet, we show in this paper that we can extend the deducibility constraint solving procedures to some infinite local inference systems. Typically, such inference systems are obtained by saturating finite non-local inference systems. We consider the case study of blind signatures and briefly mention another example (homomorphic encryption) to which a very similar procedure works.

The basic idea for solving deducibility constraints is straightforward: given $T \stackrel{?}{\vdash} u$, guess the last inference rule that yields a proof of $u\sigma$ and decompose the constraint accordingly. This hardly terminates in general. In case of a local inference system, we roughly require that the new terms appearing in the constraint are either subterms of u or subterms of T, which, together with a simple strategy, guarantees termination. If the inference system is infinite, there are a priori infinitely many possible last rules that may yield $u\sigma$, which again raises a termination issue. In case of a relatively regular set of inference rules (which is the case for blind signatures and for homomorphic encryption), we may fold infinitely many such last steps in a single one, using an additional abstraction. This is what we show: we consider the case of blind signatures and add a predicate symbol, that allows us to consider all possible last deduction steps at once. We design then a constraint solving procedure, that includes this new predicate symbol, and show that it is terminating and yields solved forms. Trace security properties can be decided using these solved forms. As a witness, we mention the decidability of the first order formulas with equalities together with the deducibility constraints. Finally, we give another example of application of the same method. This second example witnesses the scope of the method, though we do not have a general class of primitives to which it could be applied.

Application to formal verification of security protocols. In general, security protocols are specified in a process algebra like applied pi-calculus [1]. When one is interested in analysing a bounded number of sessions of the protocol, the set of all possible interleavings of actions is finite and can be determined from the specification. Each interleaving determines an infinite

set of traces corresponding to all possible executions of the protocol in that context. As shown for instance in [12], this set of traces can be symbolically represented by a deducibility constraint system $\mathcal{C}$: every solution of $\mathcal{C}$ corresponds to a trace. Then, verifying a trace-based property of the given security protocol amounts to deciding wether for all solution of $\mathcal{C}$ the property is satisfied. That is why we consider deducibility constraint systems as our formal model of security protocols.

Furthermore, we restrict our attention to a particular intruder theory, modeling blind signatures. There are at least two important applications of blind signatures. In electronic payments, they allow individuals to provide proof of payment, without third parties being able to trace the payee, time or amount of payment [5]. In electronic voting protocols, they allow administrators to check the eligibility of voters and sign their ballots, without being able to determine how voters have voted [14]. A realistic model of protocols will in general have to consider more cryptographic primitives, including for example the classical Dolev-Yao theory of encryption and pairing. While we could include the Dolev-Yao theory in our model, we prefer not to do it because: 1) this does not raise any technical challenge for our procedure and 2) we think the procedure of [10] could be combined with ours in the more general framework of combining decision procedures for disjoint intruder theories [8].

2. Preliminaries

2.1. Term algebra

Messages are represented by terms, constructed on an infinite set of *names* $\mathcal{N} = \{a, n, k, \ldots\}$, an infinite set of *variables* $\mathcal{X} = \{x, y, \ldots\}$ and a set $\mathcal{F}$ of function symbols. In this paper, $\mathcal{F} = \{\mathsf{blind}, \mathsf{sign}, \mathsf{vk}\}$ together with arities $ar(\mathsf{blind}) = ar(\mathsf{sign}) = 2$ and $ar(\mathsf{vk}) = 1$. The term $\mathsf{sign}(m, sk)$ represents the message m signed by the private key sk. The function blind is supposed to hide a message, thus the term $\mathsf{blind}(m, r)$ represents the blinding of m with the random r. This allows one to request a signature without revealing the content of the message.

We write $vars(t)$ for the set of variables occurring in t and $st(t)$ is the set of subterms of t. The *size* of a term t, denoted $|t|$, is the number of symbols occurring in it. *Substitutions* are written $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ with $dom(\sigma) = \{x_1, \ldots, x_n\}$. The application of a substitution σ to a term t is written $t\sigma$. We denote by $\#T$ the *cardinal* of the set T.

2.2. Inference system

The abilities of the attacker are modeled by a deduction system described in Figure 1. Intuitively, these deduction rules allow an intruder to compose messages by signing (rule *sign*) and blinding (rule *blind*) provided he has the corresponding keys. Conversely, he can decompose messages provided he holds the corresponding keys. For signatures, the intruder is able to verify whether a signature $\mathsf{sign}(m, sk)$ and a message m match (provided he has the verification key $\mathsf{vk}(sk)$), but this does not give him any new message. That is why this capability is not represented in the deduction system. We also consider the rule *getmsg* that expresses that an intruder can retrieve the whole message from its signature. The rule *unblind* allows one to retrieve the message m from $\mathsf{blind}(m, r)$ provided one knows the term r that has been used to hide m. Finally, the rule *unbdsign* with $n = 1$ allows one to obtain a signature from a signature of a blinded message m, once the term used to blind m is known.

$$\text{Rule } \textit{blind}\quad \frac{x \quad y}{\mathsf{blind}(x,y)} \qquad \text{Rule } \textit{unblind}\quad \frac{\mathsf{blind}(x,y) \quad y}{x} \qquad \text{Rule } \textit{sign}\quad \frac{x \quad y}{\mathsf{sign}(x,y)}$$

$$\text{Rule } \textit{getmsg}\quad \frac{\mathsf{sign}(x,y)}{x} \qquad \text{Rule } \textit{unbdsign}_n\quad \frac{\mathsf{sign}(\mathsf{b}_n(x,y_1,\ldots,y_n),z) \quad y_1 \quad \ldots \quad y_n}{\mathsf{sign}(x,z)} \quad n \geq 1$$

where $\mathsf{b}_1(u,v_1) = \mathsf{blind}(u,v_1)$ and $\mathsf{b}_{n+1}(u,v_1,\ldots,v_{n+1}) = \mathsf{blind}(\mathsf{b}_n(u,v_1,\ldots,v_n),v_{n+1})$.

Figure 1: Intruder deduction system for blind signatures

Definition 1 (deducible). *A* proof P *of* $T \vdash u$ *is a tree labeled with terms, whose leaves labels* $\mathsf{Hyp}(P)$ *are in* T*, its root label is* u*, and such that every intermediate node is an instance of one of the rules of Figure 1. A term* u *is* deducible from T *if there is a proof of* $T \vdash u$*, which we simply write* $T \vdash u$*.*

There is an infinite collection of rules *unbdsign* for $n > 1$, because we wish to get a *local* inference system: whenever there is a proof of $T \vdash u$, there is a local proof, whose all terms are in $st(T) \cup st(u)$. This infinite set of rules is necessary. Consider for instance the following proof:

$$\dfrac{\dfrac{\mathsf{sign}(\mathsf{blind}(\mathsf{blind}(u,r_1),r_2),sk) \quad r_2}{\mathsf{sign}(\mathsf{blind}(u,r_1),sk)} \quad r_1}{\mathsf{sign}(u,sk)}$$

The intermediate step $\mathsf{sign}(\mathsf{blind}(u,r_1),sk)$ is neither a subterm of the hypothesis nor a subterm of the conclusion. However, there is a local proof of the same term in the extended inference system, relying on the rule *unbdsign* with $n = 2$. In general, given a *non-local* inference system, we can obtain an equivalent (possibly infinite) local inference system by saturation, as shown in [4].

The rules *blind*, *sign*, and *unbdsign* are called *composition* rules whereas *getmsg* and *unblind* are called *decomposition* rules. For a set of terms $\{t_1,\ldots,t_n\}$, we will often use the notation $t_1,\ldots,t_n \vdash u$ instead of $\{t_1,\ldots,t_n\} \vdash u$.

2.3. Constraint systems

A *deducibility constraint* is an expression $T \stackrel{?}{\vdash} u$ where T is a finite set of terms and u is a term. The set T represents the intruder knowledge and u represents a term expected by an agent from the network. The solutions of such a constraint are the substitutions σ such that $T\sigma \vdash u\sigma$. The idea is to represent through such constraints all the possible executions of a protocol.

An introductory example. Consider a toy protocol in which A generates a nonce n, sends it to B, then B sends back this nonce signed with his private key k and finally A checks that the message she receives is $\mathsf{sign}(n,k)$. The possible traces obtained in any successful session of this protocol, between the two parties a, b are sequences of four messages:

$$n,\ x,\ \mathsf{sign}(x,k),\ \mathsf{sign}(n,k)$$

where n is the message sent by a, x is the message received by b, $\mathsf{sign}(x,k)$ is the message sent by b and $\mathsf{sign}(n,k)$ is the message received by a.

In an honest run, $x = n$. There are however other possible bindings of x, all yielding a valid trace. Actually, the only constraints that x must satisfy are:

$$\{n\} \stackrel{?}{\vdash} x \quad \text{and} \quad \{n,\ \mathsf{sign}(x,k)\} \stackrel{?}{\vdash} \mathsf{sign}(n,k)$$

Intuitively, the attacker should be able to construct x and to construct back $\mathsf{sign}(n,k)$ from n and $\mathsf{sign}(x,k)$.

Thus, traditionally (see *e.g.* [17, 10]), a constraint system is a set of deducibility constraints

$$T_1 \stackrel{?}{\vdash} u_1, \ldots, T_\ell \stackrel{?}{\vdash} u_\ell$$

such that the following conditions are satisfied:

1. *monotonicity*: $\emptyset \neq T_1 \subseteq T_2 \ldots \subseteq T_\ell$;
2. *origination*: for each $1 \leq i \leq \ell$, $vars(T_i) \subseteq vars(\{u_1, \ldots, u_{i-1}\})$;

Intuitively, the sequence of sets $T_1, \ldots, T_\ell$ represents the knowledge that is gathered by the intruder while interacting with the protocol. Thus, the monotonicity formalizes the increasing knowledge of the attacker: it is assumed that he keeps track of all available messages. For all $1 \leq i \leq \ell$, the term u_i represents a message that is expected to be received by an honest agent at a given step in the protocol: u_i can be a variable, when the agent will accept any message, or it can be a term with variables, when the agent will accept only messages of a given form. The set of terms $T_i \setminus T_{i-1}$ represents the set of messages that are sent by honest agents replying to $u_1, \ldots, u_{i-1}$. Thus, origination means that the reply of the agents depends on messages that they receive from the network.

A solution of such a constraint sytem $\mathcal{C}$ is a substitution θ such that $T\theta \vdash v\theta$ for each deducibility constraint $T \stackrel{?}{\vdash} v$ in $\mathcal{C}$. We denote by $Sol(\mathcal{C})$ the set of solutions of $\mathcal{C}$.

2.4. Overview of our procedure

Given a constraint system $\mathcal{C}$, we propose a set of rules that transforms $\mathcal{C}$ in a set of constraint systems in *solved form* $\mathcal{C}_1, \ldots, \mathcal{C}_n$. We rely on two main properties:

- the transformation rules preserve the set of solutions, *i.e* we have

  $$Sol(\mathcal{C}) = Sol(\mathcal{C}_1) \cup \ldots \cup Sol(\mathcal{C}_n)$$

  This property can also be decomposed into *soundness*, the solutions of each $\mathcal{C}_i$ are also solutions of $\mathcal{C}$, and *completeness*: any solution of $\mathcal{C}$ is a solution of some $\mathcal{C}_i$.

- a solved form always has a solution

A property of a set of traces, which is represented by a constraint $\mathcal{C}$, can then be reduced to a property of sets of traces, that are represented by solved forms. And deciding trace properties on solved forms is easy.

A crucial notion for the soundness and completeness of the transformation rules is locality (Section 3): whenever there is a proof of $T\sigma \vdash u\sigma$, there is a proof whose intermediary steps

are in $st(T\sigma) \cup st(u\sigma)$. Furthermore, origination and monotonicity of constraint systems allow us to restrict the search space to $st(T) \cup st(u)$, getting rid of σ (Section 6). An additional difficulty for the transformation rules comes from the fact that our local inference system is infinite, which is where our result fundamentally differs from [10]. Thus, we need to generalize the constraint systems in order to take into account an infinite set of constraints. That is why we introduce membership constraints (Section 4) whose semantics needs to be reflected by the transformation rules and in the definition of solved forms (Section 5).

3. Locality and simple proofs

To show that the inference system in Figure 1 satisfies locality (Lemma 1), we consider first a proof normalization procedure, described by the rules in Figure 2. These rules are terminating and confluent: each proof has a single normal form (a *normal proof*). We will show that every normal proof of $T \vdash u$ is *local*: all terms labeling its nodes are in $st(T) \cup st(u)$. The first two normalization rules state that in a normal proof unblinding should be performed on unsigned messages when possible. Note also that in both cases a non-local subproof is transformed into a local one. The third rule simply joins two consecutive applications of *unbdsign* into a single application of *unbdsign*. The last two rules remove unnecessary applications of *sign* and *blind*.

Definition 2 (normal proof). *A* normal proof *is a proof irreducible w.r.t. the rules given in Figure 2.*

Example 1. *Let* $T = \{a, r_1, r_2, \mathsf{sign}(\mathsf{blind}(\mathsf{blind}(a, r_1), r_2), sk)\}$. *A proof of* $T \vdash a$, *that uses an instance of the last rule scheme (with* $n = 2$*) is described below together with its normal form.*

$$\dfrac{\dfrac{\mathsf{sign}(\mathsf{blind}(\mathsf{blind}(a, r_1), r_2), sk) \quad r_1 \quad r_2}{\mathsf{sign}(a, sk)}}{a} \qquad \dfrac{\dfrac{\dfrac{\mathsf{sign}(\mathsf{blind}(\mathsf{blind}(a, r_1), r_2), sk)}{\mathsf{blind}(\mathsf{blind}(a, r_1), r_2)} \quad r_2}{\mathsf{blind}(a, r_1)} \quad r_1}{a}$$

Now, we can formally state the *locality property*. Getting such a property cannot be achieved with any finite subset of the inference rules of Figure 1.

Lemma 1 (locality). *Let* T *be a set of terms,* v *be a term, and* P *be a normal proof of* $T \vdash v$. *The proof* P *only contains terms in* $st(T \cup \{v\})$. *Moreover, if* P *is reduced to a leaf or ends with a decomposition rule then* $v \in st(T)$.

Proof. We prove this result by induction on P. The base case, *i.e.* when P consists of a single node is obvious. Indeed, in such a case, we immediately get that $v \in T$ and thus $v \in st(T)$. When P is not reduced to a leaf, we distinguish several cases:

- *P ends with an instance of the rule blind or sign.* In such a case, we easily conclude by relying on our induction hypothesis.

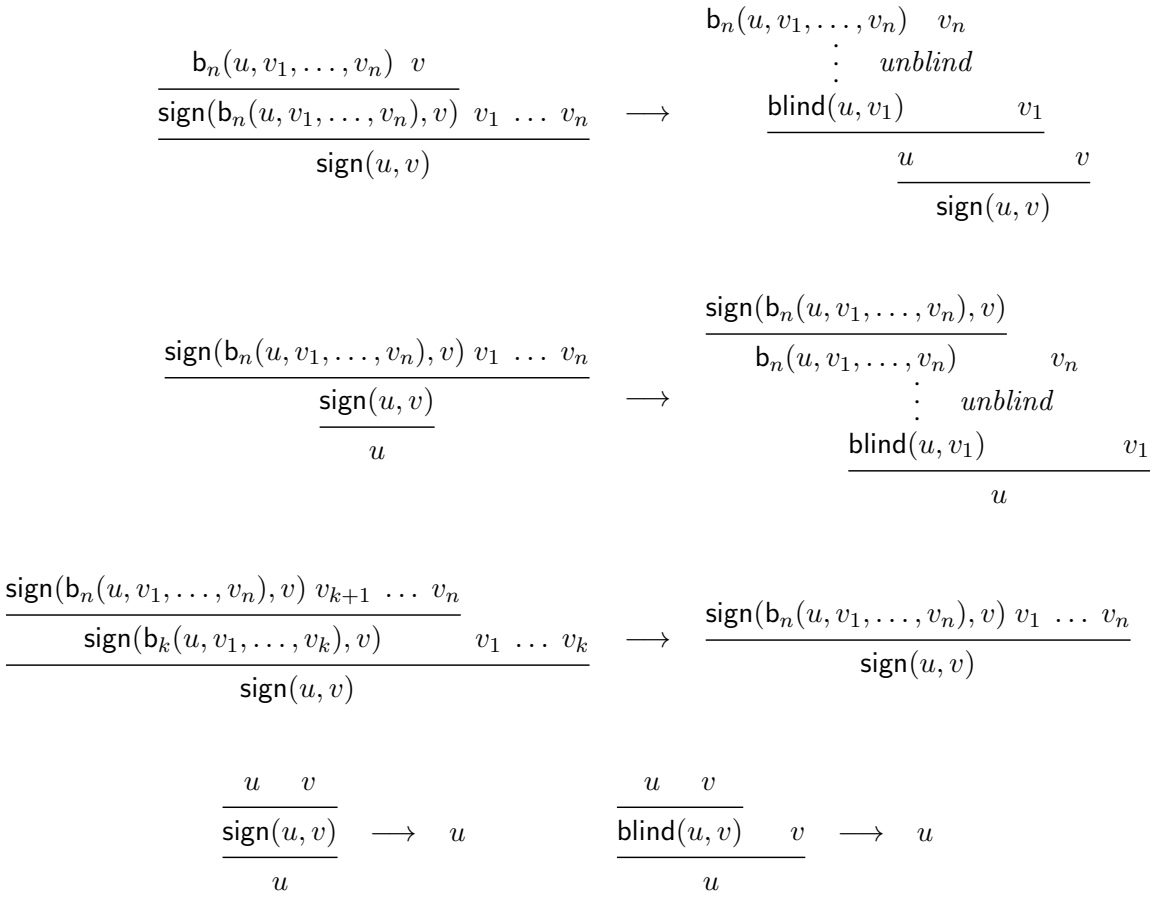

Figure 2: Proof normalization rules for blind signatures

- *P ends with an instance of a rule $unbdsign_k$ for some $k \geq 1$.* In such a case, the direct subproof P_0 of P labeled with $T \vdash \mathsf{sign}(\mathsf{b}_k(u, v_1, \ldots, v_k), v_0)$ ends with a decomposition rule. Indeed, the only composition rules that may be used are *sign* and *unbdsign*, which are not possible due to normalization rules. Therefore, thanks to our induction hypothesis, we have $\mathsf{sign}(\mathsf{b}_k(u, v_1, \ldots, v_k), v_0) \in st(T)$. This allows us to conclude.

