Deducibility constraints and blind signatures

S. Bursuc, H. Comon-Lundh, and S. Delaune

November 2010

Research report LSV-10-24
DEDUCIBILITY CONSTRAINTS AND BLIND SIGNATURES

SERGIU BURSUC, HUBERT COMON-LUNDH, AND STÉPHANIE DELAUNE

School of Computer Science, University of Birmingham, UK
e-mail address: s.bursuc@cs.bham.ac.uk

LSV, ENS Cachan & CNRS & INRIA Saclay Île-de-France, France
e-mail address: comon@lsv.ens-cachan.fr

LSV, ENS Cachan & CNRS & INRIA Saclay Île-de-France, France
e-mail address: delaune@lsv.ens-cachan.fr

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.

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 [14, 15]: 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 [12]. 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 $T \vdash u$ 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 [8, 4], modular exponentiation [5, 16, 13], any monoidal theory [9] and blind signatures [2]. 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 $\phi$ holds, can be checked by deciding the satisfiability of $\neg \phi$, together with each solved constraint. This is typically what is proposed in [7], 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 [3], 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 [11]: if there is a proof of $\Gamma$, 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 [11]. As shown in [1], 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 [3], 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 [10]: 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 $T \vdash u$, guess the last inference rule that yields a proof of $u o$ 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 $u o$, 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.
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, $\mathcal{F} = \{\text{blind}, \text{sign}, \text{vk}\}$ together with arities $ar(\text{blind}) = ar(\text{sign}) = 2$ and $ar(\text{vk}) = 1$. The term $\text{sign}(m, sk)$ represents the message $m$ signed by the private key $sk$. The function $\text{blind}$ is supposed to hide a message, thus the term $\text{blind}(m, r)$ 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 $\text{vars}(t)$ for the set of variables occurring in $t$ and $\text{st}(t)$ 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 $\text{dom}(\sigma) = \{x_1, \ldots, x_n\}$. The application of a substitution $\sigma$ 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 $\text{sign}$) and blinding (rule $\text{blind}$) 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 $\text{sign}(m, sk)$ and a message $m$ match (provided he has the verification key $\text{vk}(sk)$), 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 $\text{getmsg}$ that expresses that an intruder can retrieve the whole message from its signature. The rule $\text{unblind}$ allows one to retrieve the message $m$ from $\text{blind}(m, r)$ provided one knows the term $r$ that has been used to hide $m$. Finally, the rule $\text{unbdsign}$ 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. Actually, this last rule alone yields a non-local proof system, because it is not saturated with respect to the subterm ordering. Consider the following proof:

\[
\frac{\text{sign}(\text{blind}(\text{blind}(u, r_1), r_2), sk) \quad r_2}{\text{sign}(\text{blind}(u, r_1), sk) \quad r_1}
\]

The intermediate step $\text{sign}(\text{blind}(u, r_1), sk)$ is neither a subterm of the hypothesis nor a subterm of the conclusion. We can restore the locality property by adding new inference rules that are obtained by saturation. The saturation process yields the infinite (recursive) set of rules that are displayed in Figure 1 and which, roughly, fold the non-local typical proofs into a single inference step.

The rules $\text{blind}$, $\text{sign}$, and $\text{unbdsign}$ are called composition rules whereas $\text{getmsg}$ and $\text{unblind}$ are called decomposition rules.

**Definition 2.1 (deducible).** A proof $P$ of $T \vdash u$ is a tree labeled with terms, whose leaves labels $\text{Hyp}(P)$ 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 $T \vdash u$. 
2.3. Simple proofs. We first consider a proof normalization procedure, that allows one to consider only proofs that have the subformula property. The normalization rules described in Figure 2 are strongly normalizing.

**Definition 2.2** (normal proof). A normal proof is a proof irreducible w.r.t. the rules given in Figure 2.

**Figure 2:** Proof normalization rules for blind signatures

<table>
<thead>
<tr>
<th>Rule</th>
<th>Premise</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>blind</strong></td>
<td>$x \quad y$</td>
<td>blind$(x, y)$</td>
</tr>
<tr>
<td><strong>unblind</strong></td>
<td>blind$(x, y) \quad y$</td>
<td>$x$</td>
</tr>
<tr>
<td><strong>sign</strong></td>
<td>$x \quad y$</td>
<td>sign$(x, y)$</td>
</tr>
</tbody>
</table>

Rule *getmsg*

- Sign$(x, y) \rightarrow x$

Rule *unbdsign*$_n$

- $\text{sign}(\text{blind}(x, y), z) \quad y_1 \ldots y_n \quad n \geq 1$

where $b_1(u, v_1) = \text{blind}(u, v_1)$ and $b_{n+1}(u, v_1, \ldots, v_{n+1}) = \text{blind}(b_n(u, v_1, \ldots, v_n), v_{n+1})$.

**Figure 1:** Intruder deduction system for blind signatures
Example 2.3. Let $T = \{ a, r_1, r_2, \text{sign}(\text{blind}(\text{blind}(a, r_1), r_2), sk) \}$. 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.

\[
\frac{\text{sign}(\text{blind}(\text{blind}(a, r_1), r_2), sk)}{r_1 \ r_2} \quad \frac{\text{sign}(\text{blind}(\text{blind}(a, r_1), r_2), sk)}{\text{blind}(\text{blind}(a, r_1), r_2)} \quad \frac{\text{blind}(\text{blind}(a, r_1), r_2)}{r_2} \quad \frac{\text{blind}(a, r_1)}{r_1} \quad \frac{\text{sign}(a, sk)}{a}
\]

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 2.4 (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.
- $P$ 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 $T \vdash \text{sign}(b_k(u, v_1, \ldots, v_k), v_0)$ ends with a decomposition rule. Thanks to our induction hypothesis, $\text{sign}(b_k(u, v_1, \ldots, v_k), v_0) \in st(T)$. 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 $T \vdash \text{blind}(v, u)$ 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 $\text{blind}(v, u) \in st(T)$ 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 $T \vdash \text{sign}(v, u)$ 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 $\text{sign}(v, u) \in st(T)$ and thus $v \in st(T)$.

From Lemma 2.4, we derive the following corollary. It will be useful later on to prove completeness of our decision procedure (see Section 5.3).

Corollary 2.5. 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 composition rule. In such a case, thanks to Lemma 2.4, 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 $v = \text{blind}(v_1, v_2)$. 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 \( \text{unbdsign}_k \) for some \( k \geq 1 \). We have that \( v = \text{sign}(v_1, v_2) \). Let \( P_0 \) (resp. \( P_1, \ldots, P_k \)) be the direct subproofs of \( P \) whose root is labeled with \( T \vdash \text{sign}(b_k(v_1, t_1, \ldots, t_k), v_2) \) (resp. \( T \vdash t_1, \ldots, T \vdash t_k \)). 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 2.4. 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 \( u \in st(T) \).

\[ \square \]

We further distinguish between normal proofs that yield the same conclusion:

**Definition 2.6** (simple proof). Let \( T_1 \subseteq T_2 \subseteq \ldots \subseteq T_n \). A proof \( P \) of \( T_i \vdash u \) is left-minimal if for any \( j \) such that \( T_j \vdash u \), we have that \( \text{Hyp}(P) \subseteq T_j \). A proof \( P \) is simple if any of its subproofs is left-minimal and in normal form.

**Example 2.7.** Let \( T_1 = \{ a \} \). The normal proof given in Example 2.3 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 2.8.** 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 \( T_i \vdash u \).

### 3. Constraint systems

An elementary deducibility constraint is an expression \( T \vdash u \) where \( T \) is a finite set of terms and \( u \) is a term. The solutions of such a constraint are the substitutions \( \sigma \) such that \( T \sigma \vdash u \). The idea is to represent through such constraints all the possible executions of a protocol.

#### 3.1. 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 its private key \( k \) and finally \( A \) checks that the message he receives is \( \text{sign}(n, k) \). The possible traces obtained in any successful session of this protocol, between the two parties \( a, b \) are sequences of four messages:

\[
\begin{align*}
n, & \quad x, \quad \text{sign}(x, k), \quad \text{sign}(n, k)
\end{align*}
\]

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

\[
\begin{align*}
\{ n \} & \vdash x \quad \text{and} \quad \{ n, \text{sign}(x, k) \} \vdash \text{sign}(n, k)
\end{align*}
\]

Intuitively, the attacker should be able to construct \( x \) and to construct back \( \text{sign}(n, k) \) from \( n \) and \( \text{sign}(x, k) \). The set of such possible messages \( x \) includes \( n \), but also \( \text{blind}(n, n) \)
for instance, since the attacker can unblind \( \text{sign}(\text{blind}(n, n), k) \) and get \( \text{sign}(n, k) \). Actually, the set of possible messages \( x \) satisfying the above constraints is \( \mathcal{Bd}(\{n\}, n) \) where \( \mathcal{Bd}(T, u) \) denotes the least set \( S \) of terms that contains \( u \) and such that \( \text{blind}(s, v) \in S \) when \( s \in S \) and \( T \vdash v \). In other words, we have that
\[
\mathcal{Bd}(T, u) = \{u\} \cup \{b_k(u, v_1, \ldots, v_k) \mid k \in \mathbb{N} \text{ and } T \vdash v_i \text{ for each } 1 \leq i \leq k\}.
\]
For each \( t \in \mathcal{Bd}(\{n\}, n) \), the attacker can compute \( \text{sign}(n, k) \) using a single inference step, using one instance of the \textit{unblind} rule scheme and, these are the only ways to satisfy the last constraint. 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.

3.2. Constraint systems. We will consider two different kinds of \textit{elementary constraints}: a \textit{deducibility constraint} is a constraint of the form \( T \vdash u \), whereas a \textit{membership constraint} is a constraint of the form \( v \in \mathcal{Bd}(T, u) \). 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 \( C \).

