Toward an Automatic Analysis of Web Service Security

Yannick Chevalier, Denis Lugiez, Michael Rusinowitch

INRIA

Toward an Automatic Analysis of Web Service Security

Yannick Chevalier — Denis Lugiez   — Michaël Rusinowitch

N° 6341

Thème SYM

Rapport de recherche

Toward an Automatic Analysis of Web Service Security

Yannick Chevalier[*][†] , Denis Lugiez[‡] [§] , Michaël Rusinowitch[¶][‖]

Abstract: Web services send and receive messages in XML syntax with some parts hashed, encrypted or signed, according to the WS-Security standard. In this paper we introduce a model to formally describe the protocols that underly these services, their security properties and the rewriting attacks they might be subject to. Unlike with usual security protocols, we have to address here the facts that: (1) The Web service receive/send actions are non-deterministic to accommodate the XML format and the lack of normalization in parsing XML messages. Our model is designed to permit non-deterministic operations. (2) The Web service message format is better modelled with multiset constructors than with fixed arity symbols. Hence we had to introduce an attacker model that handles associative-commutative operators. In particular we present a decision procedure for insecurity of Web services with messages built using encryption, signature, and other cryptographic primitives.

Key-words: Security, web services, cryptographic protocols, combination of decision procedures, equational theories, rewriting.

Vers la vérification automatique de la sécurité des services web

Résumé :

Les services Web envoient et reçoivent des messages en format XML en utilisant les protocoles SOAP. Les recommandations de sécurité du consortium W3C autorisent des parties du message à être signées, cryptées,... Dans ce rapport nous décrivons un modèle pour décrire formellement les protocoles implémentés par ces services leurs propriétés de sécurité ainsi que les attaques base de réécriture dont ils font l'objet. Contrairement aux protocoles cryptographiques usuels, nous devons prendre en compte deux particularités:

(1) les actions de réception/envoi des services Web sont non-déterministes afin de refléter le format XML et le manque de normalisation pour l'analyse de des messages SOAP. Notre modèle est suffisamment souple pour traiter le non-déterminisme et permet de décrire précisement les actions que réalise chaque participant.

(2) Le format des message est mieux modélisé comme un multi-ensemble de termes plutôt que comme un terme d'arité fixe. Par conséquent nous devons définir un modèle d'attaquant plus puissant que ceux qui existent actuellement: en particulier nous donnons une procédure de décision pour les protocoles non-déterministes utilisant les multi-ensembles, qui peut être combinée avec la plupart des opérateurs algébriques pour lesquels on connait une procédure de décision pour la propriété de secret (xor, etc...).

Mots-clés : Protocoles cryptographiques, combinaison de procédures de décision, théories équationnelles

1   Introduction

Web services promise to be a standard technology for Internet and enterprise networks. They require the ability to transmit securely messages in XML syntax using SOAP protocol. Messages that travel over the networks can be observed and modified by intruders. Hence the protocol was extended by W3C for allowing to sign and encrypt some parts of the contents. Nevertheless, as for classical protocols, cryptographic mechanisms are not sufficient for securing Web services. Not only they might be subject to the same attacks (e.g. man-in-the middle) as classical cryptographic protocols, but the XML syntax and the specific way messages are processed (e.g. not examining the full content) gives the opportunity to mount new class of attacks, such as XML rewriting attacks as shown in [4].

Recently many decision procedures have been proposed for analysing cryptographic protocols. These procedures rely on automated deduction and constraint solving procedures extending semantic unification. Our objective in this work is to investigate whether these results can be lifted to Web services. First it appears immediately that the flexible XML format requires to consider message contents as sets of terms rather than terms: this introduces a first difficulty since no security decidability results exist yet for protocols using such a data structure. Second, since messages are only partially parsed by SOAP, the answer to a request might be multiple depending on the implementation: in other words we have to model a non-deterministic behaviour of the Web service protocols.

A role-based protocol model has been proposed in previous works [5]. This model admits two versions. In the first one, the *equational model*, operations (including decryption) are explicit and the basic operations are deterministic. We believe that this constraint is inherent to the model, as non-determinism in equational theories leads to incoherence. The second model, the *pattern-matching model*, relies on patterns to filter incoming messages. It has been widely studied but proofs in this model are often complex and obscure, and there is no general combination result that would permit one to extend existing decision procedures to a new operation (the multiset operator, in our case).