- *P ends with an instance of the rule unblind.* In such a case, we have that the direct subproof P_1 of P whose root is labeled with $T \vdash \mathsf{blind}(v, u)$ ends with an instance of a decomposition rule. This is due to the fact that P is in normal form. Hence, we can apply our induction hypothesis on P_1. We deduce that $\mathsf{blind}(v, u) \in st(T)$ and thus $v \in st(T)$.

- *P ends with an instance of the rule getmsg.* In such a case, the direct subproof P_1 of P whose root is labeled with $T \vdash \mathsf{sign}(v, u)$ ends with an instance of a decomposition rule. Indeed, the rules *sign* and *unbdsign* are not possible since P is in normal form. Hence, we can apply our induction hypothesis on P_1. We deduce that $\mathsf{sign}(v, u) \in st(T)$ and thus $v \in st(T)$. □

From Lemma 1, we derive the following corollary. It will be useful later on to prove completeness of our decision procedure (see Section 6.3).

Corollary 1. *Let T be a set of terms and v be a term such that $T \vdash v$. Let $u \in st(v)$. Either $u \in st(T)$ or there exists a normal proof of $T \vdash u$ that ends with a composition rule.*

Proof. We prove the result by induction on P, a normal proof of $T \vdash v$. If P is reduced to a leaf, then $v \in T$ and thus $u \in st(T)$. Otherwise, we distinguish three cases:

- *P ends with a decomposition rule.* In such a case, thanks to Lemma 1, we deduce that $v \in st(T)$ and thus $u \in st(T)$.
- *P ends with an instance of the rule blind* (the case where P ends with an instance of the rule *sign* is similar). In such a case, we have that $v = \mathsf{blind}(v_1, v_2)$. Let P_1 (resp. P_2) be the direct subproof of P whose root is labeled with $T \vdash v_1$ (resp. $T \vdash v_2$). Either $u = v$ and we conclude that P is a normal proof of $T \vdash u$ that ends with a composition rule. Otherwise $u \in st(v_1)$ (or $u \in st(v_2)$) and in such a case, we conclude by applying our induction hypothesis on P_1 (or P_2).
- *P ends with an instance of the rule $unbdsign_k$ for some $k \geq 1$.* We have that $v = \mathsf{sign}(v_1, v_2)$. Let P_0 (resp. $P_1, \ldots, P_k$) be the direct subproofs of P whose root is labeled with $T \vdash \mathsf{sign}(\mathsf{b}_k(v_1, t_1, \ldots, t_k), v_2)$ (resp. $T \vdash t_1$, ..., $T \vdash t_k$). Note that, since P is in normal form, we have that P_0 is either reduced to a leaf or ends with a decomposition rule. Therefore, we can apply Lemma 1. We deduce that $v_1, v_2 \in st(T)$. Thus, either $u = v$ and we conclude that P is a normal proof of $T \vdash u$ that ends with a composition rule. Otherwise $u \in st(v_1)$ (or $u \in st(v_2)$) and in such a case, we easily conclude that $u \in st(T)$. □

We further distinguish between normal proofs that yield the same conclusion by considering *simple proofs*. Intuitively, given an increasing sequence of sets of terms $T_1 \subseteq \ldots \subseteq T_n$ and some index i, with $1 \leq i \leq n$, a proof of $T_i \vdash u$ is simple if it is in normal form and each of its subproofs uses a minimal set of hypotheses T_j, for some $j \leq i$:

Definition 3 (simple proof). *Consider a sequence of sets of terms $T_1, \ldots, T_n$, such that $T_1 \subseteq T_2 \subseteq \ldots \subseteq T_n$. A proof P of $T_i \vdash u$ is* left-minimal *(w.r.t. $T_1, \ldots, T_n$) if for any j such that $T_j \vdash u$, we have that $\mathsf{Hyp}(P) \subseteq T_j$. A proof P is* simple *if any of its subproofs is left-minimal and in normal form.*

Example 2. *Let $T_1 = \{a\}$. The normal proof given in Example 1 is not a simple proof w.r.t. the sequence $T_1 \subseteq T$. A simple proof of $T \vdash a$ is reduced to a leaf.*

In Appendix A, we show the following lemma.

Lemma 2. *Let $T_1 \subseteq T_2 \subseteq \ldots \subseteq T_n$ be a sequence of sets of terms. If there is a proof of $T_i \vdash u$ for some $i \in \{1, \ldots, n\}$, then there is a simple proof of $T_i \vdash u$ w.r.t. $T_1, \ldots, T_n$.*

4. Constraint systems with membership constraints

A *deducibility constraint* is an expression $T \stackrel{?}{\vdash} u$ where T is a finite set of terms and u is a term. The solutions of such a constraint are the substitutions σ such that $T\sigma \vdash u\sigma$. The idea is to represent through such constraints all the possible executions of a protocol.

4.1. An introductory example

Going back to our introductory example presented in Section 2.3, we have seen that the only constraints that x must satisfy are:

$$\{n\} \stackrel{?}{\vdash} x \quad \text{and} \quad \{n,\ \mathsf{sign}(x,k)\} \stackrel{?}{\vdash} \mathsf{sign}(n,k)$$

Intuitively, the attacker should be able to construct x and to construct back $\mathsf{sign}(n,k)$ from n and $\mathsf{sign}(x,k)$. The set of such possible messages x includes n, but also $\mathsf{blind}(n,n)$ for instance, since the attacker can unblind $\mathsf{sign}(\mathsf{blind}(n,n),k)$ and get $\mathsf{sign}(n,k)$. Actually, the set of possible messages x satisfying the above constraints is $\mathcal{B}d(\{n\},n)$ where $\mathcal{B}d(T,u)$ denotes the least set S of terms that contains u and such that $\mathsf{blind}(s,v) \in S$ when $s \in S$ and $T \vdash v$. Formally, we define

$$\mathcal{B}d(T,u) = \{u\} \cup \{\mathsf{b}_m(u,v_1,\ldots,v_m) \mid m \in \mathbb{N} \text{ and } T \vdash v_i \text{ for each } 1 \leq i \leq m\}.$$

For any term $t \in \mathcal{B}d(\{n\},n)$, the attacker can compute $\mathsf{sign}(n,k)$ from $\{n,\mathsf{sign}(t,k)\}$. using one instance of the *unbdsign* rule scheme as the last inference rule. We wish to use a single constraint solving step for all these possible final inference rules, hence we introduce an appropriate abstraction, enriching the syntax with membership constraints.

4.2. Constraint systems

We consider two different kinds of *elementary constraints*: a *deducibility constraint* is a constraint of the form $T \stackrel{?}{\vdash} u$, whereas a *membership constraint* is a constraint of the form $v \stackrel{?}{\in} \mathcal{B}d(T,u)$. In both cases, T is a finite set of terms and u, v are terms. Given an elementary constraint C of the form described above, the set of terms T is called the associated set of terms of the constraint C.

Intuitively, a membership constraint $v \stackrel{?}{\in} \mathcal{B}d(T,u)$ is a symbolic representation for an infinite set of deducibility constraints: it is true if, and only if, $u = v$ or there exists $n \geq 0$ such that the constraints $T \stackrel{?}{\vdash} x_1, \ldots, T \stackrel{?}{\vdash} x_n$, and $v = \mathsf{b}_n(u,x_1,\ldots,x_n)$ are satisfied.

Given a finite set $\mathcal{D}$ of elementary constraints and $x \in \mathit{vars}(\mathcal{D})$, we let T_x be the minimal set of terms w.r.t. inclusion (when it exists) such that

- $T_x \stackrel{?}{\vdash} u \in \mathcal{D}$ with $x \in \mathit{vars}(u)$, or
- $v \stackrel{?}{\in} \mathcal{B}d(T_x,u) \in \mathcal{D}$ with $x \in \mathit{vars}(u)$.

We extend the informal definition of constraint system introduced in Section 2.3 to consider constraint systems with membership constraints. As before, we require monotonicity and origination. We also require an additional condition on variables that we explain after the definition.

Definition 4 (constraint system). *A* constraint system $\mathcal{C}$ *is either* $\top$, $\bot$ *or a set of elementary constraints. We require that the constraints in* $\mathcal{C}$ *can be ordered* $C_1, \ldots, C_\ell$ *in such a way that the following conditions are satisfied:*

1. monotonicity*:* $\emptyset \neq T_1 \subseteq T_2 \ldots \subseteq T_\ell$*;*
2. origination*: for each* $1 \leq i \leq \ell$*,* $vars(T_i \cup \{v_i\}) \subseteq vars(\{u_1, \ldots, u_{i-1}\})$*;*

where each C_i *is of the form* $T_i \stackrel{?}{\vdash} u_i$ *or of the form* $v_i \stackrel{?}{\in} \mathcal{B}d(T_i, u_i)$*.*
Lastly, we assume that for each variable $x \in vars(\mathcal{C})$*:*

- *either there exists* $T_x \stackrel{?}{\vdash} u$ *in* $\mathcal{C}$ *with* $x \in vars(u)$*,*
- *or there exists* $v \stackrel{?}{\in} \mathcal{B}d(T_x, u)$ *in* $\mathcal{C}$ *with* $x \in vars(u)$ *and such that* $T_y \subsetneq T_x$ *for every* $y \in vars(v)$*.*

The new condition on variables can be seen as an extension of origination to membership constraints. Intuitively, for all variable x, the set T_x represents the associated set of terms for the constraint that first introduces x. In a constraint system $\mathcal{C}$ that only contains deducibility constraints, it is always the case that there exists $T_x \stackrel{?}{\vdash} u$ in $\mathcal{C}$ with $x \in vars(u)$. The second item of the new condition allows for variables to be introduced in membership constraints as some deducibility constraints are transformed in membership constraints. Typically, the condition will hold because for each new constraint $v \stackrel{?}{\in} \mathcal{B}d(T_x, u)$ it will be the case that $v \in st(T_x)$.

A constraint system $\mathcal{C}$ can also be seen as a conjunction of elementary constraints. Then, we consider that $\mathcal{C} \wedge \top$ is equivalent with $\mathcal{C}$ and $\mathcal{C} \wedge \bot$ is equivalent with $\bot$.

Example 3. *The following set of constraints*

$$\{a \stackrel{?}{\vdash} y,\ \ \mathsf{blind}(y, a) \stackrel{?}{\in} \mathcal{B}d(\{a\}, x)\}$$

satisfies monotonicity and origination. However, it is not a constraint system. Indeed, the constraint $\mathsf{blind}(y, a) \stackrel{?}{\in} \mathcal{B}d(\{a\}, x)$ *introduces the variable* x *and the condition* $T_y \subsetneq \{a\}$ *is not satisfied. Actually, we have that* $T_y = \{a\}$*.*

Definition 5 (solution). *A* solution *of a set* $\mathcal{D}$ *of elementary constraints is a substitution* θ *such that* $T\theta \vdash v\theta$ *for each* $T \stackrel{?}{\vdash} v \in \mathcal{D}$*, and* $v\theta \in \mathcal{B}d(T\theta, u\theta)$ *for each* $v \stackrel{?}{\in} \mathcal{B}d(T, u) \in \mathcal{D}$*. We denote by* $Sol(\mathcal{D})$ *the set of solutions of* $\mathcal{D}$*. The constraint system* $\bot$ *has no solution and any substitution is a solution of* $\top$*.*

Given two sets $\mathcal{D}$ and $\mathcal{D}'$ of constraints, we write $\mathcal{D} \models \mathcal{D}'$ if $Sol(\mathcal{D}) \subseteq Sol(\mathcal{D}')$. We denote by $\mathcal{D}|_V$ the constraints in $\mathcal{D}$ that only contain variables in the set V, *i.e.*

$$\mathcal{D}|_V = \{C \in \mathcal{D} \mid vars(C) \subseteq V\}.$$

We will show that we can restrict ourselves to solutions that do not map two distinct subterms of the constraint system to the same term. Let T be a set of terms. A substitution σ is *non-confusing w.r.t.* T if for any $t_1, t_2 \in st(T)$ such that $t_1 \neq t_2$, we have that $t_1\sigma \neq t_2\sigma$. A *non-confusing solution* of a set $\mathcal{D}$ of elementary constraints is a substitution $\theta \in Sol(\mathcal{D})$ such that θ is non-confusing w.r.t. terms that appear in $\mathcal{D}$. We denote by $Sol_{\mathsf{NC}}(\mathcal{D})$ the set of solutions of $\mathcal{D}$ that are non-confusing.

$$
\begin{array}{lrcll}
\mathsf{S}_{\mathsf{ax}}: & u \stackrel{?}{\in} \mathcal{B}d(T,u) & \rightarrow & \top & \\
\mathsf{S}_{\mathsf{bd}}: & \mathsf{blind}(u,v) \stackrel{?}{\in} \mathcal{B}d(T,w) & \rightarrow & T \stackrel{?}{\vdash} v \wedge u \stackrel{?}{\in} \mathcal{B}d(T,w) & \text{if } \mathsf{blind}(u,v) \neq w \\
\mathsf{S}_{\mathsf{f}}: & \mathsf{f}(t_1,\ldots,t_n) \stackrel{?}{\in} \mathcal{B}d(T,v) & \rightarrow & \bot & \text{if } \mathsf{f} \neq \mathsf{blind} \text{ and } \mathsf{f}(t_1,\ldots,t_n) \neq v \\
\mathsf{S}_{\mathsf{cycle}}: & x_1 \stackrel{?}{\in} \mathcal{B}d(T_1,v_1[x_2]) \wedge \ldots \wedge x_n \stackrel{?}{\in} \mathcal{B}d(T_n,v_n[x_1]) & \rightarrow & \bot & \\
 & & & & \text{if there exists } i \text{ such that } v_i \neq \epsilon \text{ or } \#\{x_1,\ldots,x_n\} > 1.
\end{array}
$$

Figure 3: Simplification rules for membership constraints

4.3. Simplified form

The constraints are simplified according to the simplification rules described in Figure 3. They reflect the semantics of $u \stackrel{?}{\in} \mathcal{B}d(T,v)$. In the rule $\mathsf{S}_{\mathsf{cycle}}$, we use the notation $v[x]$ to denote a term that contains the variable x. If, moreover, that term is different from x, we say that $v \neq \epsilon$.

Definition 6 (simplified form). *A set $\mathcal{D}$ of elementary constraints is in* simplified form *if none of the simplification rules can be applied. $\mathcal{D}{\downarrow_S}$ is the set of irreducible constraints obtained from $\mathcal{D}$ by repeatedly applying these rules.*

Note that in a set of constraints in simplified form, each membership constraint is of the form $x \stackrel{?}{\in} \mathcal{B}d(T,u)$ where x is a variable. Moreover, we show in Appendix B that these simplification rules transform a constraint system into a constraint system and preserve the set of solutions:

Lemma 3. *Let $\mathcal{D}$ and $\mathcal{D}'$ be two sets of elementary constraints such that $\mathcal{D} \rightarrow \mathcal{D}'$. We have that:*

- *If $\mathcal{D}$ is a constraint system then $\mathcal{D}'$ is a constraint system;*
- *$Sol(\mathcal{D}') \subseteq Sol(\mathcal{D})$ and $Sol_{\mathsf{NC}}(\mathcal{D}) \subseteq Sol_{\mathsf{NC}}(\mathcal{D}')$.*

From this lemma, we easily derive the two following results.