Given a finite set \( \mathcal{D} \) of elementary constraints and \( x \in \text{vars}(\mathcal{D}) \), we let \( T_x \) be the minimal set of terms w.r.t. inclusion (when it exists) such that
- \( T_x \vdash u \in \mathcal{D} \) with \( x \in \text{vars}(u) \), or
- \( v \in \mathcal{Bd}(T_x, u) \in \mathcal{D} \) with \( x \in \text{vars}(u) \).

\textbf{Definition 3.1} (constraint system). A \textit{constraint system} \( C \) is either \( \perp \) or a set of elementary constraints. We require that the constraints in \( C \) can be ordered \( C_1, \ldots, C_\ell \) in such a way that the following conditions are satisfied:

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

where each \( C_i \) is of the form \( T_i \vdash u_i \) or of the form \( v_i \in \mathcal{Bd}(T_i, u_i) \).

Lastly, we assume that for each variable \( x \in \text{vars}(C) \):
- either there exists \( T_x \vdash u \) in \( C \) with \( x \in \text{vars}(u) \),
- or there exists \( v \in \mathcal{Bd}(T_x, u) \) in \( C \) with \( x \in \text{vars}(u) \) and such that \( T_y \subseteq T_x \) for every \( y \in \text{vars}(v) \).

A constraint system can also be seen as a conjunction of elementary constraints. \textit{Pure constraint systems} are constraint systems that only contain deducibility constraints. The set of possible traces of a (bounded) security protocol can be described by pure constraint systems.

\textbf{Example 3.2}. The following set of constraints
\[
\{a \vdash x, \text{blind}(x, a) \in \mathcal{Bd}(\{a\}, y)\}
\]
satisfies monotonicity and origination. However, it is not a constraint system. Indeed, the constraint \( \text{blind}(x, a) \in \mathcal{Bd}(\{a\}, y) \) introduces the variable \( y \) and the condition \( T_x \subseteq \{a\} \) is not satisfied. Actually, we have that \( T_x = \{a\} \).
\[
S_{ax}: \quad u \not\in Bd(T, u) \quad \rightarrow \quad \top
\]
\[
S_{bd}: \quad \text{blind}(u, v) \not\in Bd(T, w) \quad \rightarrow \quad T \vdash v \wedge u \not\in Bd(T, w) \quad \text{if \ blind}(u, v) \neq w
\]
\[
S_{f}: \quad f(t_1, \ldots, t_n) \not\in Bd(T, v) \quad \rightarrow \quad \bot \quad \text{if} \ f \neq \text{blind} \text{ and } f(t_1, \ldots, t_n) \neq v
\]
\[
S_{\text{cycle}}: \quad x_1 \in Bd(T_1, v_1[x_2]) \wedge \ldots \wedge x_n \in Bd(T_n, v_n[x_1]) \rightarrow \bot
\]
\[\text{if there exists } i \text{ such that } v_i \neq \epsilon \text{ or } \#\{x_1, \ldots, x_n\} > 1.\]

Figure 3: Simplification rules for membership constraints

\textbf{Definition 3.3 (solution).} A solution of a set \( D \) of elementary constraints is a substitution \( \theta \) such that \( T \theta \vdash v \theta \) for each \( T \vdash v \in D \), and \( v \theta \in Bd(T \theta, u \theta) \) for each \( v \in Bd(T, u) \in D \). We denote by \( \text{Sol}(D) \) the set of solutions of \( D \). The constraint system \( \bot \) has no solution.

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

\[D\vert_V = \{C \in D \mid \text{vars}(C) \subseteq V\} .\]

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 \( \sigma \) is non-confusing w.r.t. \( T \) if for any \( t_1, t_2 \in \text{st}(T) \) such that \( t_1 \neq t_2 \), we have that \( t_1 \sigma \neq t_2 \sigma \). A non-confusing solution of a set \( D \) of elementary constraints is a substitution \( \theta \in \text{Sol}(D) \) such that \( \theta \) is non-confusing w.r.t. terms that appear in \( D \). We denote by \( \text{Sol}_{\text{NC}}(D) \) the set of solutions of \( D \) that are non-confusing.

\subsection*{3.3. Simplified form.}

The constraints are simplified according to the simplification rules described in Figure 3. They reflect the semantics of \( u \not\in Bd(T, v) \). In the rule \( S_{\text{cycle}} \), 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 \( v \neq \epsilon \).

\textbf{Definition 3.4 (simplified form).} A set \( D \) of elementary constraints is in simplified form if none of the simplification rules can be applied. \( D_{\downarrow_S} \) is the set of irreducible constraints obtained from \( D \) by repeatedly applying these rules.

Note that in a set of constraints in simplified form, each membership constraint is of the form \( x \not\in Bd(T, u) \) 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. More formally, we have that:

\textbf{Lemma 3.5.} Let \( D \) and \( D' \) be two sets of elementary constraints such that \( D \rightarrow D' \). We have that:

- If \( D \) is a constraint system then \( D' \) is a constraint system;
- \( \text{Sol}(D') \subseteq \text{Sol}(D) \) and \( \text{Sol}_{\text{NC}}(D) \subseteq \text{Sol}_{\text{NC}}(D') \).

From this lemma, we easily derive the following results.

\textbf{Corollary 3.6.} Let \( D \) be a set of elementary constraints. We have that:

- If \( D \) is a constraint system then \( D_{\downarrow_S} \) is a constraint system;
Let \( \mathcal{D}_S \subseteq \text{Sol}(\mathcal{D}) \) and \( \text{Sol}_N(\mathcal{D}) \subseteq \text{Sol}_N(\mathcal{D}_S) \).

**Corollary 3.7.** Let \( \mathcal{D} \) and \( \mathcal{D}' \) be two sets of constraints such that \( \mathcal{D} \rightarrow^* \mathcal{D}' \) and \( V \subseteq \text{vars}(\mathcal{D}) \). We have that \( \text{Sol}(\mathcal{D}'|_V) \subseteq \text{Sol}(\mathcal{D}|_V) \).

**Proof.** We prove this result by induction on the length of the derivation \( \mathcal{D} \rightarrow^* \mathcal{D}' \). It is easy to see that if \( \mathcal{D}_1 \rightarrow \mathcal{D}_2 \), we are in one of the following cases: \( \mathcal{D}_1|_V \rightarrow \mathcal{D}_2|_V \) or \( \mathcal{D}_1|_V \subseteq \mathcal{D}_2|_V \) or \( \mathcal{D}_2|_V = \bot \). We easily conclude for the two last situations. We conclude by relying on Lemma 3.5 to deal with the first situation.

\[ \square \]

4. 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 4.1** (solved form). A constraint system \( \mathcal{C} = \{C_1, \ldots, C_l\} \) is in *solved form* if each \( C_i \) is either of the form \( T_i \vdash x_i \) or of the form \( x_i \in \text{Bd}(T_i, u_i) \) where \( x_i \) is a variable. Moreover, for every \( x \in \text{vars}(\mathcal{C}) \), there is a unique deducibility constraint \( T \vdash x \in \mathcal{C} \) and there is at most one membership constraint \( x \in \text{Bd}(T', u) \) in \( \mathcal{C} \) and, if this is the case, \( T' = T \).

**Example 4.2.** Below, the systems \( \mathcal{C}_2 \) and \( \mathcal{C}_3 \) are in solved form whereas \( \mathcal{C}_1 \) is not.

\[
\mathcal{C}_1 = \left\{ \begin{array}{l}
a \vdash y \\
y \in \text{Bd}\{a\}, a \\
y \in \text{Bd}\{a\}, \text{blind}(a, a)
\end{array} \right\} \quad \mathcal{C}_2 = \left\{ \begin{array}{l}
a \vdash x \\
x \in \text{Bd}\{a\}, b
\end{array} \right\} \quad \mathcal{C}_3 = \left\{ \begin{array}{l}
a \vdash x \\
a, x \vdash y \\
y \in \text{Bd}\{a, x\}, x
\end{array} \right\}
\]

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 \( \leq_D \) on \( \text{vars}(\mathcal{D}) \) as the least relation closed by transitivity and reflexivity and such that:

\[
x \in \text{Bd}(T, u) \text{ in } \mathcal{D} \text{ and } y \in \text{vars}(u) \Rightarrow y \leq_D x.
\]

Note that, due to the rule \( S_{\text{cycle}} \) and the fact that \( \mathcal{D} \) is in simplified form, if \( x \leq_D y \) and \( y \leq_D x \) then \( x = y \). Hence \( \leq_D \) is an ordering. Moreover, this ordering is compatible with monotonicity, i.e. we have the following lemma.

**Lemma 4.3.** Let \( \mathcal{C} \) be a constraint system in simplified and solved form. If \( x \leq_C y \), then \( T_x \subseteq T_y \).
A simplified constraint system

We consider an ordering between variables defined as follows:

Let \( x, y \) be two variables such that \( x \leq_C y \). By definition of \( \leq_C \), we know that there exists \( n \), and some membership constraints

\[
x_1 \in \mathcal{Bd}(T_1, u_1), \; x_2 \in \mathcal{Bd}(T_2, u_2), \ldots, \; x_n \in \mathcal{Bd}(T_n, u_n)
\]

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

Since the constraint system \( C \) is in solved form we have that \( T_{x_i} = T_i \) for \( i \in \{1, \ldots, n\} \).