Motivated by these facts, and also for the sake of deriving a uniform framework, we propose a protocol model that is more general than the two commonly used ones. It allows to manipulate XML messages as multisets of terms and can simulate non-deterministic answers. Moreover decidability and combination results (previous known only for the equational model [9]) have been lifted to this more general model. The proofs are involved and do not follow from the ones in simpler models.

Related works.   The most advanced project in the area of Web services and formal verification is Samoa Project [4]. This project offers a language to express SOAP-based security protocols and a compiler from this language to the applied pi calculus on which ProVerif resolution-based system can be run to verify secrecy and authentication properties. Our approach can be viewed as a complementary one since it provides decision procedures that are complete for finitely many sessions, even with an associative-commutative XML message constructor (and also other operators such as Dolev Yao encryption/decryption). On the

protocol side, a lot of work has been dedicated to security analysis modulo algebraic properties (see e.g. [3, 11, 14]). However no decidability results have been reported for associative or associative-commutative operators, supporting the combination with other useful cryptographic primitives (such as XOR). To our knowledge, the ones we report here are the first of this kind, and we show how they apply to deciding security properties of XML services. These results rely on intensive use of term rewriting and unification techniques [18, 2].

Paper organization.   First, we describe an example of XML-rewriting attacks in Section 2, then we recall the notions of term rewriting we use on our model in Section 3. Section 4 formally defines the deductions that can be performed by intruders and honest agents in a protocol. We introduce *symbolic derivations* as parameterized deductions. They can be viewed as a variant of *strand spaces*, a fundamental concept in protocol analysis [20]. Then we show how to combine deductions associated to different operators in Section 9. Finally we apply in Section 10 the combination scheme to a multi-set of terms constructor, and show how it provides us with a decidable analysis of XML protocols.

2   An example of Web Service and XML rewriting attack

Web services can be described as a set of receive/send action between clients and the provider, where the messages are XML terms following the SOAP format. Messages are constructed and parsed according to the security policy of the service which usually requires that some parts of the message are encrypted, signed using known algorithms like sha1, rsa,..., may contain timestamps, make reference to trusted third party,.... A SOAP message is an envelope consisting of a mandatory body and of an optional header. The header contains the useful informations about the message and usually has a security part that follows the WS-security specification defined by the OASIS organization. For instance Alice may send the following message to Bob, her correspondent in the travel agency:

```
<Envelope>
  <Header>
    <To> http://www.travel-for-free.com/~bob </>
    <Security>
        <timestamp>2007-04-16T09:56:43Z</>
        <signature>
          <signedInfo>
            <reference URI=#1>a string</>
            <signatureValue>another string </>
    </></></></>
    <Body>
      <Order id=1>
        <beneficiary>Alice</>
        <account>Alice</>
        <trip>Bremen</>
    </></></>
```

that orders a ticket for Bremen, for herself, and to be charged to her account. In the security header, the first string is a digest of the body, and the other string is the encryption of this digest using the private key of Alice. For efficiency purpose, only parts of the messages are encrypted or signed and we model the fact that a node labelled by some tag has an arbitrary number of elements by introducing a unary symbol $a(\_)$ for each tag 'a' and an associative-commutative multiset constructor. For instance, the body of the above XML message corresponds to the term $Body(Order(beneficiary(Alice) \cdot account(Alice) \cdot trip(Bremen)))$ where $e_1 \cdot \ldots \cdot e_n$ denotes the multiset $\{e_1, \ldots, e_n\}$. Another possibility is to replace the multiset operator by an associative operator for sequences. In Web services, the component of messages are selected by their tag and security policies usually refer to this name and don't consider the possibility of multiple occurrences of the same tag.