Corollary 2. *Let $\mathcal{D}$ be a set of elementary constraints. We have that:*

- *If $\mathcal{D}$ is a constraint system then $\mathcal{D}{\downarrow_S}$ is a constraint system;*
- *$Sol(\mathcal{D}{\downarrow_S}) \subseteq Sol(\mathcal{D})$ and $Sol_{\mathsf{NC}}(\mathcal{D}) \subseteq Sol_{\mathsf{NC}}(\mathcal{D}{\downarrow_S})$.*

Corollary 3. *Let $\mathcal{D}$ and $\mathcal{D}'$ be two sets of constraints such that $\mathcal{D} \rightarrow^* \mathcal{D}'$ and $V \subseteq vars(\mathcal{D})$. We have that $Sol(\mathcal{D}'|_V) \subseteq Sol(\mathcal{D}|_V)$.*

Proof. We prove the result for the case when there is a single simplification step and conclude the proof by induction on the length of the derivation. If there is a single simplification step $\mathcal{D} \rightarrow \mathcal{D}'$, we are in one of the four following cases: $\mathcal{D}|_V \rightarrow \mathcal{D}'|_V$ (when the rule S_{bd} is applied to a constraint in $\mathcal{D}|_V$), or $\mathcal{D}|_V \subseteq \mathcal{D}'|_V$ (when the rule S_{bd} is applied to a constraint in $\mathcal{D} \setminus \mathcal{D}|_V$),

or the rule $\mathsf{S_{ax}}$ is applied, or $\mathcal{D}'|_V = \bot$ (when one of the rules $\mathsf{S_f}, \mathsf{S_{cycle}}$ is applied). In the last three situations, we trivially have $Sol(\mathcal{D}'|_V) \subseteq Sol(\mathcal{D}|_V)$. In the first situation, we conclude using the Lemma 3. □

5. Constraint systems in solved form

Our aim is to design a set of transformation rules that rewrite any constraint into a finite set of *solved forms*, which are a more convenient representation of the same set of solutions.

Definition 7 (solved form). *A constraint system $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ is in* solved form *if each C_i is either of the form $T_i \overset{?}{\vdash} x_i$ or of the form $x_i \overset{?}{\in} \mathcal{B}d(T_i, u_i)$ where x_i is a variable. Moreover, for every $x \in vars(\mathcal{C})$, there is a unique deducibility constraint $T \overset{?}{\vdash} x \in \mathcal{C}$ and there is at most one membership constraint $x \overset{?}{\in} \mathcal{B}d(T', u)$ in $\mathcal{C}$ and, if this is the case, $T' = T$.*

Example 4. *Below, the systems $\mathcal{C}_2$ and $\mathcal{C}_3$ are in solved form whereas $\mathcal{C}_1$ is not.*

$$\mathcal{C}_1 = \left\{ \begin{array}{lcl} a & \overset{?}{\vdash} & y \\ y & \overset{?}{\in} & \mathcal{B}d(\{a\}, a) \\ y & \overset{?}{\in} & \mathcal{B}d(\{a\}, \mathsf{blind}(a, a)) \end{array} \right. \quad \mathcal{C}_2 = \left\{ \begin{array}{lcl} a & \overset{?}{\vdash} & x \\ x & \overset{?}{\in} & \mathcal{B}d(\{a\}, b) \end{array} \right. \quad \mathcal{C}_3 = \left\{ \begin{array}{lcl} a & \overset{?}{\vdash} & x \\ a, x & \overset{?}{\vdash} & y \\ y & \overset{?}{\in} & \mathcal{B}d(\{a, x\}, x) \end{array} \right.$$

Note that a constraint system in solved form is not necessarily satisfiable. For instance, the system $\mathcal{C}_2$ has no solution.

Fortunately, the constraints that are produced by our transformation rules satisfy an additional invariant, which we explain now.

Well-formed constraint systems. Let $\mathcal{D}$ be a set of constraints in simplified form. We define $\leq_\mathcal{D}$ on $vars(\mathcal{D})$ as the least relation closed by transitivity and reflexivity and such that:

$$y \overset{?}{\in} \mathcal{B}d(T, u) \text{ in } \mathcal{D} \text{ and } x \in vars(u) \Longrightarrow x \leq_\mathcal{D} y.$$

Intuitively, note that if $x \leq_\mathcal{D} y$, then for any solution σ of $\mathcal{D}$, we have $x\sigma \in st(y\sigma)$: proofs whose conclusion contains $y\sigma$ may depend on proofs whose conclusion contains $x\sigma$. Note that, due to the rule $\mathsf{S_{cycle}}$ and the fact that $\mathcal{D}$ is in simplified form, if $y \leq_\mathcal{D} x$ and $x \leq_\mathcal{D} y$ then $x = y$. Hence $\leq_\mathcal{D}$ is an ordering. Moreover, this ordering is compatible with the monotonicity, as we show in the following lemma. This will allow us to iteratively construct a solution σ of a solved form $\mathcal{D}$ by extending it from y to x, when $y \leq_\mathcal{D} x$.

Lemma 4. *Let $\mathcal{C}$ be a constraint system in simplified and solved form. If $x \leq_\mathcal{C} y$, then $T_x \subseteq T_y$.*

Proof. Let x and y be two variables such that $x \leq_\mathcal{C} y$. By definition of $\leq_\mathcal{C}$, we know that there exists n, and some membership constraints in $\mathcal{C}$:

$$x_1 \overset{?}{\in} \mathcal{B}d(T_1, u_1),\ x_2 \overset{?}{\in} \mathcal{B}d(T_2, u_2),\ \ldots,\ x_n \overset{?}{\in} \mathcal{B}d(T_n, u_n)$$

such that $y = x_1$, $x_{i+1} \in vars(u_i)$ for $1 \leq i < n$, and $x \in vars(u_n)$.

By the definition of solved forms, for every i, there is a unique deducibility constraint $T_i \overset{?}{\vdash} x_i$ in $\mathcal{C}$. It follows that, for every i, $T_{x_i} = T_i$.

By definition of T_x and T_{x_i} $(1 \leq i \leq n)$, we have that $T_x \subseteq T_n$ and $T_{x_i} \subseteq T_{i-1}$ $(1 < i \leq n)$. Hence, we deduce that:

$$T_x \subseteq T_n = T_{x_n} \subseteq T_{n-1} = T_{x_{n-1}} \subseteq \ldots T_2 = T_{x_2} \subseteq T_1 = T_{x_1} = T_y.$$

This allows us to conclude. □

We extend the partial order $\leq_{\mathcal{D}}$ to a pre-order on sets of variables as follows:

$$V_1 \preceq_{\mathcal{D}} V_2 \text{ if, and only if, } \forall x \in V_1,\ \exists y \in V_2 \text{ such that } x \leq_{\mathcal{D}} y.$$

We denote by $\mathcal{C}^{\preceq}_{V}$ the set of constraints in $\mathcal{C}$ containing only variables smaller or equal by $\preceq_{\mathcal{C}}$ to those in V, *i.e.* $\mathcal{C}^{\preceq}_{V} = \{C \in \mathcal{C} \mid vars(C) \preceq_{\mathcal{C}} V\}$. Note that $\mathcal{C}^{\preceq}_{V}$ is not necessarily a constraint system. The goal of $\mathcal{C}^{\preceq}_{V}$ is to determine all the constraints of $\mathcal{C}$ that affect the possible values of V in a solution of $\mathcal{C}$.

Definition 8 (well-formed). *A simplified constraint system $\mathcal{C}$ is* well-formed *if, for every constraint $y \overset{?}{\in} \mathcal{B}d(T_i, u_i)$ in $\mathcal{C}$, either $T_y \subsetneq T_i$ or else $T_y = T_i$ and $\mathcal{C}^{\preceq}_{V} \models (T_i \overset{?}{\vdash} u_i)$ where $V = vars(T_i \cup \{u_i\})$.*

Intuitively, well-formedness will help us to solve membership constraints. If a constraint $y \overset{?}{\in} \mathcal{B}d(T_i, u_i)$ is such that $T_y \subsetneq T_i$, we will show that it can be transformed into a simpler constraint, with an associated set of terms T_y. Otherwise, $\mathcal{C}^{\preceq}_{V} \models (T_i \overset{?}{\vdash} u_i)$ ensures us that any solution of $\mathcal{C}^{\preceq}_{V}$ can be extended to a solution of $y \overset{?}{\in} \mathcal{B}d(T_i, u_i)$.

Example 5. *The constraint system $\mathcal{C}_2$ (see Example 4) is not well-formed. Indeed, $T_x = \{a\}$ and thus the first condition does not hold. Moreover, $\mathcal{C}^{\preceq}_{V} = \emptyset$ and $\emptyset \not\models (a \overset{?}{\vdash} b)$. The systems $\mathcal{C}_1$ and $\mathcal{C}_3$ are well-formed.*

Lemma 5. *Any solved well-formed simplified constraint system $\mathcal{C}$ has at least one solution. Moreover, if $t_1, \ldots, t_m, u_1, \ldots, u_m$ are sequences of terms such that, for every i, t_i is distinct from u_i, then $\mathcal{C} \wedge t_1 \neq u_1 \wedge \cdots \wedge t_m \neq u_m$ has a solution.*

Proof. We consider an ordering between variables defined as follows:

$$x \leq y \text{ if, and only if, } \begin{cases} \text{either } T_x \subsetneq T_y, \\ \text{or } T_x = T_y \text{ and } x \leq_{\mathcal{C}} y. \end{cases}$$

Thanks to Lemma 4, we know that $x \leq_{\mathcal{C}} y$ implies $T_x \subseteq T_y$, and thus $x \leq y$. Hence, we know that $\leq$ is compatible with $\leq_{\mathcal{C}}$.

Let $x_1, \ldots, x_n$ be the variables in $\mathcal{C}$ renamed in such a way that $x_i \leq x_j$ implies $i \leq j$. We consider the constraint system $\mathcal{C}$ in which the constraints are ordered according to the sequence $x_1, \ldots, x_n$, *i.e.*

$$\mathcal{C} := \begin{cases} T_1 \overset{?}{\vdash} x_1 \ \wedge \ [x_1 \overset{?}{\in} \mathcal{B}d(T_1, u_1)] \\ \cdots \\ T_n \overset{?}{\vdash} x_n \ \wedge \ [x_n \overset{?}{\in} \mathcal{B}d(T_n, u_n)] \end{cases}$$

The notation $[x \overset{?}{\in} \mathcal{B}d(T, u)]$ is used to indicate that this part is optional. Thanks to the previous observation, it is clear that this ordering satisfies monotonicity. Actually we show that this ordering satisfies also origination and thus this ordering is a witness of the fact that $\mathcal{C}$ is a constraint system. To prove this, we first have to show that $vars(T_i) \subseteq \{x_1, \ldots, x_{i-1}\}$. Let $y \in vars(T_i)$. We have that $T_y \subsetneq T_i$ and thus $y \leq x_i$, *i.e.* $y \in \{x_1, \ldots, x_{i-1}, x_i\}$. Actually, $y \neq x_i$ since $\mathcal{C}$ is in solved form. This allows us to conclude. Secondly, we also have that $vars(u_i) \subseteq \{x_1, \ldots, x_{i-1}\}$. Indeed, let $y \in vars(u_i)$, we have that $y \leq x_i$, *i.e.* $y \in \{x_1, \ldots, x_{i-1}\}$ (again relying on the fact that $y \neq x_i$).

We show, by induction on n (the number of variables in the constraint system) that there is a substitution $\sigma \in Sol(\mathcal{C})$ such that, for every $1 \leq i \leq m$, $t_i\sigma \neq u_i\sigma$.

Base case: $n = 0$. Then $\mathcal{C}$ is the trivially satisfied formula. Since, for every j, t_j is distinct from u_j, the trivial (empty) substitution is a solution.

Induction step: Let x_n be a maximal variable. By induction hypothesis, there is a substitution θ, that is a solution of $T_i \overset{?}{\vdash} x_i \ \wedge \ [x_i \overset{?}{\in} \mathcal{B}d(T_i, u_i)]$ for $i < n$ and such that $t_j\theta$ and $u_j\theta$ are distinct for every j. Each equation $t_j\theta = u_j\theta$, with a single unkown x_n has at most one solution v_j. We distinguish several cases:

- If there is no constraint $T_n \overset{?}{\vdash} x_n$ in $\mathcal{C}$, then we may simply choose $\sigma = \theta \cup \{x_n \mapsto v\}$ for any $v \notin \{v_1, \ldots, v_m\}$ and we get a solution.

- If there is a constraint $T_n \overset{?}{\vdash} x_n$ in $\mathcal{C}$ but no constraint $x_n \overset{?}{\in} \mathcal{B}d(T_n, u_n)$, we let $w_0 \in T_n\theta$ and $w_{k+1} = \mathsf{sign}(w_k, w_0)$. For every k, $T_n\theta \vdash w_k$ and there is at least one $k_0 \leq m$ such that $w_{k_0} \notin \{v_1, \ldots, v_m\}$. Let $\sigma = \theta \cup \{x_n \mapsto w_{k_0}\}$. Since $x_n \notin vars(T_n\theta)$, $T_n\sigma \vdash x_n\sigma$ and since $x_n\sigma \notin \{v_1, \ldots, v_m\}$, $t_j\sigma$ and $u_j\sigma$ are distinct for every j. Hence σ has the expected property.

- Now, assume that there is a constraint $T_n \overset{?}{\vdash} x_n$ and a constraint $x_n \overset{?}{\in} \mathcal{B}d(T_n, u_n)$ in $\mathcal{C}$. We have shown that $vars(T_n \cup \{u_n\}) \subseteq \{x_1, \ldots, x_{n-1}\}$. We let $w_0 \in T_n$ and, for any $k \geq 1$, we let $w_k = \mathsf{b}_k(u_n\theta, w_0\theta, \ldots, w_0\theta)$. Then, by well-formedness of $\mathcal{C}$, $(\mathcal{C} \setminus \{T_n \overset{?}{\vdash} x_n\}) \models T_n \overset{?}{\vdash} u_n$, hence $T_n\theta \vdash u_n\theta$ and therefore $T_n\theta \vdash w_k$ for every k. Furthermore, $w_k \in \mathcal{B}d(T_n\theta, u_n\theta)$ for every k. For at least one $1 \leq k_0 \leq m + 1$, $w_{k_0} \notin \{v_1, \ldots, v_m\}$. Let $\sigma = \theta \cup \{x_n \mapsto w_{k_0}\}$. Then $T_n\sigma \vdash x_n\sigma$, $x_n\sigma \in \mathcal{B}d(T_n\sigma, u_n\sigma)$ and $t_j\sigma$ is distinct from $u_j\sigma$ for every j. □

6. Transformation rules

The constraint solving rules are displayed in Figure 4. The rules $\mathsf{R_{ax}}, \mathsf{R_{triv}}, \mathsf{R_f}, \mathsf{R_{bd}}, \mathsf{R_{get}}$, and $\mathsf{R_{bdsgn}}$ will be applied when the corresponding inference rules end the proof of an unsolved

$$\mathsf{R}_{\mathsf{ax}}: \quad T \overset{?}{\vdash} u \rightsquigarrow \top \qquad \text{if } u \in T \setminus \mathcal{X}$$

$$\mathsf{R}_{\mathsf{triv}}: \quad T \overset{?}{\vdash} x \wedge T' \overset{?}{\vdash} x \rightsquigarrow T \overset{?}{\vdash} x \qquad \text{if } T \subseteq T'$$

$$\mathsf{R}_{\mathsf{f}}: \quad T \overset{?}{\vdash} \mathsf{f}(t_1, \ldots, t_n) \rightsquigarrow T \overset{?}{\vdash} t_1, \ldots, T \overset{?}{\vdash} t_n \qquad \mathsf{f} \in \{\mathsf{sign}, \mathsf{blind}\}$$

$$\mathsf{R}_{\mathsf{bd}}: \quad T \overset{?}{\vdash} v \rightsquigarrow T \overset{?}{\vdash} \mathsf{blind}(v, u) \wedge T \overset{?}{\vdash} u \qquad \text{if } \mathsf{blind}(v, u) \in st(T)$$

$$\mathsf{R}_{\mathsf{get}}: \quad T \overset{?}{\vdash} v \rightsquigarrow T \overset{?}{\vdash} \mathsf{sign}(v, u) \qquad \text{if } \mathsf{sign}(v, u) \in st(T)$$

$$\mathsf{R}_{\mathsf{bdsgn}}: \quad T \overset{?}{\vdash} \mathsf{sign}(v, u) \rightsquigarrow T \overset{?}{\vdash} \mathsf{sign}(w, u) \wedge w \overset{?}{\in} \mathcal{B}d(T, v) \qquad \text{if } \mathsf{sign}(w, u) \in st(T)$$

$$\mathsf{R}_{\mathsf{A}}: \quad T \overset{?}{\vdash} x \wedge x \overset{?}{\in} \mathcal{B}d(T', v) \rightsquigarrow T \overset{?}{\vdash} x \wedge T \overset{?}{\vdash} v \wedge x \overset{?}{\in} \mathcal{B}d(T, v) \qquad \text{if } T \subsetneq T'$$

$$\mathsf{R}_{\mathsf{B}}: \quad T \overset{?}{\vdash} x \wedge x \overset{?}{\in} \mathcal{B}d(T', v) \rightsquigarrow T \overset{?}{\vdash} x \wedge T \overset{?}{\vdash} w \wedge x \overset{?}{\in} \mathcal{B}d(T, w) \wedge w \overset{?}{\in} \mathcal{B}d(T', v) \qquad \text{if } T \subsetneq T' \text{ and } w \in st(T)$$

$$\mathsf{R}_{\mathsf{C}}: \quad T \overset{?}{\vdash} x \wedge x \overset{?}{\in} \mathcal{B}d(T, v) \wedge x \overset{?}{\in} \mathcal{B}d(T, v') \rightsquigarrow T \overset{?}{\vdash} x \wedge x \overset{?}{\in} \mathcal{B}d(T, v) \wedge v \overset{?}{\in} \mathcal{B}d(T, v') \qquad \text{if } T_x = T$$

Figure 4: Transformation rules

deducibility constraint. Note that $\mathsf{R}_{\mathsf{bdsgn}}$ introduces a membership constraint, that captures the infinite set of inference rules $unbdsign_n, n \geq 1$. Furthermore, the rules $\mathsf{R}_{\mathsf{A}}, \mathsf{R}_{\mathsf{B}}, \mathsf{R}_{\mathsf{C}}$ transform membership constraints. The rule R_{A} will be applied when a membership constraint can be satisfied with a smaller set of hypotheses. The rule R_{B} will be applied when only a part of a membership constraint can be satisfied with a smaller set of hypotheses. Finally, the rule R_{C} will be applied when two membership constraints overlap.