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 this partial order to a pre-order on sets of variables as follows:

\[
V_1 \preceq_D V_2 \text{ if, and only if, } \forall x \in V_1, \exists y \in V_2 \text{ such that } x \leq_D y.
\]

We denote by \( C^<_V \) the set of constraints in \( C \) containing only variables smaller or equal by \( \leq_C \) to those in \( V \), i.e. \( C^<_V = \{ C \in C \mid \text{vars}(C) \leq_C V \} \). Note that \( C^<_V \) is not necessarily a constraint system.

**Definition 4.4** (well-formed). A simplified constraint system \( C \) is **well-formed** if, for every constraint \( y \in \mathcal{Bd}(T_i, u_i) \) in \( C \), either \( T_y \subseteq T_i \) or else \( T_y = T_i \) and \( C^<_V \models (T_i \vdash u_i) \) where \( V = \text{vars}(T_i \cup \{u_i\}) \).

**Example 4.5.** The constraint system \( C_2 \) (see Example 4.2) is not well-formed. Indeed, \( T_x = \{a\} \) and thus the first condition does not hold. Moreover, \( C^<_V = \emptyset \) and \( \emptyset \not\models (a \vdash b) \).

The systems \( C_1 \) and \( C_3 \) are well-formed.

**Lemma 4.6.** Any solved well-formed simplified constraint system \( C \) 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 \( C \land t_1 \neq u_1 \land \cdots \land t_m \neq u_m \) has a solution.

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

\[
x \leq y \text{ if, and only if, } \begin{cases} \text{either } T_x \subseteq T_y, \\ T_x = T_y \text{ and } x \leq_C y. \end{cases}
\]

Thanks to Lemma 4.3, we know that \( x \leq_C y \) implies \( T_x \subseteq T_y \), and thus \( x \leq y \). Hence, we know that \( \leq \) is compatible with \( \leq_C \).

Let \( x_1, \ldots, x_n \) be the variables in \( C \) renamed in such a way that \( x_i \leq x_j \) implies \( i \leq j \).

We consider the constraint system \( C \) in which the constraints are ordered according to the sequence \( x_1, \ldots, x_n \), i.e.

\[
C := \left\{ T_1 \vdash x_1 \land [x_1 \in \mathcal{Bd}(T_1, u_1)] \right\} \ldots \left\{ T_n \vdash x_n \land [x_n \in \mathcal{Bd}(T_n, u_n)] \right\}
\]

The notation \([x \in \mathcal{Bd}(T, u)]\) 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 $C$ is a constraint system. To prove this, we first have to show that $\text{vars}(T_i) \subseteq \{x_1, \ldots, x_{i-1}\}$. Let $y \in \text{vars}(T_i)$. We have that $T_y \subseteq T_i$ and thus $y \leq x_i$, i.e. $y \in \{x_1, \ldots, x_{i-1}, x_i\}$. Actually, $y \neq x_i$ since $C$ is in solved form. This allows us to conclude. Secondly, we also have that $\text{vars}(u_i) \subseteq \{x_1, \ldots, x_{i-1}\}$. Indeed, let $y \in \text{vars}(u_i)$, 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 $\sigma \in \text{Sol}(C)$ such that, for every $1 \leq i \leq m$, $t_i \sigma \neq u_i \sigma$.

**Base case:** $n = 0$. Then $C$ 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 $\theta$, that is a solution of $T_i \vdash x_i$ and $\{x_i \in \text{Bd}(T_i, u_i)\}$ 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 unknown $x_n$ has at most one solution $v_j$. We distinguish several cases:

- If there is no constraint $T_n \vdash x_n$ in $C$, 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 $T_n \vdash x_n$ in $C$ but no constraint $x_n \notin \text{Bd}(T_n, u_n)$, we let $w_0 \in T_n \theta$ and $w_{k+1} = \text{sign}(w_k, w_0)$. For every $k$, $T_n \vdash w_k$ 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 $x_n \notin \text{vars}(T_n \theta)$, $T_n \sigma \vdash x_n \sigma$ and since $x_n \sigma \notin \{v_1, \ldots, v_m\}$, $t_j \sigma$ and $u_j \sigma$ are distinct for every $j$. Hence $\sigma$ has the expected property.

- Now, assume that there is a constraint $T_n \vdash x_n$ and a constraint $x_n \notin \text{Bd}(T_n, u_n)$ in $C$. We have shown that $\text{vars}(T_n \cup \{u_n\}) \subseteq \{x_1, \ldots, x_{n-1}\}$. We let $w_0 \in T_n$ and, for any $k \geq 1$, we let $w_k = b_k(u_n \theta, w_0 \theta, \ldots, w_0 \theta)$. Then, by well-formedness of $C$, $\langle C \setminus \{T_n \vdash x_n\} \rangle \models T_n \vdash u_n$, hence $T_n \theta \vdash u_n \theta$ and therefore $T_n \theta \vdash w_k$ for every $k$. Furthermore, $w_k \in \text{Bd}(T_n \theta, u_n \theta)$ 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 $T_n \sigma \vdash x_n \sigma$, $x_n \sigma \in \text{Bd}(T_n \sigma, u_n \sigma)$ and $t_j \sigma$ is distinct from $u_j \sigma$ for every $j$.

5. Transformation rules

The constraint solving rules are displayed in Figure 4. The rules $R_{ax}, R_{\text{triv}}, R_i, R_{\text{bd}}, R_{\text{get}},$ and $R_{\text{bsgn}}$ will be applied when the corresponding inference rules end the proof of an unsolved deducibility constraint. Note that $R_{\text{bsgn}}$ introduces a membership constraint, that captures the infinite set of inference rules $\text{unbsign}_n, n \geq 1$. Furthermore, the rules $R_A, R_B, R_C$ 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 $R_g$ 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 5.1) and completeness (see Section 5.3) of our set of transformation rules. We also show that the
The rules of Figure 4 transform a constraint system into a constraint system, let $s$ satisfies the requirement since $u \notin \Sigma$. This allows us to conclude in this case. In case of variables, we have that $f \in \{\text{sign, blind}\}$.

Moreover, we have that $\exists \mathcal{E} \models \text{triv}$. The rule $\text{R}_{\text{bd}} : T \not\triangleright v \Rightarrow T \not\triangleright \text{blind}(v, u) \land T \not\triangleright u$ if blind $(v, u) \in st(T)$.

$\text{R}_{\text{get}} : T \not\triangleright v \Rightarrow T \not\triangleright \text{sign}(v, u)$ if sign $(v, u) \in st(T)$.

$\text{R}_{\text{bds}} : T \not\triangleright \text{sign}(v, u) \Rightarrow T \not\triangleright \text{sign}(w, u) \land w \in \mathcal{B}d(T, v)$ if sign $(w, u) \in st(T)$.

$\text{R}_{\text{A}} : T \not\triangleright x \land x \in \mathcal{B}d(T', v) \Rightarrow T \not\triangleright x \land T \not\triangleright v \land x \in \mathcal{B}d(T, v)$ if $T \subseteq T'$.

$\text{R}_{\text{B}} : T \not\triangleright x \land x \in \mathcal{B}d(T', v) \Rightarrow T \not\triangleright x \land T \not\triangleright w \land x \in \mathcal{B}d(T, w) \land w \in \mathcal{B}d(T', v)$ if $T \subseteq T'$ and $w \in st(T)$.

$\text{R}_{\text{C}} : T \not\triangleright x \land x \in \mathcal{B}d(T, v) \land x \in \mathcal{B}d(T, v') \Rightarrow T \not\triangleright x \land x \in \mathcal{B}d(T, v) \land v \in \mathcal{B}d(T, v')$ if $T_x = T'$.

**Figure 4: Transformation rules**

5.1. **Soundness.** We show that our rules transform a constraint system into a constraint system (Lemma 5.1) and we show in Lemma 5.2 that our rules are sound, i.e. when $D \sim D'$, we have that $\text{Sol}(D') \subseteq \text{Sol}(D)$.

**Lemma 5.1.** The rules of Figure 4 transform a constraint system into a constraint system, i.e. if $\mathcal{C}$ is a simplified constraint system and $\mathcal{C} \sim \mathcal{C}'$ then $\mathcal{C}'_{\downarrow S}$ is a constraint system. Moreover, we have that $\text{st}(\mathcal{C}'_{\downarrow S}) \subseteq \text{st}(\mathcal{C})$.

**Proof.** Let $\mathcal{C}$ be simplified constraint system and $\mathcal{C}'$ be such that $\mathcal{C} \sim \mathcal{C}'$. 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 3.1 for variables is also satisfied. The rule $\text{R}_1$ does not cause any trouble. The rules $\text{R}_{\text{ax}}, \text{R}_{\text{triv}}, \text{R}_{\text{bd}},$ and $\text{R}_{\text{get}}$ affect only elementary constraints that do not introduce variables for the first time. If the rule $\text{R}_{\text{bds}}$ is applied, then even if the additional membership constraint $w \not\in \mathcal{B}d(T, v)$ introduces a variable for the first time, we have that $T_y \subseteq T$ for each $y \in \text{vars}(w)$ since $w \in st(T)$. This allows us to conclude in this case. In case of $\text{R}_{\text{A}}$ (resp. $\text{R}_{\text{B}}$), the additional membership constraint on $x$ does not introduce any variable because of the presence of the deducibility constraint $T \not\triangleright v$ (resp. $T \not\triangleright w$). In case of $\text{R}_{\text{B}}$, assume that the membership constraint $w \in \mathcal{B}d(T', v)$ introduces a variable. We have that $T_y \subseteq T \subseteq T'$ for each $y \in \text{vars}(w)$. This is due to the fact that $w \in st(T)$. In case of $\text{R}_{\text{C}}$, the additional membership constraint satisfies the requirement since $v'$ can not introduce any variable. Indeed, otherwise, we would have that the system on which we apply this rule does not satisfy the condition on
membership constraint. This allows us to conclude that $C'$ is a constraint system. Then, we deduce that $C' \downarrow S$ is a constraint system thanks to Corollary 3.6. Moreover, if $C \leadsto^* C'$, it is clear that $sI(C') \subseteq sI(C)$ since the rules never introduce a new subterm.