In our example, the recipient Bob, performs several security checks, including the verification of the signature, then looks for a part of the *Body* labelled *Order*, orders a ticket for the requested trip and charges Alice's account. Then, he sends to Alice a SOAP message with a security header containing the digest of the original message and a body containing an access code for the trip encoded by the public key of the beneficiary (that Alice will transfer to the beneficiary). Like in the study of cryptographic protocols, we assume that cryptography is perfect, but we don't make any particular hypothesis on the implementation of parsing XML trees. An attacker Charlie may intercept the message and forge a new one that inserts a new *Order* item to the *Body*:

```
<Order id=2>
  <beneficiary>Charlie </>
  <account>Alice</>
  <trip> Hawaii</>
</>
```

just before the previous *Order*. An implementation of the service may lead to a successful verification of the signature and to sending the accesscode to Charlie depending on how the XML term is parsed. If the first match of *Order* is chosen, then Charlie gets the accesscode, since the signature is correct because the reference to the initial *Order* is still used. But another implementation could parse the children of a node from right to left and select the correct *Order* subterm. Therefore the behavior of the protocol is non-deterministic which we model by rules that select a element *Order(_)* in a multiset. To describe the behaviour of the service, we model all the operations that the participants can do and all algebraic relations that may hold between the operators, which is done by the means of deduction rules and equations. A possible abstraction of the security protocol $\mathcal{P}$ that supports the service is:

$$A \rightarrow B : \quad \text{se}(h(order(x, y, z)), SK_A) \cdot order(x, y, z)$$
$$B \rightarrow A : \quad \text{se}(h(order(x, y, z)), PK_A) \cdot \text{se}(accesscode(z), PK_x)$$

where h denotes a hashing function, $order(x, y, z)$ denotes the request for trip z for beneficiary x, with y the account to charge, h is a hash function, *accesscode(z)* is the code requested to get the ticket from automata, $\text{se}(x, y)$ denotes the encryption of x using key y, PK_x is the public key of x, SK_x is the private key of x. The intruder deductive power is given by the classical Dolev-Yao rules (see [13, 21]) extended by a rule that allows to select any element in a multiset. For some realistic implementations of the service as the one that is described in Section 4 the following attack will be possible:

$$A \rightarrow I_B : \quad \text{se}(h(order(A, A, Bremen)), SK_A) \cdot order(A, A, Bremen)$$
$$I_A \rightarrow B : \quad \text{se}(h(order(A, A, Bremen)), SK_A) \cdot order(C, A, Hawaii) \cdot order(A, A, Bremen)$$
$$B \rightarrow I_A : \quad \text{se}(h(order(A, A, Bremen)), PK_A) \cdot \text{se}(accesscode(Hawaii), PK_C)$$

where I_A (resp. I_B) denotes a malicious agent I masquerading the honest participant A (resp. B), and the secret key SK_C is known by I ($C = I$ or C has been compromised).

3   Terms, subterms and ordered rewriting

3.1   Basic notions

We consider an infinite set of free constants C and an infinite set of variables $\mathcal{X}$. For all signatures $\mathcal{F}$ (*i.e.* a set of function symbols with arities), we denote by $T(\mathcal{F})$ (resp. $T(\mathcal{F}, \mathcal{X})$) the set of terms over $\mathcal{F} \cup C$ (resp. $\mathcal{F} \cup C \cup \mathcal{X}$). The former is called the set of ground terms over $\mathcal{F}$, while the later is simply called the set of terms over $\mathcal{F}$. Variables are denoted by x, y, terms are denoted by s, t, u, v, and finite sets of terms are written $E, F, ...$, and decorations thereof, respectively. For finite sets of terms, we abbreviate $E \cup F$ by E, F, the union $E \cup \{t\}$ by E, t and $E \setminus \{t\}$ by $E \setminus t$.

Given a signature $\mathcal{F}$ a *constant* is either a free constant in C or a function symbol of arity 0 in $\mathcal{F}$. Given a term t we denote by $\mathrm{Var}(t)$ the set of variables occurring in t and by $\mathrm{Cons}(t)$ the set of constants occurring in t. An *atom* is either a variable or a constant and we denote by $\mathrm{Atoms}(t)$ the set $\mathrm{Var}(t) \cup \mathrm{Cons}(t)$.