Implicitly in what follows, every set of elementary constraints obtained after applying a transformation rule is put in simplified form. We show soundness (see Section 6.1) and completeness (see Section 6.3) of our set of transformation rules. We also show that the notion of well-formedness introduced in the previous section is an invariant (see Section 6.2).

6.1. Soundness

We show that our rules transform a constraint system into a constraint system (Lemma 6) and we show in Lemma 7 that our rules are sound, *i.e.* when $\mathcal{C} \rightsquigarrow \mathcal{C}'$, we have that $Sol(\mathcal{C}') \subseteq Sol(\mathcal{C})$.

Lemma 6. *The rules of Figure 4 transform a constraint system into a constraint system, i.e. if $\mathcal{C}$ is a simplified constraint system and $\mathcal{C} \rightsquigarrow \mathcal{C}'$ then $\mathcal{C}'{\downarrow_S}$ is a constraint system. Moreover, we have that $st(\mathcal{C}'{\downarrow_S}) \subseteq st(\mathcal{C})$.*

Proof. Let $\mathcal{C}$ be simplified constraint system and $\mathcal{C}'$ be such that $\mathcal{C} \rightsquigarrow \mathcal{C}'$. We want to show that $\mathcal{C}'$ is a constraint system. First, it is clear that monotonicity and origination are preserved. Now, let us check that the condition stated in Definition 4 for variables is also satisfied. The rule R_{f} does not cause any trouble. The rules R_{ax}, $\mathsf{R}_{\mathsf{triv}}$, R_{bd}, and $\mathsf{R}_{\mathsf{get}}$ affect only

elementary constraints that do not introduce variables for the first time. If the rule $\mathsf{R_{bdsgn}}$ is applied, then even if the additional membership constraint $w \overset{?}{\in} \mathcal{B}d(T, v)$ introduces a variable for the first time, we have that $T_y \subsetneq T$ for each $y \in vars(w)$ since $w \in st(T)$. This allows us to conclude in this case. In case of $\mathsf{R_A}$ (resp. $\mathsf{R_B}$), the additional membership constraint on x does not introduce any variable because of the presence of the deducibility constraint $T \overset{?}{\vdash} v$ (resp. $T \overset{?}{\vdash} w$). In case of $\mathsf{R_B}$, assume that the membership constraint $w \overset{?}{\in} \mathcal{B}d(T', v)$ introduces a variable. We have that $T_y \subsetneq T \subsetneq T'$ for each $y \in vars(w)$. This is due to the fact that $w \in st(T)$. In case of $\mathsf{R_C}$, the additional membership constraint satisfies the requirement since v' can not introduce any variable. Indeed, otherwise, we would have that the sytem on which we apply this rule does not satisfy the condition on membership constraint. This allows us to conclude that $\mathcal{C}'$ is a constraint system. Then, we deduce that $\mathcal{C}'{\downarrow_S}$ is a constraint system thanks to Corollary 2. Moreover, if $\mathcal{C} \rightsquigarrow^* \mathcal{C}'$, it is clear that $st(\mathcal{C}') \subseteq st(\mathcal{C})$ since the rules never introduce a new subterm. □

Lemma 7 (soundness). *Let $\mathcal{D}$ be a set of elementary constraints in simplified form and $\mathcal{D}'$ be a set of constraints such that $\mathcal{D} \rightsquigarrow \mathcal{D}'$. We have that $Sol(\mathcal{D}'{\downarrow_S}) \subseteq Sol(\mathcal{D})$.*

Proof. Let R be the transformation rule used in the step $\mathcal{D} \rightsquigarrow \mathcal{D}'$ and $\sigma \in Sol(\mathcal{D}'{\downarrow_S})$. First, thanks to Lemma 3, we have that $\sigma \in Sol(\mathcal{D}')$. Then we show that $\sigma \in Sol(\mathcal{D})$ by case analysis on R.

Case $\mathsf{R_{ax}}$, $\mathsf{R_{triv}}$, $\mathsf{R_f}$, $\mathsf{R_{bd}}$, $\mathsf{R_{get}}$, *and* $\mathsf{R_{bdsgn}}$: The proof trees witnessing the fact that $\sigma \in Sol(\mathcal{D})$ are easily obtained from those witnessing the fact that $\sigma \in Sol(\mathcal{D}')$.

Case $\mathsf{R_A}$: The proof trees witnessing the fact that $\sigma \in Sol(\mathcal{D}')$ can be used to show that $\sigma \in Sol(\mathcal{D})$. The proof tree witnessing $T \overset{?}{\vdash} v$ is even not useful for that.

Case $\mathsf{R_B}$: We have to group together the sequences of proof trees witnessing $x\sigma \overset{?}{\in} \mathcal{B}d(T\sigma, w\sigma)$ and $w\sigma \overset{?}{\in} \mathcal{B}d(T'\sigma, v\sigma)$ in order to obtain a witness of the fact that $x\sigma \in \mathcal{B}d(T'\sigma, v\sigma)$.

Case $\mathsf{R_C}$: We have to group together the sequences of proof trees witnessing $x\sigma \overset{?}{\in} \mathcal{B}d(T\sigma, v\sigma)$ and $v\sigma \overset{?}{\in} \mathcal{B}d(T\sigma, v'\sigma)$ in order to show that $x\sigma \in \mathcal{B}d(T\sigma, v'\sigma)$. □

From the lemma above, we easily derive the following result.

Proposition 1 (soundness). *Let $\mathcal{D}$ and $\mathcal{D}'$ be two sets of elementary constraints in simplified form. If $\mathcal{D} \rightsquigarrow^* \mathcal{D}'$ and $\sigma \in Sol(\mathcal{D}')$ then $\sigma \in Sol(\mathcal{D})$.*

6.2. Well-formed

The goal of this section is to show that our rules transform a well-formed constraint system into a well-formed constraint system. To establish this invariant, we first prove some useful properties. Proofs of Lemma 8 and Proposition 2 are detailed in Appendix C.

Lemma 8 (property of $\leq_{\mathcal{D}}$). *Let $\mathcal{D}$ and $\mathcal{D}'$ be two sets of constraints in simplified form such that $\mathcal{D} \rightsquigarrow \mathcal{D}'$ and $\mathcal{D}' \neq \bot$. We have that $\leq_{\mathcal{D}} \subseteq \leq_{\mathcal{D}'}$.*

Proposition 2. *Let $\mathcal{D}$ and $\mathcal{D}'$ be two sets of constraints in simplified form such that $\mathcal{D} \rightsquigarrow \mathcal{D}'$, and $V \subseteq vars(\mathcal{D})$. We have that $\mathcal{D}'^{\prec}_V \models \mathcal{D}^{\prec}_V$.*

Proposition 3. *Let $\mathcal{C}$ be a simplified constraint system that is well-formed and such that $\mathcal{C} \rightsquigarrow \mathcal{C}'$. Then $\mathcal{C}'{\downarrow_S}$ is a well-formed constraint system.*

Proof.

We consider each possible transformation rule applied on $\mathcal{C}$ and show that each membership constraint $M = x \overset{?}{\in} \mathcal{B}d(T, u)$ in $\mathcal{C}'{\downarrow_S}$ is such that:

- either $T_x \subsetneq T$,
- or $T_x = T$ and $(\mathcal{C}'{\downarrow_S})^{\preceq}_{\overrightarrow{V}} \models (T \overset{?}{\vdash} u)$ where $V = vars(T \cup \{u\})$.

Let us first consider a membership constraint $M = x \overset{?}{\in} \mathcal{B}d(T, u)$ that is also in $\mathcal{C}$. If we have that $T_x \subsetneq T$ in the constraint system $\mathcal{C}$ then $T_x \subsetneq T$ also holds in the constraint system $\mathcal{C}'{\downarrow_S}$ and we conclude. Otherwise, we have that $T_x = T$ (in $\mathcal{C}$) and $\mathcal{C}^{\preceq}_{\overrightarrow{V}} \models (T \overset{?}{\vdash} u)$. Applying Proposition 2 we deduce that $(\mathcal{C}'{\downarrow_S})^{\preceq}_{\overrightarrow{V}} \models \mathcal{C}^{\preceq}_{\overrightarrow{V}}$, and thus $(\mathcal{C}'{\downarrow_S})^{\preceq}_{\overrightarrow{V}} \models (T \overset{?}{\vdash} u)$. Therefore, in the following, we concentrate only on membership constraints that are in $\mathcal{C}'{\downarrow_S}$ and not in $\mathcal{C}$.

- Rules $\mathsf{R_{ax}}$, $\mathsf{R_{triv}}$, $\mathsf{R_f}$, $\mathsf{R_{bd}}$ and $\mathsf{R_{get}}$. There are no new membership constraints. Hence, we easily conclude.
- Rule $\mathsf{R_{bdsgn}}$. We have that $\mathcal{C}' = (\mathcal{C} \setminus \{T \overset{?}{\vdash} \mathsf{sign}(v, u)\}) \cup \{T \overset{?}{\vdash} \mathsf{sign}(w, u),\ w \overset{?}{\in} \mathcal{B}d(T, v)\}$ with $\mathsf{sign}(w, u) \in st(T)$. Since $w \in st(T)$, we have that $T_y \subsetneq T$ for every $y \in vars(w)$. This allows us to conclude.
- Rule $\mathsf{R_A}$. We have that $\mathcal{C}' = (\mathcal{C} \setminus \{x \overset{?}{\in} \mathcal{B}d(T', v)\}) \cup \{T \overset{?}{\vdash} v,\ x \overset{?}{\in} \mathcal{B}d(T, v)\}$ with $T \subsetneq T'$. Clearly, we have that $(\mathcal{C}'{\downarrow_S})^{\preceq}_{\overrightarrow{V}} \models (T \overset{?}{\vdash} v)$.
- Rule $\mathsf{R_B}$. We have that $\mathcal{C}' = (\mathcal{C} \setminus \{x \overset{?}{\in} \mathcal{B}d(T, v')\}) \cup \{T \overset{?}{\vdash} w,\ x \overset{?}{\in} \mathcal{B}d(T, w),\ w \overset{?}{\in} \mathcal{B}d(T', v)\}$. We have that $T_y \subsetneq T'$ for every $y \in vars(w)$ and $T \overset{?}{\vdash} w$ is in $\mathcal{C}'$. This allows us to conclude.
- Rule $\mathsf{R_C}$. We have that $\mathcal{C}' = (\mathcal{C} \setminus \{x \overset{?}{\in} \mathcal{B}d(T, v')\}) \cup \{v \in \mathcal{B}d(T, v')\}$. Using $T_x = T$ and $x \overset{?}{\in} \mathcal{B}d(T, v')$ in $\mathcal{C}$ (well-formed), we deduce that $\mathcal{C}^{\preceq}_{\overrightarrow{V'}} \models (T \overset{?}{\vdash} v')$ where $V' = vars(T \cup \{v'\})$. Thus, thanks to Proposition 2, $(\mathcal{C}'{\downarrow_S})^{\preceq}_{\overrightarrow{V'}} \models (T \overset{?}{\vdash} v')$. This allows us to conclude. □

6.3. Completeness

We prove here that, for any solution σ of an unsolved constraint system $\mathcal{C}$, there is a rule such that $\mathcal{C}$ rewrites to a constraint $\mathcal{C}'$ for which σ is a solution. Moreover, the simple proofs in $\mathcal{C}'$ witnessing the solution σ are smaller than the corresponding witness proofs in $\mathcal{C}$.

In the remainder we consider a constraint system $\mathcal{C}$ and we let $T_1 \subseteq T_2 \subseteq \ldots \subseteq T_n$ be the sequence of left members of deducibility constraints. When we consider a simple proof of

$T\sigma \overset{?}{\vdash} u\sigma$ for $T \overset{?}{\vdash} u \in \mathcal{C}$, we refer to the sequence $T_1\sigma \subseteq \ldots \subseteq T_n\sigma$. Given a set of terms T, we denote by $\mathcal{C}(T)$ the elementary constraints in $\mathcal{C}$ that have T as associated set of terms, *i.e.*

$$\mathcal{C}(T) = \{C \in \mathcal{C} \mid C = T \overset{?}{\vdash} u \text{ or } C = v \overset{?}{\in} \mathcal{B}d(T, u) \text{ for some terms } u, v\}$$

Let $\mathcal{C}$ be an unsolved simplified constraint system. We denote by $T_{\min}$ the minimal (w.r.t. inclusion) set of terms such that $\bigcup_{T_i \subseteq T_{\min}} \mathcal{C}(T_i)$ is unsolved. Let $S \subseteq \mathcal{C}(T_{\min})$ be a maximal set of constraints such that $\bigcup_{T_i \subsetneq T_{min}} \mathcal{C}(T_i) \cup S$ is solved. Note that S is not necessarily unique and we consider any set among the possible ones. Then, we define $\mathsf{M}_{\mathcal{C}} = \mathcal{C}(T_{\min}) \setminus S$ to be the minimal unsolved constraints of $\mathcal{C}$.

Example 6. *Consider the constraint system $\mathcal{C}_1$ given in Example 4. This system is unsolved. $\mathcal{C}_1(\{a\}) = \mathcal{C}_1$, and $T_{\min} = \{a\}$. For $\mathsf{M}_{\mathcal{C}}$, there are two possibilities: $\mathsf{M}_{\mathcal{C}} = \{x \overset{?}{\in} \mathcal{B}d(\{a\}, \mathsf{blind}(a, a))\}$ or $\mathsf{M}_{\mathcal{C}} = \{x \overset{?}{\in} \mathcal{B}d(\{a\}, a)\}$.*

To establish completeness, we first lift Lemma 1 to deal with deducibility constraints with variables.

Lemma 9. *Let $\mathcal{C}$ be an unsolved constraint system. Let $\sigma \in Sol(\mathcal{C})$ and t be a term such that $T_{\min}\sigma \vdash t$. Let P be a simple proof of $T_{\min}\sigma \vdash t$. If P is reduced to a leaf or ends with a decomposition rule, then there exists $u \in st(T_{\min}) \setminus \mathcal{X}$ such that $u\sigma = t$.*

Proof. We know that $T_{\min}\sigma \vdash t$. Let i_0 be the minimal indice we need to consider to have that $T_{i_0}\sigma \vdash t$. Since P is simple, P is also a simple proof of $T_{i_0}\sigma \vdash t$. Thanks to Lemma 1, we have that $t \in st(T_{i_0}\sigma)$. Now, we show that $t \in (st(T_{i_0}) \setminus \mathcal{X})\sigma$. Note that $T_{i_0} \subseteq T_{\min}$. We consider two cases:

Case: $i_0 = 1$. In such a case, we have that $T_1\sigma \vdash t$. By definition of a constraint system, we know that T_1 is a set of ground terms. Hence, we have that $t \in (st(T_1) \setminus \mathcal{X})\sigma$ since $t \in st(T_1)$ and $(st(T_1) \setminus \mathcal{X})\sigma = st(T_1)$.