**Lemma 5.2** (soundness). Let $D$ be a set of elementary constraints in simplified form and $D'$ be a set of constraints such that $D \leadsto D'$. We have that $\text{Sol}(D' \downarrow S) \subseteq \text{Sol}(D)$.

*Proof.* Let $R$ be the transformation rule used in the step $D \leadsto D'$ and $\sigma \in \text{Sol}(D' \downarrow S)$. First, thanks to Lemma 3.5, we have that $\sigma \in \text{Sol}(D')$. Then we show that $\sigma \in \text{Sol}(D)$ by case analysis on $R$.

- **Case $R_{sx}$, $R_{\text{triv}}$, $R_{t}$, $R_{\text{bd}}$, $R_{\text{get}}$, and $R_{\text{bdsgn}}$**: The proof trees witnessing the fact that $\sigma \in \text{Sol}(D)$ are easily obtained from those witnessing the fact that $\sigma \in \text{Sol}(D')$.

- **Case $R_{A}$**: The proof trees witnessing the fact that $\sigma \in \text{Sol}(D')$ can be used to show that $\sigma \in \text{Sol}(D)$. The proof tree witnessing $T \vdash v$ is even not useful for that.

- **Case $R_{B}$**: We have to group together the sequences of proof trees witnessing $x\sigma \notin Bd(T\sigma, w\sigma)$ and $w\sigma \in Bd(T'\sigma, v\sigma)$ in order to obtain a witness of the fact that $x\sigma \in Bd(T'\sigma, v\sigma)$.

- **Case $R_{C}$**: We have to group together the sequences of proof trees witnessing $x\sigma \notin Bd(T\sigma, v\sigma)$ and $v\sigma \in Bd(T\sigma, v'\sigma)$ in order to show that $x\sigma \in Bd(T\sigma, v'\sigma)$.

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

**Proposition 5.3** (soundness). Let $D$ and $D'$ be two sets of elementary constraints in simplified form. If $D \leadsto^* D'$ and $\sigma \in \text{Sol}(D')$ then $\sigma \in \text{Sol}(D)$.

### 5.2. 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 5.4 and Proposition 5.5 are detailed in Appendix C.

**Lemma 5.4** (property of $\leq_D$). Let $D$ and $D'$ be two sets of constraints in simplified form such that $D \leadsto D'$ and $D' \not\leq \bot$. We have that $\leq_D \leq \leq_{D'}$.

**Proposition 5.5.** Let $D$ and $D'$ be two sets of constraints in simplified form such that $D \leadsto D'$, and $V \subseteq \text{vars}(D)$. We have that $D_V \models D'_{V'}$.

**Proposition 5.6.** Let $C$ be a simplified constraint system that is well-formed and such that $C \leadsto C'$. Then $C' \downarrow S$ is a well-formed constraint system.

*Proof.* We consider each possible transformation rule applied on $C$ and show that each membership constraint $M = x \in Bd(T, u)$ in $C' \downarrow S$ is such that:

- either $T_x \not\subseteq T$,
- or $T_x = T$ and $(C' \downarrow S)_{V'} \models (T \vdash u)$ where $V = \text{vars}(T \cup \{u\})$.

Let us first consider a membership constraint $M = x \in Bd(T, u)$ that is also in $C$. If we have that $T_x \not\subseteq T$ in the constraint system $C$ then $T_x \not\subseteq T$ also holds in the constraint
system $C'_\downarrow_S$ and we conclude. Otherwise, we have that $T_x = T$ (in $C$) and $C'_\uparrow \models (T \vdash \exists u)$. Applying Proposition 5.5 we deduce that $(C'_\downarrow_S)^\uparrow \models C'_\uparrow$, and thus $(C'_\downarrow_S)^\uparrow \models (T \vdash \exists u)$. Therefore, in the following, we concentrate only on membership constraints that are in $C'_\downarrow_S$ and not in $C$.

- Rules $R_{ax}$, $R_{triv}$, $R_f$, $R_{bd}$ and $R_{get}$. There are no new membership constraints. Hence, we easily conclude.

- Rule $R_{bdsign}$. We have that $C' = (C \setminus \{T \vdash \exists \text{sign}(v,u)\}) \cup \{T \vdash \exists \text{sign}(w,u), w \in Bd(T,v)\}$ with $\text{sign}(w,u) \in \text{st}(T)$. Since $w \in \text{st}(T)$, we have that $T_y \subseteq T$ for every $y \in \text{vars}(w)$. This allows us to conclude.

- Rule $R_A$. We have that $C' = (C \setminus \{x \in Bd(T',v)\}) \cup \{T \vdash v, x \in Bd(T,v)\}$ with $T \subseteq T'$. Clearly, we have that $(C'_\downarrow_S)^\uparrow \models (T \vdash v)$.

- Rule $R_B$. We have that $C' = (C \setminus \{x \in Bd(T,v')\}) \cup \{T \vdash w, x \in Bd(T,w), w \in Bd(T',v)\}$. We have that $T_y \subseteq T'$ for every $y \in \text{vars}(w)$ and $T \vdash w$ is in $C'$. This allows us to conclude.

- Rule $R_C$. We have that $C' = (C \setminus \{x \in Bd(T,v')\}) \cup \{v \in Bd(T,v')\}$. Using $T_x = T$ and $x \in Bd(T,v')$ in $C$ (well-formed), we deduce that $C'_\uparrow \models (T \vdash v')$ where $V' = \text{vars}(T \cup \{v'\})$. Thus, thanks to Proposition 5.5, $(C'_\downarrow_S)^\uparrow \models (T \vdash v')$. This allows us to conclude.

\[\square\]

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

In the remainder we consider a constraint system $C$ 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 $T\sigma \vdash u\sigma$ for $T \vdash u \in C$, we refer to the sequence $T_1\sigma \subseteq \ldots \subseteq T_n\sigma$. Given a set of terms $T$, we denote by $C(T)$ the elementary constraints in $C$ that have $T$ as associated set of terms, i.e.

\[C(T) = \{C \in C \mid C = T \vdash u \text{ or } C = v \in Bd(T,u) \text{ for some terms } u, v\}\]

Let $C$ be an unsolved simplified constraint system. We denote by $T_{\text{min}}$ the minimal (w.r.t. inclusion) set of terms such that $\bigcup_{T_1 \subseteq T_{\text{min}}} C(T_i)$ is unsolved. Let $S \subseteq C(T_{\text{min}})$ be a maximal set of constraints such that $\bigcup_{T_i \subseteq T_{\text{min}}} C(T_i) \cup S$ is solved. Then $M_C = C(T_{\text{min}}) \setminus S$ are the minimal unsolved constraints of $C$. 

Example 5.7. Consider the constraint system $C_1$ given in Example 4.2. This system is unsolved. $C_1(\{a\}) = C_1$, and $T_{\text{min}} = \{a\}$. For $M_C$, there are two possibilities: $M_C = \{x \in Bd(\{a\}, \text{blind}(a, a))\}$ or $M_C = \{x \in Bd(\{a\}, a)\}$.

To establish completeness, we first lift Lemma 2.4 to deal with deducibility constraints with variables.

Lemma 5.8. Let $\mathcal{C}$ be an unsolved constraint system. Let $\sigma \in \text{Sol}(\mathcal{C})$ and $t$ be a term such that $T_{\text{min}} \vdash t$. Let $P$ be a simple proof of $T_{\text{min}} \vdash t$. If $P$ is reduced to a leaf or ends with a decomposition rule, then there exists $u \in st(T_{\text{min}}) \setminus X$ such that $\sigma u = t$.

Proof. We know that $T_{\text{min}} \vdash t$. Let $i_0$ be the minimal indice we need to consider to have that $T_{i_0} \vdash t$. Since $P$ is simple, $P$ is also a simple proof of $T_{i_0} \vdash t$. Thanks to Lemma 2.4, we have that $t \in st(T_{i_0})$. Now, we show that $t \in (st(T_{i_0}) \setminus X)\sigma$. Note that $T_{i_0} \subseteq T_{\text{min}}$. We consider two cases:

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

Case $i_0 > 1$: In such a case, either $t \in st(T_{i_0}) \setminus X)\sigma$ and we easily conclude. Otherwise, we have that $t \in st(x\sigma)$ for some $x \in \text{vars}(T_{i_0})$. Let us consider such a variable $x$ with $T_x \subseteq T_{i_0}$. Then, by definition of $T_{\text{min}}$, we know that there exists $T_x \vdash x \in \mathcal{C}$ and thus we have that $T_x \vdash x\sigma$. By corollary 2.5, either $t \in st(T_x)\sigma$, or else we have $T_x \vdash t$. In the latter case, we contradict the minimality of $i_0$, since $T_x \subseteq T_{i_0}$. In the former case, we have either $t \in (st(T_x) \setminus X)\sigma$, and we conclude, or else $t \in st(y)\sigma$, for some $y \in \text{vars}(T_x)$. By the choice of $x$, the last case is impossible. Therefore, in all cases we conclude that $t \in (st(T_{i_0}) \setminus X)\sigma$, and thus $t \in (st(T_{\text{min}}) \setminus X)\sigma$. □

Let $\mathcal{C}$ be a constraint system, $\sigma \in \text{Sol}(\mathcal{C})$ and $P_1, \ldots, P_k$ be a sequence of simple proofs witnessing the fact that $\sigma$ 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 $\mu(\mathcal{C}\sigma)$ 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 5.9 (completeness - deducibility constraint). Let $\mathcal{C}_{\downarrow S}$ be an unsolved constraint system such that $M_{\mathcal{C}_{\downarrow S}}$ contains a deducibility constraint. Let $\sigma \in \text{Sol}_{\text{NC}}(\mathcal{C}_{\downarrow S})$. There exists a constraint system $\mathcal{C}'$ such that $\mathcal{C}_{\downarrow S} \rightsquigarrow \mathcal{C}'$, $\sigma \in \text{Sol}_{\text{NC}}(\mathcal{C}')$ and $\mu(\mathcal{C}'\sigma) < \mu(\mathcal{C}_{\downarrow S}\sigma)$.
### Proposition 5.10 (completeness - membership constraint)

Let $C_{\downarrow S}$ be an unsolved constraint system such that $M_{C_{\downarrow S}}$ only contains membership constraints. Let $\sigma \in \text{Sol}_{\text{NC}}(C_{\downarrow S})$. There exists a constraint system $C'$ such that $C_{\downarrow S} \rightsquigarrow C'$, $\sigma \in \text{Sol}_{\text{NC}}(C')$ and $\mu(C'\sigma) < \mu(C_{\downarrow S}\sigma)$.

<table>
<thead>
<tr>
<th>$C_{\downarrow S}$ contains among others</th>
<th>$M_{C_{\downarrow S}}$ contains $C$</th>
<th>Last rule in the proof $P$ witness of $C$</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T \vdash x$</td>
<td>$T' \vdash x$</td>
<td>$\text{axiom}$</td>
<td>$R_{\text{triv}}$</td>
</tr>
<tr>
<td>$T \vdash u$</td>
<td>$T' \vdash u$</td>
<td>$\text{sign, blind}$</td>
<td>$R_{\text{ax}}$</td>
</tr>
<tr>
<td>$T \vdash f(u_1, u_2)$</td>
<td>$T' \vdash f(u_1, u_2)$</td>
<td>$\text{unblind}$</td>
<td>$R_{\text{triv}}$</td>
</tr>
<tr>
<td>$T \vdash u$</td>
<td>$T' \vdash u$</td>
<td>$\text{getmsg}$</td>
<td>$R_{\text{ax}}$</td>
</tr>
<tr>
<td>$T \vdash \text{sign}(u_1, u_2)$</td>
<td>$T' \vdash \text{sign}(u_1, u_2)$</td>
<td>$\text{unbdsign}$</td>
<td>$R_{\text{bd}}$</td>
</tr>
</tbody>
</table>

### Corollary 5.11.

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

**Proof.** We show this result by induction on $\mu(C\sigma)$.

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

**Induction step:** $\mu(C\sigma) \neq \emptyset$. Either $C$ 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 5.9 or Proposition 5.10. We deduce that there exists $C''$ such that $C \rightsquigarrow C''$, $\sigma \in \text{Sol}_{\text{NC}}(C'')$ and $\mu(C''\sigma) < \mu(C\sigma)$. Thanks to Lemma 3.5, we deduce that $\sigma \in \text{Sol}_{\text{NC}}(C''_{\downarrow S})$, and it is easy to check that our measure $\mu$ does not increase when we apply the simplification rules, i.e. $\mu(C''_{\downarrow S}\sigma) \leq \mu(C''\sigma)$. Altogether, this allows us to conclude.

### 6. Decision procedure

Let $C_0$ be a pure constraint system. Our simplification procedure works as follows:
(1) Guess a set of equalities \( E \) between subterms of \( C_0 \). Solve \( E \): let \( \theta_E \) be a mgu of \( E \) (if there is no solution, then return \( \perp \)).

(2) Apply non-deterministically the transformation rules (Figure 4) on each \( C \theta_E \) until either a solved form is reached or a loop is detected (i.e. \( C \sim^* C \) with a derivation of length at most one), in which case we return \( \perp \).

Considering all possible non-deterministic choices that do not yield \( \perp \), the procedure computes \( \text{solve}(C_0) \) a finite set of pairs \( (E_i, C_i) \) such that every \( C_i \) is in solved form.

**Theorem 6.1.** 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.

**6.1. Guessing equalities.** As a first step, we guess all equalities between subterms of a constraint system \( C \). 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 \( C_0 \), we guess an equivalence relation \( \approx \) on \( st(C_0) \). Then we consider the unification problem \( \bigwedge_{s \in t} s = t \). Let \( \theta \) be a mgu of this set of equations (if any), which does not introduce new variables. Then we consider the pure constraint system \( C_0 \theta \). 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. Given two substitutions \( \sigma \) and \( \theta \), their composition is a substitution denoted by \( \sigma \circ \theta \) and defined by \( x(\sigma \circ \theta) = (x\theta)\sigma \), for all variable \( x \).

**Lemma 6.2.** Let \( C \) be a constraint system and \( \sigma \in \text{Sol}(C) \). There exists an equivalence relation \( \approx \) on \( st(C) \) and \( \sigma' \in \text{Sol}_{nc}(C\theta) \) such that \( \sigma = \sigma' \circ \theta \) where \( \theta = \text{mgu}(\bigwedge_{s \in t} s = t) \).

**Proof.** Let \( \sigma \) be a solution of \( C \). We let \( \approx \) be the equivalence relation on \( st(C) \) defined by \( t \approx u \) if, and only if, \( t\sigma = u\sigma \). Let \( \sigma_\approx \) be the most general unifier. By definition of the mgu, there is a substitution \( \sigma' \) such that \( \sigma = \sigma' \circ \sigma_\approx \), which implies that \( \sigma' \) is a solution of \( C\sigma_\approx \). Now, in order to show that \( \sigma' \in \text{Sol}_{nc}(C\sigma_\approx) \), it remains to show that \( \sigma' \) is non-confusing.

We use the following observation: if \( \tau \) is a mgu of \( u, v \) which does not introduce any new variable, then, for every variable \( x \in \text{vars}(u, v) ), \( x\tau \in (st(\{u, v\}) \setminus \lambda)\tau \).

Let \( t \in st(C\sigma_\approx) \). We want to show that \( t \in st(C)\sigma_\approx \). We have that either \( t \in st(C)\sigma_\approx \) or there is a variable \( x \) such that \( t \in st(x\sigma_\approx) \). In the latter case, let \( \{x_1, \ldots, x_n\} \subseteq \text{vars}(C) \) such that \( t \in st(x_i\sigma_\approx) \). Note that \( n \geq 1 \). Let \( i_0 \) be an index 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 \( x_{i_0}\sigma_\approx \in (st(C) \setminus \lambda)\sigma_\approx \). Hence, there exists \( t_{i_0} \in st(C \setminus \lambda) \) 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 index \( i_0 \).

So, for any \( t, u \in st(C\sigma_\approx) \), there are \( t_0, u_0 \in st(C) \) 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 \( t = u \).
6.2. 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 $R_f$ and $R_{bd}$ 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 $(E, C_s)$ is a pair returned by our procedure, and $\sigma$ is a solution of $C_s$, then there exists $\theta$ an mgu of $E$. Let $C_1 = C_0 \theta$. $C_1$ is a pure constraint system and thus $C_1$ is well-formed. $C_1 \rightsquigarrow C_s$ by a derivation of a finite length. Thanks to Proposition 5.3, $\sigma \in \text{Sol}(C_1)$ and thus $\sigma \circ \theta \in \text{Sol}(C_0)$.

Completeness. Let $\sigma \in \text{Sol}(C_0)$. By Lemma 6.2, there exists an equivalence relation $\approx$ on $st(C_0)$ and $\sigma' \in \text{Sol}_{NC}(C_0 \theta)$ such that $\theta = \text{mgu}(E)$ where $E = \Lambda_{\text{ass}} s = t$. We compute such a system $C_0 \theta$ during our first step. Let $\sigma = \sigma' \circ \theta$ and $C_1 = C_0 \theta$. $C_1$ is a pure constraint system. Now, we apply our transformation rules and thanks to Corollary 5.11, there exists $C_s$ in solved form such that $C_1 \rightsquigarrow C_s$ and $\sigma' \in \text{Sol}_{NC}(C_s)$. Then $\sigma = \sigma' \circ \theta$ is a solution of $(E, C_s)$.

Complexity. Deciding whether a constraint is satisfiable amounts to decide the existence of a solved form, thanks to Lemma 4.6. Though we did not prove it formally, this problem should be NP-complete, using a proof similar to those of [7, 14]: using a DAG representation of terms and avoiding redundant transformations, the length of a transformation sequence should be polynomial.

6.3. Application to trace properties. Consider now a general security property $\phi$, which is stated as a first-order formula whose free variables may contain some of the free variables of a constraint system $C$. For instance agreement properties can be naturally expressed in this way [7]. $C \land \neg \phi$ expresses the existence of an attack.

**Corollary 6.3.** The satisfiability of $C \land \neg \phi$ is decidable.

Indeed, it suffices to solve first $\phi$, getting finitely many formulas (using for instance the algorithm of [6]).

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

Then replace $x_i$ with $t_i$ in $C$ and compute the corresponding solved forms of the previous section. We conclude using Lemma 4.6: there is an attack if, and only if, there is a solved form for which no disequality became trivial.

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 [7].
Let only proofs that have the subformula property. The following lemma and corollary still hold.

\[
\text{Dolev-Yao inference rules, the infinite local deduction system displayed in Figure 5. We again, such rules impair the locality property.}
\]

**Lemma 7.3.**

**Corollary 7.2.**

\[
\text{Figure 5: Deduction rules for a homomorphic encryption mode}
\]

### 7. 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 components of a pair from the encryption of the pair itself:

\[
\begin{align*}
T \vdash u & \quad T \vdash v \\
T \vdash \text{enc}(u, v) & \\
\end{align*}
\]

\[
\begin{align*}
T \vdash \langle u, v \rangle & \\
T \vdash \text{enc}(u_1, v) & \\
T \vdash \text{enc}(u_2, v) & \text{for any } u_1 \in \mathcal{P}(T, u_2)
\end{align*}
\]

Again, such rules impair the locality property.