Case $i_0 > 1$*:* In such a case, either $t \in (st(T_{i_0}) \setminus \mathcal{X})\sigma$ and we easily conclude. Otherwise, we have that $t \in st(x\sigma)$ for some $x \in vars(T_{i_0})$. Let us consider such a variable x with $T_x \subsetneq T_{i_0}$ minimal. Then, by definition of $T_{\min}$ we know that there exists $T_x \overset{?}{\vdash} x \in \mathcal{C}$ and thus we have that $T_x\sigma \vdash x\sigma$. By corollary 1, either $t \in st(T_x\sigma)$, or else we have $T_x\sigma \vdash t$. In the latter case, we condradict the minimality of i_0, since $T_x \subsetneq T_{i_0}$. In the former case, we have either $t \in (st(T_x) \setminus \mathcal{X})\sigma$, and we conclude, or else $t \in st(y\sigma)$, for some $y \in vars(T_x)$. By the choice of x, the last case is impossible. Therefore, in all cases we conclude that $t \in (st(T_{i_0}) \setminus \mathcal{X})\sigma$, and thus $t \in (st(T_{\min}) \setminus \mathcal{X})\sigma$. □

Let $\mathcal{C}$ be a constraint system, $\sigma \in Sol(\mathcal{C})$ and $P_1, \ldots, P_k$ be a sequence of simple proofs witnessing the fact that σ is indeed a solution of $\mathcal{C}$. The witness for a deducibility constraint is a single proof, whereas the witness for a membership constraint is a sequence of proofs. Let $\mu(C\sigma)$ be the multiset of pairs (T, n) (one for each P_i) where:

- T is the set of terms that occur in the constraint we consider;

- n is the number of nodes in P_i.

Multisets are ordered according to the multiset extension of the orderings on their elements. Pairs are ordered lexicographically.

We are now able to prove the following propositions. The proofs are given in Appendix D. The tables below show which rule has to be applied depending on the situation.

Proposition 4 (completeness - deducibility constraint). *Let $\mathcal{C}\downarrow_S$ be an unsolved constraint system such that $\mathsf{M}_{\mathcal{C}\downarrow_S}$ contains a deducibility constraint. Let $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}\downarrow_S)$. There exists a constraint system $\mathcal{C}'$ such that $\mathcal{C}\downarrow_S \rightsquigarrow \mathcal{C}'$, $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}')$ and $\mu(\mathcal{C}'\sigma) < \mu(\mathcal{C}\downarrow_S\sigma)$.*

$\mathcal{C}\downarrow_S$ contains among others	$\mathsf{M}_{\mathcal{C}\downarrow_S}$ contains C	Last rule in the proof P witness of C	Rule
$T \overset{?}{\vdash} x$	$T' \overset{?}{\vdash} x$		$\mathsf{R}_{\mathsf{triv}}$
	$T \overset{?}{\vdash} u$	*axiom*	R_{ax}
	$T \overset{?}{\vdash} \mathsf{f}(u_1, u_2)$	*sign*, *blind*	R_{f}
	$T \overset{?}{\vdash} u$	*unblind*	R_{bd}
	$T \overset{?}{\vdash} u$	*getmsg*	$\mathsf{R}_{\mathsf{get}}$
	$T \overset{?}{\vdash} \mathsf{sign}(u_1, u_2)$	*unbdsign*	$\mathsf{R}_{\mathsf{bdsgn}}$

Proposition 5 (completeness - membership constraint). *Let $\mathcal{C}\downarrow_S$ be an unsolved constraint system such that $\mathsf{M}_{\mathcal{C}\downarrow_S}$ only contains membership constraints. Let $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}\downarrow_S)$. There exists a constraint system $\mathcal{C}'$ such that $\mathcal{C}\downarrow_S \rightsquigarrow \mathcal{C}'$, $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}')$ and $\mu(\mathcal{C}'\sigma) < \mu(\mathcal{C}\downarrow_S\sigma)$.*

$T \overset{?}{\vdash} x$	$x \overset{?}{\in} \mathcal{B}d(T', u)$ with $T \subsetneq T'$	T' can be weakened to T in all the side proofs	R_{A}
$T \overset{?}{\vdash} x$	$x \overset{?}{\in} \mathcal{B}d(T', u)$ with $T \subsetneq T'$	T' can be weakened to T in some of the side proofs	R_{B}
$T \overset{?}{\vdash} x$ $x \overset{?}{\in} \mathcal{B}d(T, w)$	$x \overset{?}{\in} \mathcal{B}d(T, u)$		R_{C}

Corollary 4. *Let $\mathcal{C}$ be a pure constraint system in simplified form and $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C})$. There exists a constraint system $\mathcal{C}'$ in solved form such that $\mathcal{C} \rightsquigarrow^* \mathcal{C}'$ (by a derivation of a finite length) and $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}')$.*

Proof. We show this result by induction on $\mu(\mathcal{C}\sigma)$. Note that the ordering $<$ that we use to compare $\mu(\mathcal{C}\sigma)$ and $\mu(\mathcal{C}'\sigma')$ is well-founded, as it is based on multiset extension and lexicographic composition of well-founded orderings.

Base case: $\mu(\mathcal{C}\sigma) = \emptyset$. In such a case, $\mathcal{C} = \emptyset$ and thus in solved form. We easily conclude.

Induction step: $\mu(\mathcal{C}\sigma) \neq \emptyset$. Either $\mathcal{C}$ is in solved form and we easily conclude. Otherwise, we consider its first unsolved constraint and depending on whether this constraint is a deducibility constraint or a membership constraint, we apply Proposition 4 or Proposition 5. We deduce that there exists $\mathcal{C}''$ such that $\mathcal{C} \rightsquigarrow \mathcal{C}''$, $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}'')$ and $\mu(\mathcal{C}''\sigma) < \mu(\mathcal{C}\sigma)$. Thanks to Lemma 3, we deduce that $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}''{\downarrow_S})$, and it is easy to check that our measure μ does not increase when we apply the simplification rules, *i.e.* $\mu(\mathcal{C}''{\downarrow_S}\sigma) \leq \mu(\mathcal{C}''\sigma)$. Altogether, this allows us to conclude. □

7. Decision procedure

Given two substitutions σ and θ, their composition is a substitution denoted by $\sigma \circ \theta$ and defined by $x(\sigma \circ \theta) = (x\theta)\sigma$, for all variable x. Given a set of equations E, we denote by $mgu(E)$ its most general unifier: $mgu(E)$ is a solution of E and for all solution σ of E there is a substitution σ' such that $\sigma = \sigma' \circ mgu(E)$.
Let $\mathcal{C}_0$ be a pure constraint system. Our simplification procedure works as follows:

1. Guess a set of equalities E between subterms of $\mathcal{C}_0$. Solve E: let θ_E be a mgu of E (if there is no solution, then return $\bot$).

2. Apply non-deterministically the transformation rules (Figure 4) and simplification rules (Figure 3) on each $\mathcal{C}\theta_E$ until either a solved form is reached or a loop is detected (*i.e.* $\mathcal{C} \rightsquigarrow^* \mathcal{C}$ with a derivation of length at least one), in which case we return $\bot$.

Considering all possible non-deterministic choices that do not yield $\bot$, the procedure computes $\mathsf{solve}(\mathcal{C}_0)$ a finite set of pairs $(E_i, \mathcal{C}_i)$ such that every $\mathcal{C}_i$ is in solved form.

Theorem 1. *The procedure described above is sound (any solution of an output pair is a solution of the input), complete (any solution of the input is a solution of some output pair) and terminates.*

Thanks to Theorem 1 and Lemma 5, deciding whether a constraint system is satisfiable amounts to decide the existence of a solved form. The proof of Theorem 1 is summarized in sections 7.1 and 7.2.

7.1. Guessing equalities

As a first step, we guess all equalities between subterms of a constraint system $\mathcal{C}$. This might not be the best way if we want to implement the constraint solving procedure, as we would immediately get an exponential branching. However, it simplifies a lot the presentation and the proofs.

Formally, given a pure constraint system $\mathcal{C}_0$, we guess an equivalence relation $\approx$ on $st(\mathcal{C}_0)$. Then we consider the unification problem $\bigwedge_{s \approx t} s = t$. Let θ be a mgu of this set of equations (if any), which does not introduce new variables. Then we consider the pure constraint system $\mathcal{C}_0\theta$. Note that this is indeed a constraint system.

We show that we can restrict ourselves to solutions that do not map two distinct subterms of the constraint system to the same term.

Lemma 10. *Let $\mathcal{C}$ be a constraint system and $\sigma \in Sol(\mathcal{C})$. There exists an equivalence relation $\approx$ on $st(\mathcal{C})$ and $\sigma' \in Sol_{\mathsf{NC}}(\mathcal{C}\theta)$ such that $\sigma = \sigma' \circ \theta$ where $\theta = mgu(\bigwedge_{s \approx t} s = t)$.*

Proof. Let σ be a solution of $\mathcal{C}$. We let $\approx$ be the equivalence relation on $st(\mathcal{C})$ defined by $t \approx u$ if, and only if, $t\sigma = u\sigma$. Let $\sigma_\approx$ be the most general unifier of $\bigwedge_{s\approx t} s = t$. By definition of the mgu, there is a substitution σ' such that $\sigma = \sigma' \circ \sigma_\approx$, which implies that σ' is a solution of $\mathcal{C}\sigma_\approx$. Now, in order to show that $\sigma' \in Sol_{\mathsf{NC}}(\mathcal{C}\sigma_\approx)$, it remains to show that σ' is non-confusing. We use the following observation: if τ is a mgu of u, v which does not introduce any new variable, then, for every variable $x \in vars(u, v)$, $x\tau \in (st(\{u, v\}) \setminus \mathcal{X})\tau$.

Let $t \in st(\mathcal{C}\sigma_\approx)$. We want to show that $t \in st(\mathcal{C})\sigma_\approx$. We have that either $t \in st(\mathcal{C})\sigma_\approx$ or there is a variable x such that $t \in st(x\sigma_\approx)$. In the latter case, let $\{x_1, \ldots, x_n\} \subseteq vars(\mathcal{C})$ such that $t \in st(x_i\sigma_\approx)$. Note that $n \geq 1$. Let i_0 be an indice in $\{1, \ldots, n\}$ such that $x_{i_0}\sigma_\approx$ is minimal w.r.t. the subterm ordering among $\{x_1\sigma_\approx, \ldots, x_n\sigma_\approx\}$. Thanks to our observation, we have in particular that $x_{i_0}\sigma_\approx \in (st(\mathcal{C}) \setminus \mathcal{X})\sigma_\approx$. Hence, there exists $t_{i_0} \in st(\mathcal{C}) \setminus \mathcal{X}$ such that $t_{i_0}\sigma_\approx = x_{i_0}\sigma_\approx$. If $t = x_{i_0}\sigma_\approx$ then we easily conclude. Otherwise, we have that $t \in st(y)\sigma_\approx$ for some variable $y \in st(t_{i_0})$. Hence, $y\sigma_\approx \in st(x_{i_0}\sigma_\approx)$ with $t \in st(y\sigma_\approx)$. This contradicts the choice of the indice i_0.This case is impossible.

So, for any $t, u \in st(\mathcal{C}\sigma_\approx)$, there are $t_0, u_0 \in st(\mathcal{C})$ such that $t = t_0\sigma_\approx$ and $u = u_0\sigma_\approx$. Then, $t\sigma' = u\sigma'$ implies $t_0\sigma = u_0\sigma$, hence $t_0 \approx u_0$ and therefore $t_0\sigma_\approx = u_0\sigma_\approx$, yielding $t = u$. □

7.2. Soundness, completeness, and termination

Termination. We cannot expect termination without any restriction on the application of the rules: this is quite straightforward if we consider rules $\mathsf{R_f}$ and $\mathsf{R_{bd}}$ only. We could avoid non-termination by a careful control on the rules. This would however require a heavy machinery. We prefer to keep a non-terminating system. Indeed, since the set of subterms is bounded by the subterms of the original system, there is a finite number of simplified systems and any non-terminating sequence must include a loop. Then getting a terminating system (yet complete and correct) is easy: simply cut-off looping branches.

Soundness. If $(E, \mathcal{C}_s)$ is a pair returned by our procedure, and σ is a solution of $\mathcal{C}_s$, then there exists θ an *mgu* of E. Let $\mathcal{C}_1 = \mathcal{C}_0\theta$. $\mathcal{C}_1$ is a pure constraint system and thus $\mathcal{C}_1$ is well-formed. $\mathcal{C}_1 \rightsquigarrow^* \mathcal{C}_s$ by a derivation of a finite length.Thanks to Proposition 1, $\sigma \in Sol(\mathcal{C}_1)$ and thus $\sigma \circ \theta \in Sol(\mathcal{C}_0)$.

Completeness. Let $\sigma \in Sol(\mathcal{C}_0)$. By Lemma 10, there exists an equivalence relation $\approx$ on $st(\mathcal{C}_0)$ and $\sigma' \in Sol_{\mathsf{NC}}(\mathcal{C}_0\theta)$ such that $\theta = mgu(E)$ where $E = \bigwedge_{s\approx t} s = t$. We compute such a system $\mathcal{C}_0\theta$ during our first step. Let $\sigma = \sigma' \circ \theta$ and $\mathcal{C}_1 = \mathcal{C}_0\theta$. $\mathcal{C}_1$ is a pure constraint system. Now, we apply our transformation rules and thanks to Corollary 4, there exists $\mathcal{C}_s$ in solved form such that $\mathcal{C}_1 \rightsquigarrow^* \mathcal{C}_s$ and $\sigma' \in Sol_{\mathsf{NC}}(\mathcal{C}_s)$. Then $\sigma = \sigma' \circ \theta$ is a solution of $(E, \mathcal{C}_s)$.

Complexity. Our procedure does not aim to achieve the best computation time, but leaves instead some flexibility in the choice of the simplification strategy. The size of a constraint system obtained after any transformation sequence is bounded by a polynomial in the size of the original system (provided we keep a DAG representation of terms). However, our termination proof only shows an exponential upper bound in the length of a transformation sequence, because there are, a priori, exponentially many constraint systems that are built using the subterms of the original system only. Furthermore, our procedure is non-deterministic in many respects. In summary, we can only derive a non-deterministic exponential upper bound on the complexity, but we conjecture that a careful design of the strategy would yield a NP decision procedure.

7.3. Application to trace properties

Secrecy is a trace property that can be expressed by a constraint system $\mathcal{C}$. However, to handle more general trace properties, for instance agreement [10], we consider a first-order formula ϕ, whose free variables may contain some of the free variables of $\mathcal{C}$. The formula ϕ is interpreted over the same domain as $\mathcal{C}$, that is the set of first-order terms. Then, the conjunction $\mathcal{C} \wedge \neg\phi$ expresses the existence of an attack against the desired property ϕ.

Corollary 5. *The satisfiability of $\mathcal{C} \wedge \neg\phi$ is decidable.*

Indeed, it suffices to solve first $\neg\phi$ (using for instance the algorithm of [9]), getting finitely many formulas of the form:

$$\exists \vec{z}.\ x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge u_1 \neq v_1 \wedge \ldots \wedge u_m \neq v_m$$

Then replace x_i with t_i in $\mathcal{C}$, guess the equalities of $\mathcal{C}$ to obtain a substitution θ and compute the corresponding solved forms of $\mathcal{C}\theta$. We conclude using Lemma 5: there is an attack if, and only if, there is a solved form for which no disequality became a trivial equality $u_i\theta = v_i\theta$.

Trace properties, that are defined in other logics (such as the absence of key cycles or the absence of timing attacks) can also benefit from the solved form computation, as demonstrated in [10].

8. Application to homomorphic encryption

We sketch here another example of security primitive, for which we can compute solved forms in the same way as we did for blind signatures. We consider the Dolev-Yao inference rules for symmetric encryption, however using an ECB encryption mode (or a homomorphic encryption). For such an encryption mode, the attacker may retrieve the encrypted components of a pair from the encryption of the pair itself:

$$\frac{T \vdash \mathsf{enc}(\langle x_1, x_2\rangle, y)}{T \vdash \mathsf{enc}(x_1, y)} \qquad \frac{T \vdash \mathsf{enc}(\langle x_1, x_2\rangle, y)}{T \vdash \mathsf{enc}(x_2, y)}$$

Again, such rules impair the locality property.

$$\dfrac{\dfrac{\mathsf{enc}(\langle\langle u, v\rangle, w\rangle, k)}{\mathsf{enc}(\langle u, v\rangle, k)}}{\mathsf{enc}(v, k)}$$

contains an intermediate step $\mathsf{enc}(\langle u, v\rangle, k)$, which is neither a subterm of the hypotheses, nor a subterm of the conclusion. By saturation, we get however, together with the classical Dolev-Yao inference rules, the infinite local deduction system displayed in Figure 5. We denote by $\mathcal{P}(T, u)$ the least set S of terms that contains u and such that, for every v, $\langle u, v\rangle$ and $\langle v, u\rangle$ are in S when $u \in S$. The rules *enc*, *pair*, and *hom* are called *compositions*, whereas the rules *dec*, $proj_l$, and $proj_r$ are called *decompositions*.

We consider a proof normalization procedure (see Figure 6), that allows one to consider only proofs that have the subformula property. The following lemma and corollary still hold in this case.