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

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.

**Lemma 7.1** (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 7.2.** 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 7.3.** 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 $T \vdash u$
- a membership constraint is a constraint of the form $v \in \mathcal{P}(T, u)$.

The definition of constraint system is similar to the one given for blind signatures. It is obtained by replacing $\mathcal{Bd}(T, u)$ 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 $\mathcal{C} \downarrow \mathcal{S}$ denotes a set of constraint systems.
Simplification rules described in Figure 7 transform a constraint system into a constraint system.

Lemma 7.4. 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 \( \mathcal{C} \sim \mathcal{C}' \) 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 7.5 (solved form). A constraint system \( \mathcal{C} = \{ C_1, \ldots, C_l \} \) is in solved form if: each \( C_i \) is either of the form \( T_i \vdash x_i \) or of the form \( x_i \in \mathcal{P}(T_i, u_i) \) where \( x_i \) is a variable.

Moreover, for every \( x \in \text{vars}(\mathcal{C}) \) there is a unique deducibility constraint \( T \vdash x \in \mathcal{C} \) and every membership constraint \( x \in \mathcal{P}(T', v) \) is such that \( T = T' \).

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 \( \text{vars}(\mathcal{D}) \) as the least relation closed by transitivity and reflexivity and such that:

\[
x \in \mathcal{P}(T, u) \in \mathcal{D} \text{ and } y \in \text{vars}(u) \implies y \leq_{\mathcal{D}} x.
\]

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 7.6. 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 \( \mathcal{C} \land t_1 \neq u_1 \land \cdots \land t_m \neq u_m \) 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 \( \mathbf{R_A} \). Indeed, this rule will be sufficient to reach a solved form. The transformation rule \( \mathbf{R_C} \) is not needed anymore since we allow several membership constraints of
Figure 9: Summary: completeness proof

<table>
<thead>
<tr>
<th>$C \downarrow_S$ contains among others</th>
<th>$M_{C \downarrow_S}$ contains $C$</th>
<th>Last rule in the proof $P$ witness of $C$</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T \vdash x$</td>
<td>$T' \vdash x$</td>
<td>$\text{axiom}$</td>
<td>$R_{\text{triv}}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{enc}$, $\text{pair}$</td>
<td>$R_f$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{proj}_1$, $\text{proj}_r$</td>
<td>$R_{\langle \rangle}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{dec}$</td>
<td>$R_{\text{dec}}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{hom}$</td>
<td>$R_{\text{homenc}}$</td>
</tr>
<tr>
<td>$T \vdash x$</td>
<td>$x \in P(T', u)$ with $T \subseteq T'$</td>
<td></td>
<td>$R_A$</td>
</tr>
</tbody>
</table>

the form $x \in P(T, v_1), \ldots, x \in P(T, v_n)$ in a solved form. Moreover, $R_B$ is not useful: it is always possible to apply $R_A$ since the semantics of $P(T, v)$ does not depend on its first argument.

**Proposition 7.7** (completeness). Let $C \downarrow_S$ be an unsolved constraint system. Let $\sigma \in \text{Sol}_{\text{NC}}(C \downarrow_S)$. There exists a constraint system $C'$ such that $C \downarrow_S \rightarrow C'$, $\sigma \in \text{Sol}_{\text{NC}}(C')$ and $\mu(C'\sigma) < \mu(C \downarrow_S \sigma)$.

**Theorem 7.8.** 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 7.9.** The satisfiability of $C \land \phi$, where $C$ is a pure constraint system and $\phi$ is a first-order formula with equality, is decidable for homomorphic encryption.

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

**References**


APPENDIX A. Existence of simple proofs

Lemma 2.8. 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 $T_i \vdash u$.

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

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

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

Case $i > 1$ and there is $j < i$ such that $T_j \vdash u$: Let $\pi'$ be a proof of $T_j \vdash u$. In such a case, we apply our induction hypothesis on $\pi'$ 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 $\pi$ of $T_i \vdash u$. Let $\pi'_1, \ldots, \pi'_n$ be the resulting simple proofs. If $\pi$ is obtained by applying an inference rule $R$ to $\pi_1, \ldots, \pi_n$ then let $\pi'$ be the proof obtained by applying $R$ to $\pi'_1, \ldots, \pi'_n$. Note that $\pi'$ 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 $\text{sign}$: in such a case the proof $\pi'$ is in normal form and we easily conclude.

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

Case 3. $R$ is $\text{getmsg}$: if $\pi'$ is already in normal form, we conclude. If $\pi'_1$ ends with an instance of $\text{sign}$ 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 $\pi'_1$ ends with an instance of $\text{unbdsign}$, we reach a normal form after application of the normalisation rule ($\text{unbdsign} / \text{getmsg}$). 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 $\text{sign}(b_n(u, v_1, \ldots, v_n), v)$ be the term labeling the root of $\pi'_1$. The “new” intermediate nodes are labeled with $b_k(u, v_1, \ldots, v_k)$ for some $k \in \{1, \ldots, n\}$. We have that $T_i \vdash b_k(u, v_1, \ldots, v_k)$ for any $1 \leq k \leq n$. Let $k$ be the smallest integer such that $T_j \vdash b_k(u, v_1, \ldots, v_k)$ for some $j < i$. If such a $k$ does not exist, this means that the proof $\pi'$ is left-minimal. Otherwise, there exists $k$ and $j < i$ such that $T_j \vdash b_k(u, v_1, \ldots, v_k)$. By relying on our induction hypothesis, we know that there exists $\pi_0$ a simple proof of $T_j \vdash b_k(u, v_1, \ldots, v_k)$. We have also some simple proofs $\pi'_1, \ldots, \pi'_k$ of $T_i \vdash v_1, \ldots, T_i \vdash v_k$. The proof obtained by applying several instances of $\text{unblind}$ on $\pi_0$ with $\pi'_1, \ldots, \pi'_k$ 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 $\pi_0$ can not end with an instance of $\text{blind}$. Otherwise, this would mean that $T_j \vdash b_{k-1}(u, v_1, \ldots, v_{k-1})$ and thus $k$ was not minimal.

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

• \( R_3 \) is \textit{sign}. Assume that the root of \( \pi'_1 \) is labeled with \texttt{sign}(b\(_n\)(u\(_0\), v\(_1\), . . . , v\(_n\)), v\(_0\)). Let \( \pi'' \) be the proof obtained from \( \pi' \) after normalisation. After application of the normalisation rule (\texttt{sign/ unbdsign}), the only rule that is applied to reach such a normal form is (\texttt{blind /unblind}). Subproofs that occur in \( \pi'' \) and not in \( \pi' \) are labeled with \( b_p(u_0, v_1, . . . , v_p) \) for some \( p \in \{1, . . . , n\} \). Moreover, we have that \( T_i \vdash b_k(u_0, v_1, . . . , v_k) \) for any \( 1 \leq k \leq n \). Let \( k \) be the smallest integer such that \( T_j \vdash b_k(u_0, v_1, . . . , v_k) \) 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 \( \pi'' \) is left-minimal. Otherwise, there exists \( k \) and \( j < i \) such that \( T_j \vdash b_k(u_0, v_1, . . . , v_k) \). By relying on our induction hypothesis, we know that there exists \( \pi_0 \) a simple proof of \( T_j \vdash b_k(u_0, v_1, . . . , v_k) \). (When \( k = 0 \), this means that we have a simple proof of \( T_j \vdash u_0 \)). We have also some simple proofs \( \pi^*_1, . . . , \pi^*_k \) of \( T_i \vdash v_1, . . . , T_i \vdash v_k \). The proof obtained by applying several instances of \texttt{unblind} on \( \pi_0 \) with \( \pi^*_1, . . . , \pi^*_k \) yields to a proof of \( T_i \vdash u_0 \) that is simple. Let \( \pi_L \) be the resulting proof. Indeed, the proof is left-minimal. The proof is also in normal form since \( \pi_0 \) can not end with an instance of \texttt{blind}. Otherwise, this would mean that \( T_j \vdash b_{k-1}(u, v_1, . . . , v_{k-1}) \) and thus \( k \) was not minimal. By induction hypothesis, we have also a simple proof \( \pi_R \) of \( T_i \vdash v_0 \). Then, applying the rule \texttt{sign} on \( \pi_L \) and \( \pi_R \), we obtain a simple proof of \( T_i \vdash \text{sign}(u_0, v_0) \). Indeed, this proof is left-minimal and in normal form.

\( \square \)

Appendix B. About the simplification rules

Lemma 3.5. 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;

• \( \text{Sol}(\mathcal{D}') \subseteq \text{Sol}(\mathcal{D}) \) and \( \text{Sol}_{\text{NC}}(\mathcal{D}) \subseteq \text{Sol}_{\text{NC}}(\mathcal{D}') \).

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. \( S_{\text{ax}} \)), it is clear that this constraint does not introduce any new variable. The transformation performed by applying \( S_{\text{bd}} \) also preserves origination.

Now, we have to check that the condition on variables stated in Definition 3.1 holds. Let \( x \in \text{vars}(\mathcal{D}') \). We have that \( x \in \text{vars}(\mathcal{D}) \). Let \( T_x \) be the set of terms that introduces \( x \) in \( \mathcal{D} \). If there exists \( T_x \vdash u \in \mathcal{D} \) with \( x \in \text{vars}(u) \), then this constraint still exists in \( \mathcal{D}' \) and we easily conclude. Otherwise, there exists \( v \in \mathcal{D} \) with \( x \in \text{vars}(u) \) and \( T_y \subseteq T_x \) for every \( y \in \text{vars}(v) \). The only case where we have to pay attention is the case of the rule \( S_{\text{bd}} \). However, since \( \text{vars}(u) \subseteq \text{vars}(\text{blind}(u, v)) \), we easily conclude.
We consider each simplification rule in turn. In case of $S_{ax}$ both inclusions follow immediately from the semantics of $\mathcal{E}$. Now, for the other simplification rules, it is easy to see that $\text{Sol}(\mathcal{D}') \subseteq \text{Sol}(\mathcal{D})$. For the other inclusion $\text{Sol}_{NC}(\mathcal{D}) \subseteq \text{Sol}_{NC}(\mathcal{D}')$, we have to rely on the fact that the solution $\sigma$ we consider is non-confusing. In case of $S_{bd}$ (the case of the rule $S_t$ is similar), since $\text{blind}(u, v) \neq w$, we have also that $\text{blind}(u, v) \neq w$. Since $\text{blind}(u, v) \in \mathcal{B}(T_\sigma, w)$, we necessarily have that $u \in \mathcal{B}(T_\sigma, w)$ and $T_\sigma \vdash v$. This allows us to conclude that $\sigma \in \text{Sol}(\mathcal{D}')$. Since the simplification rules do not introduce subterms, we deduce that $\sigma \in \text{Sol}_{NC}(\mathcal{D}')$. In case of $S_{cycle}$, by relying on the fact that $\sigma$ is non-confusing, we will obtain a contradiction. Thus we will deduce that $\text{Sol}_{NC}(\mathcal{D}) = \emptyset$.

\[\square\]

Appendix C. Well-formedness

Lemma 5.4 (property of $\leq_{\mathcal{D}}$). Let $\mathcal{D}$ and $\mathcal{D}'$ be two sets of constraints in simplified form such that $\mathcal{D} \leadsto \mathcal{D}'$ 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 $R_{ax}$, $R_{bd}$, $R_{t}$, $R_{b}$ and $R_{f}$ are easy since they do not affect membership constraints. The rule $R_{bd}$ 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:

- $R_{A}$: $\mathcal{D}_0 \land T \parallel x \land x \in \mathcal{B}(T', v) \leadsto \mathcal{D}_0 \land T \parallel x \land T \vdash v \land x \in \mathcal{B}(T, v)$ with $T \subseteq T'$. In this case, the ordering is not affected. We have that $\leq_{\mathcal{D}} = \leq_{\mathcal{D}'}$.

- $R_{B}$: $\mathcal{D}_0 \land T \parallel x \land x \in \mathcal{B}(T', v) \leadsto \mathcal{D}_0 \land T \parallel x \land T \vdash w \land x \in \mathcal{B}(T, w) \land w \in \mathcal{B}(T', v)$ with $T \subseteq T'$ and $w \in st(T)$. We have to show that $y \leq_{\mathcal{D}'} x$ for every $y \in \text{vars}(v)$. The simplification rules applied to $w \in \mathcal{B}(T', v)$ lead to a membership constraint of the form $z \in \mathcal{B}(T', v)$ with $z \in \text{vars}(w)$ or disappears only if $v \in \text{st}(w)$. In both cases, we easily conclude.

- $R_{C}$: $\mathcal{D}_0 \land T \parallel x \land x \in \mathcal{B}(T, v) \land x \in \mathcal{B}(T, v') \leadsto \mathcal{D}_0 \land T \parallel x \land x \in \mathcal{B}(T, v) \land v \in \mathcal{B}(T, v')$ with $T_x = T$. We have to show that $y \leq_{\mathcal{D}'} x$ for every $y \in \text{vars}(v')$. The simplification rules applied to $v \in \mathcal{B}(T, v')$ lead to a membership constraint of the form $z \in \mathcal{B}(T, v')$ with $z \in \text{vars}(v)$ or disappears only if $v' \in \text{st}(v)$. Again, in both cases, we easily conclude.

Proposition 5.5. Let $\mathcal{D}$ and $\mathcal{D}'$ be two sets of constraints in simplified form such that $\mathcal{D} \leadsto \mathcal{D}'$, and $V \subseteq \text{vars}(\mathcal{D})$. We have that $\mathcal{D}'^V \models \mathcal{D}'_V$.

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

$\mathcal{E}'^V \overset{\text{def}}{=} \{ C \mid C \in \mathcal{D}'' \land \text{vars}(C) \leq_{\mathcal{D}'} V \} \models \{ C \mid C \in \mathcal{D} \land \text{vars}(C) \leq_{\mathcal{D}} V \} (= \mathcal{D}'_V)$. 
Thanks to Lemma 5.4, we know that $\leq_D \subseteq \leq_{D'}$ and thus $\leq_D \subseteq \leq_{D''}$.

- Rule $R_{\text{ax}}$: $D_0 \land T \updownarrow^? u \leadsto D_0$. Let $C = T \uparrow^? u$. If $C \in D_0^V$ then $D_0^V \leadsto E' \subseteq E''$. Otherwise, if $C \notin D_0^V$, we have that $D_0^V \subseteq E''$. In both cases, we have that $E'' \models D_0^V$ (Lemma 5.2).

- Rule $R_{\text{triv}}$: $D_0 \land T \uparrow^? x \land T' \uparrow^? x \leadsto D_0 \land T \uparrow^? x$. Let $C = T' \uparrow^? x$. If $C \in D_0^V$ then $T \uparrow^? x$ is also in $D_0^V$ and we have that $D_0^V \leadsto E' \subseteq E''$. Otherwise, if $C \notin D_0^V$, we have that $D_0^V \subseteq E''$. In both cases, we have that $E'' \models D_0^V$ (Lemma 5.2).

- Rule $R_f$: $D_0 \land T \uparrow^? f(t_1, \ldots, t_n) \leadsto D_0 \land T \uparrow^? t_1 \land \ldots \land T \uparrow^? t_n$. Let $C = T \uparrow^? f(t_1, \ldots, t_n)$ and $C_i = T \uparrow^? t_i$ for $1 \leq i \leq n$. If $C \in D_0^V$ then $C_i \in E''$ for each $1 \leq i \leq n$, and thus $D_0^V \leadsto E' \subseteq E''$. Otherwise, if $C \notin D_0^V$, we have that $D_0^V \subseteq E''$. In both cases, we have that $E'' \models D_0^V$.

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

- Rule $R_{\text{bdcsgn}}$: $D_0 \land T \uparrow^? \text{sign}(v, u) \leadsto D_0 \land T \uparrow^? \text{sign}(w, u) \land w \in Bd(T, v)$ with $\text{sign}(w, u) \in st(T)$. Let $C = T \uparrow^? \text{sign}(v, u)$. If $C \in D_0^V$ then $T \uparrow^? \text{sign}(w, u)$ and $w \in Bd(T, v)$ are in $E''$ and thus $D_0^V \leadsto E' \subseteq E''$. Otherwise, if $C \notin D_0^V$, we have that $D_0^V \subseteq E''$. In both cases, we have that $E'' \models D_0^V$.

- Rule $R_{\text{g}}$: $D_0 \land T \uparrow^? x \land x \in Bd(T', v) \leadsto D_0 \land T \uparrow^? x \land T' \uparrow^? v \land x \in Bd(T, v)$ with $T \subseteq T'$. Either $\text{vars}(T' \cup \{v, x\}) \prec_{D'} V$ and then $\text{vars}(T' \cup \{v, x\}) \prec_{D'} V$ (thanks to Lemma 5.4) and thus $T \uparrow^? x$, $T \uparrow^? v$ and $x \in Bd(T, v)$ are in $E''$. We have that $D_0^V \leadsto E' \subseteq E''$. Otherwise, we have that $D_0^V \subseteq E''$. In both cases, we have that $E'' \models D_0^V$.

- Rule $R_{\text{r}}$: $D_0 \land T \uparrow^? x \land x \in Bd(T', v) \leadsto D_0 \land T \uparrow^? x \land T' \uparrow^? w \land x \in Bd(T, w) \land w \in Bd(T', v)$ with $T \subseteq T'$ and $w \in st(T)$. Either $\text{vars}(T' \cup \{v, x\}) \prec_{D'} V$ and then $\text{vars}(T' \cup \{v, x\}) \prec_{D'} V$ (thanks to Lemma 5.4) and thus $T \uparrow^? x$, $T \uparrow^? w$, $x \in Bd(T, w)$ and $w \in Bd(T', v)$ are in $E''$. We have that $D_0^V \leadsto E' \subseteq E''$. Otherwise, we have that $D_0^V \subseteq E''$. In both cases, we have that $E'' \models D_0^V$.

- Rule $R_{\text{c}}$: $D_0 \land T \uparrow^? x \land x \in Bd(T, v) \land x \in Bd(T, v') \leadsto D_0 \land T \uparrow^? x \land x \in Bd(T, v) \land v \in Bd(T, v')$ with $T_x = T$. First note that $\text{vars}\{v, v'\} \leq_x T$. Therefore we have that the three constraints $x \in Bd(T, v)$, $x \in Bd(T, v')$, and $T \uparrow^? x$ are either all in $D_0^V$ or none of them are in $D_0^V$. In the first case, we have that $D_0^V \leadsto E' \subseteq E''$. Otherwise we have that $D_0^V \subseteq E''$. In both cases, we have that $E'' \models D_0^V$.