Rule *enc*
$$\frac{T \vdash u \quad T \vdash v}{T \vdash \mathsf{enc}(u,v)}$$

Rule *dec*
$$\frac{T \vdash \mathsf{enc}(u,v) \quad T \vdash v}{T \vdash u}$$

Rule $proj_l$
$$\frac{T \vdash \langle u,v\rangle}{T \vdash u}$$

Rule $proj_r$
$$\frac{T \vdash \langle u,v\rangle}{T \vdash v}$$

Rule *pair*
$$\frac{T \vdash u \quad T \vdash v}{T \vdash \langle u,v\rangle}$$

Rule *hom*
$$\frac{T \vdash \mathsf{enc}(u_1,v)}{T \vdash \mathsf{enc}(u_2,v)} \text{ for any } u_1 \in \mathcal{P}(T,u_2)$$

Figure 5: Deduction rules for a homomorphic encryption mode

Lemma 11 (locality). *Let T be a set of terms and v be a term such that $T \vdash v$. Let P be a normal proof of $T \vdash v$. The proof P only contains terms in $st(T \cup \{v\})$. Moreover, if P is reduced to a leaf or ends with a decomposition rule then $v \in st(T)$.*

Corollary 6. *Let T be a set of terms and v be a term such that $T \vdash v$. Let $u \in st(v)$. Either $u \in st(T)$ or there exists a normal proof of $T \vdash u$ that ends with a composition rule.*

The notion of simple proof is defined in the same way and we can show the following lemma.

Lemma 12. *Let $T_1 \subseteq T_2 \subseteq \ldots \subseteq T_n$ be a sequence of sets of terms. If $T_i \vdash u$ for some $i \in \{1,\ldots,n\}$, then there is a simple proof of $T_i \vdash u$.*

We consider two different kinds of *elementary constraints*:

- a *deducibility constraint* is a constraint of the form $T \stackrel{?}{\vdash} u$
- a *membership constraint* is a constraint of the form $v \stackrel{?}{\in} \mathcal{P}(T,u)$.

The definition of constraint system is similar to the one given for blind signatures. It is obtained by replacing $\mathcal{B}d(T,u)$ with $\mathcal{P}(T,u)$. The notion of solutions and non-confusing solutions is adapted to this case.

The membership constraints are simplified according to the simplification rules given in Figure 7. A difference with blind signatures is the fact that the simplified form of a constraint system $\mathcal{C}$ is not unique, so $\mathcal{C}{\downarrow_S}$ denotes a set of constraint systems.

Lemma 13. *Simplification rules described in Figure 7 transform a constraint system into a constraint system,* i.e. *if $\mathcal{C}$ is a simplified constraint system and $\mathcal{C} \rightsquigarrow \mathcal{C}'$ then $\mathcal{C}'{\downarrow_S}$ is a set of constraints system. Moreover, we have that $st(\mathcal{C}'{\downarrow_S}) \subseteq st(\mathcal{C})$.*

The notion of solved forms is similar, however allowing several membership constraints for the same variable:

Definition 9 (solved form). *A constraint system $\mathcal{C} = \{C_1,\ldots,C_\ell\}$ is in* solved form *if: each C_i is either of the form $T_i \stackrel{?}{\vdash} x_i$ or of the form $x_i \stackrel{?}{\in} \mathcal{P}(T_i,u_i)$ where x_i is a variable. Moreover, for every $x \in vars(\mathcal{C})$ there is a unique deducibility constraint $T \stackrel{?}{\vdash} x \in \mathcal{C}$ and every membership constraint $x \stackrel{?}{\in} \mathcal{P}(T',v') \in \mathcal{C}$ is such that $T = T'$.*

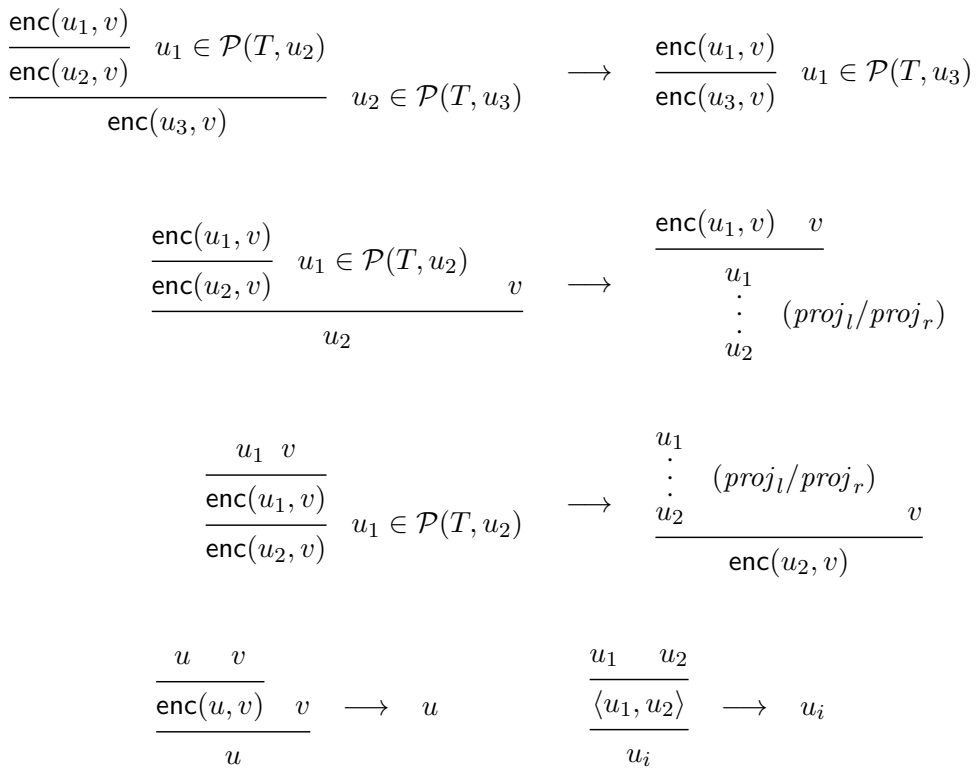

Figure 6: Proof normalization rules for homomorphic encryption

Again, a constraint system in solved form is not necessarily satisfiable and we keep a well-formedness invariant. Let $\mathcal{D}$ be a set of constraints in simplified form. We define $\leq_{\mathcal{D}}$ on $vars(\mathcal{D})$ as the least relation closed by transitivity and reflexivity and such that:

$$x \stackrel{?}{\in} \mathcal{P}(T, u) \ \in \mathcal{D} \text{ and } y \in vars(u) \Longrightarrow y \leq_{\mathcal{D}} x.$$

This defines an ordering, which is extended into a quasi-ordering on sets of variables allowing us to define the notion of well-formed constraint system as for blind signatures. We get:

Lemma 14. *Any solved well-formed simplified constraint system $\mathcal{C}$ has at least one solution. Moreover, if $t_1, \ldots, t_m, u_1, \ldots, u_m$ are sequences of terms such that, for every i, t_i is distinct from u_i, then $\mathcal{C} \wedge t_1 \neq u_1 \wedge \cdots \wedge t_m \neq u_m$ has a solution.*

Then, we use the transformation rules of Figure 8 and get the same sequence of results as for blind signatures.

Completeness. The proof is similar to the case of blind signatures. Note that the witness for a deducibility constraint is a single proof, whereas the witness for a membership constraint will be (by convention) a proof of size 0. To deal with membership constraints, we only have the rule $\mathsf{R_A}$. Indeed, this rule will be sufficient to reach a solved form. The transformation rule $\mathsf{R_C}$ is not needed anymore since we allow several membership constraints of the form $x \stackrel{?}{\in} \mathcal{P}(T, v_1), \ldots, x \stackrel{?}{\in} \mathcal{P}(T, v_n)$ in a solved form. Moreover, $\mathsf{R_B}$ is not useful: it is always possible to apply $\mathsf{R_A}$ since the semantics of $\mathcal{P}(T, v)$ does not depend on its first argument.

$$
\begin{array}{lrcll}
\mathsf{S}_{\mathsf{ax}}: & u \stackrel{?}{\in} \mathcal{P}(T,u) & \to & \top & \\
\mathsf{S}_{\mathsf{left}}: & \langle u,v\rangle \stackrel{?}{\in} \mathcal{P}(T,w) & \to & u \stackrel{?}{\in} \mathcal{P}(T,w) & \text{if } \langle u,v\rangle \neq w \\
\mathsf{S}_{\mathsf{right}}: & \langle u,v\rangle \stackrel{?}{\in} \mathcal{P}(T,w) & \to & v \stackrel{?}{\in} \mathcal{P}(T,w) & \text{if } \langle u,v\rangle \neq w \\
\mathsf{S}_{\mathsf{f}}: & \mathsf{f}(t_1,\ldots,t_n) \stackrel{?}{\in} \mathcal{P}(T,v) & \to & \bot & \text{if } \mathsf{f} \neq \langle\,\rangle \text{ and } \mathsf{f}(t_1,\ldots,t_n) \neq v \\
\mathsf{S}_{\mathsf{cycle}}: & x_1 \stackrel{?}{\in} \mathcal{P}(T_1,v_1[x_2]) \wedge \ldots \wedge x_n \stackrel{?}{\in} \mathcal{P}(T_n,v_n[x_1]) & \to & \bot & \\
 & & & & \text{if there exists } i \text{ such that } v_i \neq \epsilon \text{ or } \#\{x_1,\ldots,x_n\} > 1
\end{array}
$$

Figure 7: Simplification rules for homomorphic encryption

$$
\begin{array}{lll}
\mathsf{R}_{\mathsf{ax}}: & T \stackrel{?}{\vdash} u \rightsquigarrow \top & \text{if } u \in T \setminus \mathcal{X} \\
\mathsf{R}_{\mathsf{triv}}: & T \stackrel{?}{\vdash} x \wedge T' \stackrel{?}{\vdash} x \rightsquigarrow T \stackrel{?}{\vdash} x & \text{if } T \subseteq T' \\
\mathsf{R}_{\mathsf{f}}: & T \stackrel{?}{\vdash} \mathsf{f}(t_1,\ldots,t_n) \rightsquigarrow T \stackrel{?}{\vdash} t_1,\ldots,T \stackrel{?}{\vdash} t_n & \mathsf{f} \in \{\langle\,\rangle, \mathsf{enc}\} \\
\mathsf{R}_{\langle\,\rangle}: & T \stackrel{?}{\vdash} u_i \rightsquigarrow T \stackrel{?}{\vdash} \langle u_1,u_2\rangle & \text{if } \langle u_1,u_2\rangle \in st(T) \\
\mathsf{R}_{\mathsf{dec}}: & T \stackrel{?}{\vdash} v \rightsquigarrow T \stackrel{?}{\vdash} \mathsf{enc}(v,u) \wedge T \stackrel{?}{\vdash} u & \text{if } \mathsf{enc}(v,u) \in st(T) \\
\mathsf{R}_{\mathsf{homenc}}: & T \stackrel{?}{\vdash} \mathsf{enc}(v,u) \rightsquigarrow T \stackrel{?}{\vdash} \mathsf{enc}(w,u) \wedge w \stackrel{?}{\in} \mathcal{P}(T,v) & \text{if } \mathsf{enc}(w,u) \in st(T) \\
\mathsf{R}_{\mathsf{A}}: & T \stackrel{?}{\vdash} x \wedge x \stackrel{?}{\in} \mathcal{P}(T',v) \rightsquigarrow T \stackrel{?}{\vdash} x \wedge T \stackrel{?}{\vdash} v \wedge x \stackrel{?}{\in} \mathcal{P}(T,v) & \text{if } T \subsetneq T'
\end{array}
$$

Figure 8: Transformation rules for homomorphic encryption

Proposition 6 (completeness). *Let $\mathcal{C}{\downarrow_S}$ be an unsolved constraint system. Let $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}{\downarrow_S})$. There exists a constraint system $\mathcal{C}'$ such that $\mathcal{C}{\downarrow_S} \rightsquigarrow \mathcal{C}'$, $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}')$ and $\mu(\mathcal{C}'\sigma) < \mu(\mathcal{C}{\downarrow_S}\sigma)$.*

Theorem 2. *There is a procedure that is terminating, sound, complete, and that transforms any pure constraint system into a finite set of well-formed solved forms.*

Corollary 7. *The satisfiability of $\mathcal{C} \wedge \phi$, where $\mathcal{C}$ is a pure constraint system and ϕ is a first-order formula with equality, is decidable for homomorphic encryption.*

9. Conclusion

We claim that the key property of the inference system, that allows one to solve the deducibility constraints, is locality. Given an inference system, the general procedure then consists in completing the inference rules into a local inference system. When such a system is infinite, we need additional abstractions and constraint solving rules. We have shown in this paper that this is possible, in the case study of blind signatures. We demonstrated that the method is general enough, by giving another example of application. It remains to provide

$\mathcal{C}\downarrow_S$ contains among others	$\mathsf{M}_{\mathcal{C}\downarrow_S}$ contains C	Last rule in the proof P witness of C	Rule
$T \overset{?}{\vdash} x$	$T' \overset{?}{\vdash} x$		$\mathsf{R}_{\mathsf{triv}}$
	$T \overset{?}{\vdash} u$	*axiom*	R_{ax}
	$T \overset{?}{\vdash} \mathsf{f}(u_1, u_2)$	*enc*, *pair*	R_{f}
	$T \overset{?}{\vdash} u$	$proj_l, proj_r$	$\mathsf{R}_{\langle\,\rangle}$
	$T \overset{?}{\vdash} u$	*dec*	$\mathsf{R}_{\mathsf{dec}}$
	$T \overset{?}{\vdash} \mathsf{enc}(u_1, u_2)$	*hom*	$\mathsf{R}_{\mathsf{homenc}}$
$T \overset{?}{\vdash} x$	$x \overset{?}{\in} \mathcal{P}(T', u)$ with $T \subsetneq T'$		R_{A}

Figure 9: Summary: completeness proof

with a general way of abstracting some classes of infinite inference systems, that would be amenable to deducibility constraint solving.

Acknowledgements. This work has been partially supported by the ANR projects PROSE and JCJC VIP n^o 11 JS02 006 01.

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *POPL*, pages 104–115, 2001.

[2] D. Basin and H. Ganzinger. Automated complexity analysis based on ordered resolution. *Journal of the Association of Computing Machinery*, 48(1):70–109, January 2001.

[3] V. Bernat and H. Comon-Lundh. Normal proofs in intruder theories. In *Proceedings of the 11th Asian Computing Science Conference (ASIAN'06)*, volume 4435 of *LNCS*. Springer, 2008.

[4] S. Bursuc, S. Delaune, and H. Comon-Lundh. Deducibility constraints. In *Proceedings of the 13th Asian Computing Science Conference (ASIAN'09)*, volume 5913 of *LNCS*, pages 24–38. Springer, 2009.

[5] D. Chaum. Blind signature system. In D. Chaum, editor, *CRYPTO*, page 153. Plenum Press, New York, 1983.

[6] Y. Chevalier, R. Kuester, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)*. IEEE Computer Society Press, 2003.

[7] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with Diffie-Hellman exponentiation and products in exponents. In *Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'03)*, volume 2914 of *LNCS*. Springer, 2003.

[8] Y. Chevalier and M. Rusinowitch. Symbolic protocol analysis in the union of disjoint intruder theories: Combining decision procedures. *Theor. Comput. Sci.*, 411(10):1261–1282, 2010.

[9] H. Comon and P. Lescanne. Equational Problems and Disunification. *Journal of Symbolic Computation*, 7:371–425, 1989.

[10] H. Comon-Lundh, V. Cortier, and E. Zalinescu. Deciding security properties of cryptographic protocols. Application to key cycles. *Transaction on Computational Logic*, 11(2), 2010.

[11] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in preence of exclusive or. In *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)*. IEEE Computer Society Press, 2003.

[12] S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi calculus. *Journal of Computer Security*, 18(2):317–377, Mar. 2010.

[13] S. Delaune, P. Lafourcade, D. Lugiez, and R. Treinen. Symbolic protocol analysis for monoidal equational theories. *Information and Computation*, 206(2-4):312–351, 2008.

[14] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In J. Seberry and Y. Zheng, editors, *AUSCRYPT*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1992.

[15] S. Kremer and M. D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *Proceedings of the 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005.

[16] D. McAllester. Automatic recognition of tractability in inference relations. *Journal of the ACM*, 40(2), 1993.

[17] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, 2001.

[18] J. Millen and V. Shmatikov. Symbolic protocol analysis with an abelian group operator or Diffie-Hellman exponentiation. *Journal of Computer Security*, 13(3):515–564, 2005.

[19] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is np-complete. In *Proceedings of the 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society Press, 2001.

[20] M. Rusinowitch and M. Turuani. Protocol insecurity with a finite number of sessions, composed keys is NP-complete. *Theoretical Computer Science*, 1-3(299):451–475, 2003.

[21] V. Shmatikov. Decidable analysis of cryptographic protocols with products and modular exponentiation. In *Proceedings of the 13th European Symposium on Programming (ESOP'04)*, volume 2986 of *LNCS*. Springer, 2004.

[22] A. Tiu, R. Goré, and J. E. Dawson. A proof theoretic analysis of intruder theories. *Logical Methods in Computer Science*, 6(3), 2010.