Let $V' = \{x \mid x \leq_{D'} y \text{ for some } y \in V\}$. Thanks to Corollary 3.7 and since $D'' \to^* D'$, we have that $D''|_{V'} = D'|_{V'} (= E'')$. This allows us to conclude that $D'' \models D''|_{V'}$. $\square$
APPENDIX D. COMPLETENESS

Proposition 5.9 (completeness - deducibility constraint). Let \( \mathcal{C}_{\downarrow S} \) be an unsolved constraint system such that \( \mathcal{M}_{\mathcal{C}_{\downarrow S}} \) contains a deducibility constraint. Let \( \sigma \in \text{Sol}_{NC}(\mathcal{C}_{\downarrow S}) \). There exists a constraint system \( \mathcal{C}' \) such that \( \mathcal{C}_{\downarrow S} \rightsquigarrow \mathcal{C}', \sigma \in \text{Sol}_{NC}(\mathcal{C}') \) and \( \mu(\mathcal{C}' \sigma) < \mu(\mathcal{C}_{\downarrow S} \sigma) \).

Proof. Let \( T \vdash u \) be a deducibility constraint in \( \mathcal{M}_{\mathcal{C}_{\downarrow S}} \). 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 \( \text{R}_{\text{triv}} \) could be applied. We have that \( \sigma \in \text{Sol}_{NC}(\mathcal{C}') \) 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 \( \sigma \) is non-confusing. Hence, we can apply \( \text{R}_{\text{ax}} \). Clearly, we have that \( \sigma \in \text{Sol}_{NC}(\mathcal{C}') \) and \( \mu(\mathcal{C}' \sigma) < \mu(\mathcal{C}_{\downarrow S} \sigma) \).

Now, if \( P \) ends with an instance of the rule \( \text{unblind} \), we have that there exists \( w \) such that the direct subproof \( P_1 \) (resp. \( P_2 \)) of \( P \) is labeled with \( T \vdash \text{blind}(u \sigma, w) \) (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 5.8, we deduce that \( \text{blind}(u \sigma, w) \in (st(T) \setminus X)\sigma \). Hence, by relying on the fact that \( \sigma \) is non-confusing, there exists \( \text{blind}(u, w') \in st(T) \) such that \( w' \sigma = w \). Hence, we can apply \( \text{R}_{\text{bd}} \). Let \( \mathcal{C}' \) be the constraint system obtained after application of the transformation rule. Note that \( \sigma \in \text{Sol}_{NC}(\mathcal{C}') \) 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 \( \sigma \) is a solution of \( \mathcal{C}' \). The case where \( P \) ends with an instance of \( \text{getmsg} \) is similar.

- \( P \) ends with an instance of a composition rule. If \( P \) ends with an instance of \( \text{sign} \) (resp. \( \text{blind} \)), we have that \( u = \text{sign}(u_1, u_2) \) (resp. \( u = \text{blind}(u_1, u_2) \)). In such a case, we can apply the transformation rule \( \text{R}_f \), and we easily conclude. Now, it remains to deal with the case where \( P \) ends with an instance of \( \text{unblind} \). In such a case, we have that \( u = \text{sign}(u_1, u_2) \). Let \( P_0, P_1, \ldots, P_k \) be the direct subtrees of \( P \). We have that \( P_1 \) is labeled with \( T \sigma \vdash \text{sign}(w, u_2 \sigma) \). Moreover, note that \( P_0 \) is normal and therefore does not end with a composition rule. Hence, thanks to Lemma 5.8, we have that \( \text{sign}(w, u_2 \sigma) \in (st(T) \setminus X)\sigma \), i.e., by using the fact that \( \sigma \) is non-confusing, there exists \( \text{sign}(w', u_2) \in st(T) \) such that \( w' \sigma = w \). We deduce that the rule \( \text{R}_{\text{bdsign}} \) can be applied. Let \( \mathcal{C}' \) be the resulting constraint system. We have that \( \sigma \in \text{Sol}_{NC}(\mathcal{C}') \) and \( \mu(\mathcal{C}' \sigma) < \mu(\mathcal{C}_{\downarrow S} \sigma) \).

Proposition 5.10 (completeness - membership constraint). Let \( \mathcal{C}_{\downarrow S} \) be an unsolved constraint system such that \( \mathcal{M}_{\mathcal{C}_{\downarrow S}} \) only contains membership constraints. Let \( \sigma \in \text{Sol}_{NC}(\mathcal{C}_{\downarrow S}) \). There exists a constraint system \( \mathcal{C}' \) such that \( \mathcal{C}_{\downarrow S} \rightsquigarrow \mathcal{C}', \sigma \in \text{Sol}_{NC}(\mathcal{C}') \) 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 \( x \in \mathcal{Bd}(T, u) \). Let us consider the set of variables \( V_M = \{ x \mid x \in \mathcal{Bd}(T', u) \in \mathcal{M}_{\mathcal{C}_{\downarrow S}} \} \).

Let us choose a constraint \( x \in \mathcal{Bd}(T', u) \in \mathcal{M}_{\mathcal{C}_{\downarrow S}} \) 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 $T \vdash x$ that occurs in $C \downarrow \mathcal{S}$ 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 $T \vdash x$ and $x \in \text{Bd}(T', u)$ are in $C \downarrow \mathcal{S}$ and $T \subseteq T'$. We will show that we can apply $\mathcal{R}_A$ or $\mathcal{R}_B$. We have that $x \sigma = b_k(u \sigma, v_1, \ldots, v_k)$ with $T' \sigma \vdash v_1, \ldots, T' \sigma \vdash v_k$. Note that $k > 0$ since $\sigma$ is non-confusing. Depending on whether all proofs of $T' \sigma \vdash v_i$ can be weakened or not, we apply either $\mathcal{R}_A$ or $\mathcal{R}_B$.
  
  Assume that $T \sigma \vdash v_i$ for every $1 \leq i \leq k$. Let $C'$ be the constraint system obtained by applying the rule $\mathcal{R}_A$. 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 $\text{unblind}$ 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 $\sigma \in \text{Sol}(C')$, and since no subterm is introduced, we have that $\sigma \in \text{Sol}_{\text{NC}}(C')$. Lastly, we have that $\mu(C' \sigma) < \mu(C \downarrow \mathcal{S} \sigma)$ 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 $T \sigma \subseteq T' \sigma$ (this strict inclusion is due to the fact that $T \subsetneq T'$ and $\sigma$ non-confusing).

  Otherwise, let $i_0$ be such that $T \sigma \vdash v_j$ for each $j > i_0$ and $T \sigma \not\vdash v_{i_0}$. 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 5.8, we deduce that there exists $t \in \text{st}(T) \setminus X$ such that $t \sigma = x \sigma$. Since $\sigma$ 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 $T \sigma \vdash \text{blind}(\ldots \text{blind}(u \sigma, v_1), \ldots, v_{i_0})$ by taking the proof of $T \sigma \vdash x \sigma$, the proofs of $T \sigma \vdash v_{i_0}, \ldots, T \sigma \vdash v_{i_0+1}$ and applying $k - i_0$ times the rule $\text{unblind}$. Let $P$ be a simple proof of $T \sigma \vdash b_{i_0}(u \sigma, v_1, \ldots, v_{i_0})$. Now, since $T \sigma \not\vdash v_{i_0}, P$ can not end with a composition rule. Hence, by Lemma 5.8, we deduce that there exists $w \in (\text{st}(T) \setminus X)$ such that $w \sigma = b_{i_0}(u \sigma, v_1, \ldots, v_{i_0})$, implying that $w \sigma \in \text{Bd}(T' \sigma, u \sigma)$. Therefore, we can apply the rule $\mathcal{R}_B$ to get the constraint system $C'$. We have seen that $\sigma$ satisfies each constraint that has been added in $C'$, hence $\sigma \in \text{Sol}(C')$ and it is easy to see that actually $\sigma \in \text{Sol}_{\text{NC}}(C')$. Lastly, we have that $\mu(C' \sigma) < \mu(C \downarrow \mathcal{S} \sigma)$ 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 $T \sigma \subseteq T' \sigma$ (this strict inclusion is due to the fact that $T \subsetneq T'$ and $\sigma$ non-confusing).

- Case $T = T'$. In such a case, since $x \in \text{Bd}(T', u) \in M_{C \downarrow \mathcal{S}}$, we must have that there exists another membership constraint $x \in \text{Bd}(T'', u)$. 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

  - $x \sigma = b_k(u \sigma, v_1, \ldots, v_k)$ with $T \sigma \vdash v_i$ for $1 \leq i \leq k$; and
  - $x \sigma = b_p(u \sigma, v'_1, \ldots, v'_p)$ with $T \sigma \vdash v'_i$ for $1 \leq i \leq p$.

  Now, clearly, we have that either $u \sigma \in \text{st}(u \sigma)$ and $w \sigma \in \text{Bd}(T \sigma, u \sigma)$ or symmetrically $w \sigma \in \text{st}(w \sigma)$ and $u \sigma \in \text{Bd}(T \sigma, w \sigma)$. 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 $\mathcal{R}_C$ to
get a constraint system $C' = C \setminus \{x \in \text{Bd}(T, u)\} \cup \{w \in \text{Bd}(T, u)\}$. It is clear that $
abla_{\text{NC}}(C')$. Now, we have to show that the sequence of proof trees witnessing $w\sigma \in \text{Bd}(T\sigma, u\sigma)$ is strictly smaller than the sequence witnessing $x\sigma \in \text{Bd}(T\sigma, u\sigma)$. This is due to the fact that $w\sigma$ is a strict subterm of $x\sigma$. Indeed, we have that $w\sigma \in \text{st}(x\sigma)$ and $w\sigma = x\sigma$ is not possible since otherwise, we would have $w = x$ and this would contradict the fact that $C_{\text{S}}$ is simplified.

\[\square\]