Appendix A. Existence of simple proofs

Lemma 2. *Let $T_1 \subseteq T_2 \subseteq \ldots \subseteq T_n$ be a sequence of sets of terms. If there is a proof of $T_i \vdash u$ for some $i \in \{1, \ldots, n\}$, then there is a simple proof of $T_i \vdash u$ w.r.t. $T_1, \ldots, T_n$.*

Given a proof π of $T_i \vdash u$ that is not necessarily left-minimal, $\mathsf{level}(\pi) = j$ when π is reduced to an axiom and j is the minimal indice such that $u \in T_j$. Otherwise $\mathsf{level}(\pi) = max(\mathsf{level}(\pi_1), \ldots, \mathsf{level}(\pi_n))$ where $\pi_1, \ldots, \pi_n$ are the direct subproofs of π.

Proof. We prove the result by induction on the pair (i, m) (considering the lexicographic ordering) where $i = \mathsf{level}(\pi)$ and m is the size of the proof π.

Case $i = 1$: Let π' be the proof obtained after applying the normalisation rules on π. We have that $\mathsf{level}(\pi') = 1$ and thus π' is left-minimal and in normal form.

Case $i > 1$ and there is $j < i$ such that $T_j \vdash u$: Let π' be a proof of $T_j \vdash u$. In such a case, we apply our induction hypothesis on π' to obtain the existence of a simple proof of $T_j \vdash u$. This proof is also a simple proof of $T_i \vdash u$.

Case $i > 1$ and there is no $j < i$ such that $T_j \vdash u$: In such a case, we apply our induction hypothesis on the immediate subproofs $\pi_1, \ldots, \pi_n$ of the proof π of $T_i \vdash u$. Let $\pi'_1, \ldots, \pi'_n$ be the resulting simple proofs. If π is obtained by applying an inference rule R to $\pi_1, \ldots, \pi_n$ then let π' be the proof obtained by applying R to $\pi'_1, \ldots, \pi'_n$. Note that π' is left-minimal but not necessarily in normal form. However, all its subproofs are in normal form. We distinguish several cases depending on the inference rule R:

Case 1. R is blind or sign: in such a case the proof π' is in normal form and we easily conclude.

Case 2. R is unblind: Either π'_1 ends with an application of the rule *blind*, and after one normalisation step, we obtain a subproof of π'_1 that is a simple proof. Otherwise, the proof π' was already in normal form.

Case 3. R is getmsg: if π' is already in normal form, we conclude. If π'_1 ends with an instance of *sign* we easily conclude since after application of one normalisation rule, we obtain a proof in normal form and that is still left-minimal. Otherwise, if π'_1 ends with an instance of *unbdsign*, we reach a normal form after application of the normalisation rule (*unbdsign /getmsg*). This can be shown by inspecting the normalisation rules. However, the resulting proof is not necessarily left-minimal because of the "new" intermediate nodes. Let $\mathsf{sign}(\mathsf{b}_n(u, v_1, \ldots, v_n), v)$ be the term labeling the root of π'_1. The "new" intermediate nodes are labeled with $\mathsf{b}_k(u, v_1, \ldots, v_k)$ for some $k \in \{1, \ldots, n\}$. We have that $T_i \vdash \mathsf{b}_k(u, v_1, \ldots, v_k)$ for any $1 \leq k \leq n$. Let k be the smallest integer such that $T_j \vdash \mathsf{b}_k(u, v_1, \ldots, v_k)$ for some $j < i$. If such a k does not exist, this means that the proof π' is left-minimal. Otherwise, there exists k and $j < i$ such that $T_j \vdash \mathsf{b}_k(u, v_1, \ldots, v_k)$. By relying on our induction hypothesis, we know that there exists π_0 a simple proof of $T_j \vdash \mathsf{b}_k(u, v_1, \ldots, v_k)$. We have also some simple proofs $\pi^s_1, \ldots, \pi^s_k$ of $T_i \vdash v_1$, ..., $T_i \vdash v_k$. The proof obtained by applying several instances of *unblind* on π_0 with π^s_k, ..., π^s_1 yields to a proof of $T_i \vdash u$ that is simple. Indeed, the proof is left-minimal. The proof is also in normal form since π_0 can not end with an instance of *blind*. Otherwise, this would mean that $T_j \vdash \mathsf{b}_{k-1}(u, v_1, \ldots, v_{k-1})$ and thus k was not minimal.

Case 4. R is unbdsign and $u = \mathsf{sign}(u_0, v_0)$: Either π' is in normal form or π'_1 ends with an instance of R_1 that is either *unbdsign* or *sign*.

- R_1 *is unbdsign.* We can apply the corresponding normalisation rule, we obtained a proof in normal form and we do not introduce "new strict subproofs". Thus all the subproofs are left-minimal and since there is no $j < i$ such that $T_j \vdash u$, we can not "improve the level for the root".

- R_1 *is sign.* Assume that the root of π'_1 is labeled with $\mathsf{sign}(\mathsf{b}_n(u_0, v_1, \ldots, v_n), v_0)$. Let π'' be the proof obtained from π' after normalisation. After application of the normalisation rule (*sign/ unbdsign*), the only rule that is applied to reach such a normal form is (*blind /unblind*). Subproofs that occur in π'' and not in π' are labeled with $\mathsf{b}_p(u_0, v_1, \ldots, v_p)$ for some $p \in \{1, \ldots, n\}$. Moreover, we have that $T_i \vdash \mathsf{b}_k(u_0, v_1, \ldots, v_k)$ for any $1 \leq k \leq n$. Let k be the smallest integer such that $T_j \vdash \mathsf{b}_k(u_0, v_1, \ldots, v_k)$ for some $j < i$. If $T_j \vdash u_0$ for some $j < i$, we will consider, by convention, that k is equal to 0. If such a k does not exist, this means that the proof π'' is left-minimal. Otherwise, there exists k and $j < i$ such that $T_j \vdash \mathsf{b}_k(u_0, v_1, \ldots, v_k)$. By relying on our induction hypothesis, we know that there exists π_0 a simple proof of $T_j \vdash \mathsf{b}_k(u_0, v_1, \ldots, v_k)$. (When $k = 0$, this means that we have a simple proof of $T_j \vdash u_0$). We have also some simple proofs $\pi^s_1, \ldots, \pi^s_k$ of $T_i \vdash v_1$, ..., $T_i \vdash v_k$. The proof obtained by applying several instances of *unblind* on π_0 with π^s_k, ..., π^s_1 yields to a proof of $T_i \vdash u_0$ that is simple. Let π_L be the resulting proof. Indeed, the proof is left-minimal. The proof is also in normal form since π_0 can not end with an instance of *blind*. Otherwise, this would mean that $T_j \vdash \mathsf{b}_{k-1}(u, v_1, \ldots, v_{k-1})$ and thus k was not minimal. By induction hypothesis, we have also a simple proof π_R of $T_i \vdash v_0$. Then, applying the rule *sign* on π_L and π_R, we obtain a simple proof of $T_i \vdash \mathsf{sign}(u_0, v_0)$. Indeed, this proof is left-minimal and in normal form. □

Appendix B. About the simplification rules

Lemma 3. *Let $\mathcal{D}$ and $\mathcal{D}'$ be two sets of elementary constraints such that $\mathcal{D} \to \mathcal{D}'$. We have that:*

- *If $\mathcal{D}$ is a constraint system then $\mathcal{D}'$ is a constraint system;*
- *$Sol(\mathcal{D}') \subseteq Sol(\mathcal{D})$ and $Sol_{\mathsf{NC}}(\mathcal{D}) \subseteq Sol_{\mathsf{NC}}(\mathcal{D}')$.*

Proof. We show the two points separately.

- First, it is clear that monotonicity still holds after the application of a simplification rule. Origination still holds since when we remove a constraint (i.e. S_{ax}), it is clear that this constraint does not introduce any new variable. The transformation performed by applying S_{bd} also preserves origination.

  Now, we have to check that the condition on variables stated in Definition 4 holds. Let $x \in vars(\mathcal{D}')$. We have that $x \in vars(\mathcal{D})$. Let T_x be the set of terms that introduces x in $\mathcal{D}$. If there exists $T_x \overset{?}{\vdash} u \in \mathcal{D}$ with $x \in vars(u)$, then this constraint still exists in $\mathcal{D}'$ and we easily conclude. Otherwise, there exists $v \overset{?}{\in} \mathcal{B}d(T_x, u) \in \mathcal{D}$ with $x \in vars(u)$ and $T_y \subsetneq T_x$ for every $y \in vars(v)$. The only case where we have to pay attention is the case of the rule S_{bd}. However, since $vars(u) \subseteq vars(\mathsf{blind}(u, v))$, we easily conclude.

- We consider each simplification rule in turn. In case of $\mathsf{S_{ax}}$ both inclusions follow immediately from the semantics of $\overset{?}{\in}$. Now, for the other simplication rules, it is easy to see that $Sol(\mathcal{D}') \subseteq Sol(\mathcal{D})$. For the other inclusion $Sol_{\mathsf{NC}}(\mathcal{D}) \subseteq Sol_{\mathsf{NC}}(\mathcal{D}')$, we have to rely on the fact that the solution σ we consider is non-confusing. In case of $\mathsf{S_{bd}}$ (the case of the rule $\mathsf{S_f}$ is similar), since $\mathsf{blind}(u, v) \neq w$, we have also that $\mathsf{blind}(u\sigma, v\sigma) \neq w\sigma$. Since $\mathsf{blind}(u\sigma, v\sigma) \in \mathcal{B}d(T\sigma, w\sigma)$, we necessarily have that $u\sigma \in \mathcal{B}d(T\sigma, w\sigma)$ and $T\sigma \vdash v\sigma$. This allows us to conclude that $\sigma \in Sol(\mathcal{D}')$. Since the simplification rules do not introduce subterms, we deduce that $\sigma \in Sol_{\mathsf{NC}}(\mathcal{D}')$. In case of $\mathsf{S_{cycle}}$, by relying on the fact that σ is non-confusing, we will obtain a contradiction. Thus we will deduce that $Sol_{\mathsf{NC}}(\mathcal{D}) = \emptyset$. □

Appendix C. Well-formedness

Lemma 8 (property of $\leq_{\mathcal{D}}$). *Let $\mathcal{D}$ and $\mathcal{D}'$ be two sets of constraints in simplified form such that $\mathcal{D} \rightsquigarrow \mathcal{D}'$ and $\mathcal{D}' \neq \bot$. We have that $\leq_{\mathcal{D}} \subseteq \leq_{\mathcal{D}'}$.*

Proof. We consider each transformation rule in turn. The cases of the rules $\mathsf{R_{ax}}$, $\mathsf{R_{triv}}$, $\mathsf{R_f}$, $\mathsf{R_{bd}}$ and $\mathsf{R_{get}}$ are easy since they do not affect membership constraints. The rule $\mathsf{R_{bdsgn}}$ introduces a new membership constraint in $\mathcal{D}'$ and thus we easily conclude that $\leq_{\mathcal{D}} \subseteq \leq_{\mathcal{D}'}$. There remain the following cases:

- $\mathsf{R_A}$: $\mathcal{D}_0 \wedge T \overset{?}{\vdash} x \wedge x \overset{?}{\in} \mathcal{B}d(T', v) \rightsquigarrow \mathcal{D}_0 \wedge T \overset{?}{\vdash} x \wedge T \overset{?}{\vdash} v \wedge x \overset{?}{\in} \mathcal{B}d(T, v)$ with $T \subsetneq T'$. In this case, the ordering is not affected. We have that $\leq_{\mathcal{D}} = \leq_{\mathcal{D}'}$.
- $\mathsf{R_B}$: $\mathcal{D}_0 \wedge T \overset{?}{\vdash} x \wedge x \overset{?}{\in} \mathcal{B}d(T', v) \rightsquigarrow \mathcal{D}_0 \wedge T \overset{?}{\vdash} x \wedge T \overset{?}{\vdash} w \wedge x \overset{?}{\in} \mathcal{B}d(T, w) \wedge w \overset{?}{\in} \mathcal{B}d(T', v)$ with $T \subsetneq T'$ and $w \in st(T)$. We have to show that $y \leq_{\mathcal{D}'} x$ for every $y \in vars(v)$. The simplification rules applied to $w \overset{?}{\in} \mathcal{B}d(T', v)$ lead to a membership constraint of the form $z \overset{?}{\in} \mathcal{B}d(T', v)$ with $z \in vars(w)$ or disappears only if $v \in st(w)$. In both cases, we easily conclude.
- $\mathsf{R_C}$: $\mathcal{D}_0 \wedge T \overset{?}{\vdash} x \wedge x \overset{?}{\in} \mathcal{B}d(T, v) \wedge x \overset{?}{\in} \mathcal{B}d(T, v') \rightsquigarrow \mathcal{D}_0 \wedge T \overset{?}{\vdash} x \wedge x \overset{?}{\in} \mathcal{B}d(T, v) \wedge v \overset{?}{\in} \mathcal{B}d(T, v')$ with $T_x = T$. We have to show that $y \leq_{\mathcal{D}'} x$ for every $y \in vars(v')$. The simplification rules applied to $v \overset{?}{\in} \mathcal{B}d(T, v')$ leads to a membership constraint of the form $z \overset{?}{\in} \mathcal{B}d(T, v')$ with $z \in vars(v)$ or disappears only if $v' \in st(v)$. Again, in both cases, we easily conclude. □

Proposition 2. *Let $\mathcal{D}$ and $\mathcal{D}'$ be two sets of constraints in simplified form such that $\mathcal{D} \rightsquigarrow \mathcal{D}'$, and $V \subseteq vars(\mathcal{D})$. We have that $\mathcal{D}'^{\prec}_{V} \models \mathcal{D}^{\prec}_{V}$.*

Proof. Let $\mathcal{D}''$ be the set of constraints obtained after application of the transformation rule on $\mathcal{D}$. We have that $\mathcal{D} \rightsquigarrow \mathcal{D}''$ and $\mathcal{D}''{\downarrow_S} = \mathcal{D}'$. First, by considering each transformation rule in turn, we show that:

$$\mathcal{E}'' \overset{\mathsf{def}}{=} \{C \mid C \in \mathcal{D}'' \wedge vars(C) \preceq_{\mathcal{D}'} V\} \models \{C \mid C \in \mathcal{D} \wedge vars(C) \preceq_{\mathcal{D}} V\}\ (= \mathcal{D}^{\preceq}_{V}).$$

Thanks to Lemma 8, we know that $\leq_{\mathcal{D}} \subseteq \leq_{\mathcal{D}'}$ and thus $\preceq_{\mathcal{D}} \subseteq \preceq_{\mathcal{D}'}$.

- Rule $\mathsf{R_{ax}}$: $\mathcal{D}_0 \wedge T \overset{?}{\vdash} u \;\rightsquigarrow\; \mathcal{D}_0$. Let $C = T \overset{?}{\vdash} u$. If $C \in \mathcal{D}_V^{\preceq}$ then $\mathcal{D}_V^{\preceq} \rightsquigarrow \mathcal{E}' \subseteq \mathcal{E}''$. Otherwise, if $C \notin \mathcal{D}_V^{\preceq}$, we have that $\mathcal{D}_V^{\preceq} \subseteq \mathcal{E}''$. In both cases, we have that $\mathcal{E}'' \models \mathcal{D}_V^{\preceq}$ (Lemma 7).

- Rule $\mathsf{R_{triv}}$: $\mathcal{D}_0 \wedge T \overset{?}{\vdash} x \wedge T' \overset{?}{\vdash} x \;\rightsquigarrow\; \mathcal{D}_0 \wedge T \overset{?}{\vdash} x$. Let $C = T' \overset{?}{\vdash} x$. If $C \in \mathcal{D}_V^{\preceq}$ then $T \overset{?}{\vdash} x$ is also in $\mathcal{D}_V^{\preceq}$ and we have that $\mathcal{D}_V^{\preceq} \rightsquigarrow \mathcal{E}' \subseteq \mathcal{E}''$. Otherwise, if $C \notin \mathcal{D}_V^{\preceq}$, we have that $\mathcal{D}_V^{\preceq} \subseteq \mathcal{E}''$.

- Rule $\mathsf{R_f}$: $\mathcal{D}_0 \wedge T \overset{?}{\vdash} \mathsf{f}(t_1, \ldots, t_n) \;\rightsquigarrow\; \mathcal{D}_0 \wedge T \overset{?}{\vdash} t_1 \wedge \ldots \wedge T \overset{?}{\vdash} t_n$. Let $C = T \overset{?}{\vdash} \mathsf{f}(t_1, \ldots, t_n)$ and $C_i = T \overset{?}{\vdash} t_i$ for $1 \leq i \leq n$. If $C \in \mathcal{D}_V^{\preceq}$ then $C_i \in \mathcal{E}''$ for each $1 \leq i \leq n$, and thus $\mathcal{D}_V^{\preceq} \rightsquigarrow \mathcal{E}' \subseteq \mathcal{E}''$. Otherwise, if $C \notin \mathcal{D}_V^{\preceq}$, we have that $\mathcal{D}_V^{\preceq} \subseteq \mathcal{E}''$. In both cases, we have that $\mathcal{E}'' \models \mathcal{D}_V^{\preceq}$ (Lemma 7).

- Rules $\mathsf{R_{bd}}$ and $\mathsf{R_{get}}$ can be done similarly.

- Rule $\mathsf{R_{bdsgn}}$: $\mathcal{D}_0 \wedge T \overset{?}{\vdash} \mathsf{sign}(v, u) \;\rightsquigarrow\; \mathcal{D}_0 \wedge T \overset{?}{\vdash} \mathsf{sign}(w, u) \wedge w \overset{?}{\in} \mathcal{B}d(T, v)$ with $\mathsf{sign}(w, u) \in st(T)$. Let $C = T \overset{?}{\vdash} \mathsf{sign}(v, u)$. If $C \in \mathcal{D}_V^{\preceq}$ then $T \overset{?}{\vdash} \mathsf{sign}(w, u)$ and $w \overset{?}{\in} \mathcal{B}d(T, v)$ are in $\mathcal{E}''$ and thus $\mathcal{D}_V^{\preceq} \rightsquigarrow \mathcal{E}' \subseteq \mathcal{E}''$. Otherwise, if $C \notin \mathcal{D}_V^{\preceq}$, we have that $\mathcal{D}_V^{\preceq} \subseteq \mathcal{E}''$. In both cases, we have that $\mathcal{E}'' \models \mathcal{D}_V^{\preceq}$.

- Rule $\mathsf{R_A}$: $\mathcal{D}_0 \wedge T \overset{?}{\vdash} x \wedge x \overset{?}{\in} \mathcal{B}d(T', v) \;\rightsquigarrow\; \mathcal{D}_0 \wedge T \overset{?}{\vdash} x \wedge T \overset{?}{\vdash} v \wedge x \overset{?}{\in} \mathcal{B}d(T, v)$ with $T \subsetneq T'$. Either $vars(T' \cup \{v, x\}) \prec_{\mathcal{D}} V$ and then $vars(T' \cup \{v, x\}) \prec_{\mathcal{D}'} V$ (thanks to Lemma 8) and thus $T \overset{?}{\vdash} x$, $T \overset{?}{\vdash} v$ and $x \overset{?}{\in} \mathcal{B}d(T, v)$ are in $\mathcal{E}''$. We have that $\mathcal{D}_V^{\preceq} \rightsquigarrow \mathcal{E}' \subseteq \mathcal{E}''$. Otherwise, we have that $\mathcal{D}_V^{\preceq} \subseteq \mathcal{E}''$. In both cases, we have that $\mathcal{E}'' \models \mathcal{D}_V^{\preceq}$.

- Rule $\mathsf{R_B}$: $\mathcal{D}_0 \wedge T \overset{?}{\vdash} x \wedge x \overset{?}{\in} \mathcal{B}d(T', v) \;\rightsquigarrow\; \mathcal{D}_0 \wedge T \overset{?}{\vdash} x \wedge T \overset{?}{\vdash} w \wedge x \overset{?}{\in} \mathcal{B}d(T, w) \wedge w \overset{?}{\in} \mathcal{B}d(T', v)$ with $T \subsetneq T'$ and $w \in st(T)$. Either $vars(T' \cup \{v, x\}) \prec_{\mathcal{D}} V$ and then $vars(T' \cup \{v, x\}) \prec_{\mathcal{D}'} V$ (thanks to Lemma 8) and thus $T \overset{?}{\vdash} x$, $T \overset{?}{\vdash} w$, $x \overset{?}{\in} \mathcal{B}d(T, w)$ and $w \overset{?}{\in} \mathcal{B}d(T', v)$ are in $\mathcal{E}''$. We have that $\mathcal{D}_V^{\preceq} \rightsquigarrow \mathcal{E}' \subseteq \mathcal{E}''$. Otherwise, we have that $\mathcal{D}_V^{\preceq} \subseteq \mathcal{E}''$ In both cases, we have that $\mathcal{E}'' \models \mathcal{D}_V^{\preceq}$.

- Rule $\mathsf{R_C}$: $\mathcal{D}_0 \wedge T \overset{?}{\vdash} x \wedge x \overset{?}{\in} \mathcal{B}d(T, v) \wedge x \overset{?}{\in} \mathcal{B}d(T, v') \;\rightsquigarrow\; \mathcal{D}_0 \wedge T \overset{?}{\vdash} x \wedge x \overset{?}{\in} \mathcal{B}d(T, v) \wedge v \overset{?}{\in} \mathcal{B}d(T, v')$ with $T_x = T$. First note that $vars(\{v, v'\}) \preceq_{\mathcal{D}} x$. Therefore we have that the three constraints $x \overset{?}{\in} \mathcal{B}d(T, v)$, $x \overset{?}{\in} \mathcal{B}d(T, v')$, and $T \overset{?}{\vdash} x$ are either all in $\mathcal{D}_V^{\preceq}$ or none of them are in $\mathcal{D}_V^{\preceq}$. In the first case, we have that $\mathcal{D}_V^{\preceq} \rightsquigarrow \mathcal{E}' \subseteq \mathcal{E}''$. Otherwise we have that $\mathcal{D}_V^{\preceq} \subseteq \mathcal{E}''$. In both cases, we have that $\mathcal{E}'' \models \mathcal{D}_V^{\preceq}$.

Let $V' = \{x \mid x \preceq_{\mathcal{D}'} y \text{ for some } y \in V\}$. Thanks to Corollary 3 and since $\mathcal{D}'' \rightarrow^* \mathcal{D}'$, we have that $\mathcal{D}'|_{V'} \models \mathcal{D}''|_{V'}$ $(= \mathcal{E}'')$. This allows us to conclude that $\mathcal{D}'^{\preceq}_V \models \mathcal{D}_V^{\preceq}$. $\square$

Appendix D. Completeness

Proposition 4 (completeness - deducibility constraint). *Let $\mathcal{C}\downarrow_S$ be an unsolved constraint system such that $\mathsf{M}_{\mathcal{C}\downarrow_S}$ contains a deducibility constraint. Let $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}\downarrow_S)$. There exists a constraint system $\mathcal{C}'$ such that $\mathcal{C}\downarrow_S \rightsquigarrow \mathcal{C}'$, $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}')$ and $\mu(\mathcal{C}'\sigma) < \mu(\mathcal{C}\downarrow_S\sigma)$.*

Proof. Let $T \vdash u$ be a deducibility constraint in $\mathsf{M}_{\mathcal{C}\downarrow_S}$. Let P be a simple proof of $T\sigma \vdash u\sigma$. We distinguish several cases:

- *u is a variable.* In such a situation the rule $\mathsf{R}_{\mathsf{triv}}$ could be applied. We have that $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}')$ and $\mu(\mathcal{C}'\sigma) < \mu(\mathcal{C}\downarrow_S\sigma)$.
- *P is a reduced to a leaf or ends with a decomposition rule.* If P is reduced to a leaf then $u\sigma \in T\sigma$ and thus $u \in T$ since σ is non-confusing. Hence, we can apply R_{ax}. Clearly, we have that $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}')$ and $\mu(\mathcal{C}'\sigma) < \mu(\mathcal{C}\downarrow_S\sigma)$.

  Now, if P ends with an instance of the rule *unblind*, we have that there exists w such that the direct subproof P_1 (resp. P_2) of P is labeled with $T \vdash \mathsf{blind}(u\sigma, w)$ (resp. $T \vdash w$). We have that P is a simple normal proof, thus P_1 can not end with a composition rule. Thanks to Lemma 9, we deduce that $\mathsf{blind}(u\sigma, w) \in (st(T) \setminus \mathcal{X})\sigma$. Hence, by relying on the fact that σ is non-confusing, there exists $\mathsf{blind}(u, w') \in st(T)$ such that $w'\sigma = w$. Hence, we can apply R_{bd}. Let $\mathcal{C}'$ be the constraint system obtained after application of the transformation rule. Note that $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}')$ and $\mu(\mathcal{C}\downarrow_S\sigma) > \mu(\mathcal{C}'\sigma)$, since we have removed one inference rule in a proof tree witnessing the fact that σ is a solution of $\mathcal{C}'$. The case where P ends with an instance of *getmsg* is similar.
- *P ends with an instance of a composition rule.* If P ends with an instance of *sign* (resp. *blind*), we have that $u = \mathsf{sign}(u_1, u_2)$ (resp. $u = \mathsf{blind}(u_1, u_2)$). In such a case, we can apply the transformation rule R_{f}, and we easily conclude. Now, it remains to deal with the case where P ends with an instance of *unbdsign*. In such a case, we have that $u = \mathsf{sign}(u_1, u_2)$. Let $P_0, P_1, \ldots, P_k$ be the direct subtrees of P. We have that P_0 is labeled with $T\sigma \vdash \mathsf{sign}(w, u_2\sigma)$. Moreover, note that P_0 is normal and therefore does not end with a composition rule. Hence, thanks to Lemma 9, we have that $\mathsf{sign}(w, u_2\sigma) \in (st(T) \setminus \mathcal{X})\sigma$, *i.e.*, by using the fact that σ is non-confusing, there exists $\mathsf{sign}(w', u_2) \in st(T)$ such that $w'\sigma = w$. We deduce that the rule $\mathsf{R}_{\mathsf{bdsgn}}$ can be applied. Let $\mathcal{C}'$ be the resulting constraint system. We have that $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}')$ and $\mu(\mathcal{C}'\sigma) < \mu(\mathcal{C}\downarrow_S\sigma)$. □

Proposition 5 (completeness - membership constraint). *Let $\mathcal{C}\downarrow_S$ be an unsolved constraint system such that $\mathsf{M}_{\mathcal{C}\downarrow_S}$ only contains membership constraints. Let $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}\downarrow_S)$. There exists a constraint system $\mathcal{C}'$ such that $\mathcal{C}\downarrow_S \rightsquigarrow \mathcal{C}'$, $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}')$ and $\mu(\mathcal{C}'\sigma) < \mu(\mathcal{C}\downarrow_S\sigma)$.*

Proof. Let us note that, since $\mathcal{C}\downarrow_S$ is simplified, all the membership constraints are of the form $x \overset{?}{\in} \mathcal{B}d(T, u)$. Let us consider the set of variables $V_M = \{x \mid x \overset{?}{\in} \mathcal{B}d(T', u) \in \mathsf{M}_{\mathcal{C}\downarrow_S}\}$. Let us choose a constraint $x \overset{?}{\in} \mathcal{B}d(T', u) \in \mathsf{M}_{\mathcal{C}\downarrow_S}$ such that x is maximal in V_M with respect to $\leq_{\mathcal{C}\downarrow_S}$. Note that, thanks to origination, definition of solved forms and the maximality of

x, we know that there exists a deducibility constraint $T \stackrel{?}{\vdash} x$ that occurs in $\mathcal{C}{\downarrow_S}$ with $T \subseteq T'$. We distinguish several cases depending on the fact that $T \subsetneq T'$ or $T = T'$.

- Case $T \subsetneq T'$. In such a case, we have that $T \stackrel{?}{\vdash} x$ and $x \stackrel{?}{\in} \mathcal{B}d(T', u)$ are in $\mathcal{C}{\downarrow_S}$ and $T \subsetneq T'$. We will show that we can apply $\mathsf{R_A}$ or $\mathsf{R_B}$. We have that $x\sigma = \mathsf{b}_k(u\sigma, v_1, \ldots, v_k)$ with $T'\sigma \vdash v_1$, $\ldots$, $T'\sigma \vdash v_k$. Note that $k > 0$ since σ is non-confusing. Depending on whether all proofs of $T'\sigma \vdash v_i$ can be weakened or not, we apply either $\mathsf{R_A}$ or $\mathsf{R_B}$.

  - Assume that $T\sigma \vdash v_i$ for every $1 \leq i \leq k$. Let $\mathcal{C}'$ be the constraint system obtained by applying the rule $\mathsf{R_A}$. We have that $T\sigma \vdash x\sigma$, thus we deduce that $T\sigma \vdash u\sigma$. A proof tree witnessing this fact can be obtained by applying k times the rule *unblind* on the proof of $T \vdash x\sigma$ and by using the proof tree witnessing $T\sigma \vdash v_i$ for every $1 \leq i \leq k$. Hence, we have that $\sigma \in Sol(\mathcal{C}')$, and since no subterm is introduced, we have that $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}')$. Lastly, we have that $\mu(\mathcal{C}'\sigma) < \mu(\mathcal{C}{\downarrow_S}\sigma)$ since we replace at least one pair $(T'\sigma, n)$ (the one corresponding to the proof of $T'\sigma \vdash v_1$) by a set of pairs whose first component is $T\sigma \subsetneq T'\sigma$ (this strict inclusion is due to the fact that $T \subsetneq T'$ and σ non-confusing).
  - Otherwise, let i_0 be such that $T\sigma \vdash v_j$ for each $j > i_0$ and $T\sigma \nvdash v_{i_0}$. Note that such a i_0 exists since otherwise we fall into the first case. Note also that if $i_0 = k$, we have that any proof of $T\sigma \vdash x\sigma$ ends with an axiom or a decomposition rule. Thus, by Lemma 9, we deduce that there exists $t \in st(T) \setminus \mathcal{X}$ such that $t\sigma = x\sigma$. Since σ is non-confusing, this case is impossible. Thus we have that $i_0 \neq k$. Now, let us consider the case where $1 \leq i_0 < k$. We have that $T\sigma \vdash \mathsf{blind}(\ldots \mathsf{blind}(u\sigma, v_1), \ldots, v_{i_0})$ by taking the proof of $T\sigma \vdash x\sigma$, the proofs of $T\sigma \vdash v_k$, $\ldots$, $T\sigma \vdash v_{i_0+1}$ and applying $k - i_0$ times the rule *unblind*. Let P be a simple proof of $T\sigma \vdash \mathsf{b}_{i_0}(u\sigma, v_1, \ldots, v_{i_0})$. Now, since $T\sigma \nvdash v_{i_0}$, P can not end with a composition rule. Hence, by Lemma 9, we deduce that there exists $w \in (st(T) \setminus \mathcal{X})$ such that $w\sigma = \mathsf{b}_{i_0}(u\sigma, v_1, \ldots, v_{i_0})$, implying that $w\sigma \in \mathcal{B}d(T'\sigma, u\sigma)$. Therefore, we can apply the rule $\mathsf{R_B}$ to get the constraint system $\mathcal{C}'$. We have seen that σ satisfies each constraint that has been added in $\mathcal{C}'$, hence $\sigma \in Sol(\mathcal{C}')$ and it is easy to see that actually $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}')$. Lastly, we have that $\mu(\mathcal{C}'\sigma) < \mu(\mathcal{C}{\downarrow_S}\sigma)$ since we replace at least one pair $(T'\sigma, n)$ (the one corresponding to the proof of $T'\sigma \vdash v_k$) by several pairs whose first component is $T\sigma \subsetneq T'\sigma$ (this strict inclusion is due to the fact that $T \subsetneq T'$ and σ non-confusing).

- Case $T = T'$. In such a case, since $x \stackrel{?}{\in} \mathcal{B}d(T', u) \in \mathsf{M}_{\mathcal{C}{\downarrow_S}}$, we must have that there exists another membership constraint $x \stackrel{?}{\in} \mathcal{B}d(T'', w)$. Note that, by definition of solved form, we have $T'' = T' = T$. Moreover, we have that $T_x = T$. In such a case, we have that

  - $x\sigma = \mathsf{b}_k(u\sigma, v_1, \ldots, v_k)$ with $T\sigma \vdash v_i$ for $1 \leq i \leq k$; and
  - $x\sigma = \mathsf{b}_p(w\sigma, v'_1, \ldots, v'_p)$ with $T\sigma \vdash v'_i$ for $1 \leq i \leq p$.

  Now, clearly, we have that either $u\sigma \in st(w\sigma)$ and $w\sigma \in \mathcal{B}d(T\sigma, u\sigma)$ or symmetrically $w\sigma \in st(u\sigma)$ and $u\sigma \in \mathcal{B}d(T\sigma, w\sigma)$. We will assume w.l.o.g. that we are in the first case. The other one can be done in a similar way. Hence, we can apply $\mathsf{R_C}$ to get a constraint system $\mathcal{C}' = \mathcal{C} \setminus \{x \stackrel{?}{\in} \mathcal{B}d(T, u)\} \cup \{w \stackrel{?}{\in} \mathcal{B}d(T, u)\}$. It is clear that $\sigma \in Sol_{\mathsf{NC}}(\mathcal{C}')$. Now,

we have to show that the sequence of proof trees witnessing $w\sigma \in \mathcal{B}d(T\sigma, u\sigma)$ is strictly smaller than the sequence witnessing $x\sigma \in \mathcal{B}d(T\sigma, u\sigma)$. This is due to the fact that $w\sigma$ is a strict subterm of $x\sigma$. Indeed, we have that $w\sigma \in st(x\sigma)$ and $w\sigma = x\sigma$ is not possible since otherwise, we would have $w = x$ and this would contradict the fact that $\mathcal{C}{\downarrow_S}$ is simplified. □