A substitution σ is an involutive mapping from $\mathcal{X}$ to $\mathrm{T}(\mathcal{F}, \mathcal{X})$ such that $\mathrm{Supp}(\sigma) = \{x \,|\, \sigma(x) \neq x\}$, the *support* of σ, is a finite set. The application of a substitution σ to a term t (resp. a set of terms E) is denoted $t\sigma$ (resp. $E\sigma$) and is equal to the term t (resp. the terms in E) where all variables x have been replaced by the term $x\sigma$. A substitution σ is *ground* w.r.t. a signature $\mathcal{F}$ and a set of variables V if the image of $\mathrm{Supp}(\sigma)$ is included in $\mathrm{T}(\mathcal{F})$ and $V \subseteq \mathrm{Supp}(\sigma)$. We simply say that a substitution σ is ground if $\mathcal{F}$ and V are clear from the context.

An *equational presentation* $\mathcal{H} = (\mathcal{F}, A)$ is defined by a set A of equations $r = t$ with $r, t \in \mathrm{T}(\mathcal{F}, \mathcal{X})$. For any equational presentation $\mathcal{H}$ the relation $=_{\mathcal{H}}$ denotes the equational theory generated by $(\mathcal{F}, A)$ on $\mathrm{T}(\mathcal{F}, \mathcal{X})$, that is the smallest congruence containing all instances of axioms of A. We shall not distinguish between an equational presentation $\mathcal{H}$ over a signature $\mathcal{F}$ and a set A of equations presenting it and denote both $\mathcal{H}$. We will also often refer to $\mathcal{H}$ as an equational theory (meaning the equational theory presented by $\mathcal{H}$). An equational theory $\mathcal{H}$ is *consistent* if there exists at least one model of $\mathcal{H}$ with more than one element. Equivalently a theory $\mathcal{H}$ is consistent if there does not exist two free constants x and y such that $x \neq y$ and $x =_{\mathcal{H}} y$.

The *syntactic subterms* of a term t are denoted $\mathrm{Sub}_{\mathrm{syn}}(t)$ and are defined recursively as follows. If t is an atom then $\mathrm{Sub}_{\mathrm{syn}}(t) = \{t\}$ otherwise, let $t = f(t_1, \ldots, t_n)$ and $\mathrm{Sub}_{\mathrm{syn}}(t) = \{t\} \cup \bigcup_{i=1}^{n} \mathrm{Sub}_{\mathrm{syn}}(t_i)$. The *positions* in a term t are sequences of integers defined recursively as follows, ϵ being the empty sequence. The term t is at position ϵ in t. If u is a syntactic subterm of t at position p and if $u = f(u_1, \ldots, u_n)$ then u_i is at position $p \cdot i$ in t for $i \in \{1, \ldots, n\}$. The replacement of the subterm of t at position p is denoted by $t[p \leftarrow s]$, for a set of positions Π, $t[\Pi \leftarrow s]$ denotes the replacement of all subterms at position $p \in \Pi$ by s.

3.2 Congruences and ordered rewriting

Ordered rewriting [12] is a useful tool that has been used (*e.g.* [2]) to prove the correctness of combination of unification algorithms.

Let $<$ be a simplification ordering on $\mathrm{T}(\mathcal{F})$ [1] assumed to be total on $\mathrm{T}(\mathcal{F})$ and such that (i) the minimal element for $<$ is a constant $c_{\mathrm{min}} \in C$; (ii) each non-free constant is smaller than any non-constant ground term.

We denote by C_{spe} the set containing the constants in $\mathcal{F}$ and c_{min}.

Given a possibly infinite set of equations $\mathcal{O}$ on the signature $\mathrm{T}(\mathcal{F})$, the ordered rewriting relation $\rightarrow_{\mathcal{O}}$ is defined by $t \rightarrow_{\mathcal{O}} t'$ iff there exists a position p in t, an equation $l = r$ in $\mathcal{O}$ and a substitution τ such that $t = t[p \leftarrow l\tau]$, $t' = t[p \leftarrow r\tau]$, and $l\tau > r\tau$. Since $<$ is monotonic, note that this last condition implies $t' < t$. It has been shown (see [12]) that applying

---

[1]by definition $<$ satisfies for all $s, t, u \in \mathrm{T}(\mathcal{F})$ $s < t[s]$ and $s < u$ implies $t[s] < t[u]$

the *unfailing completion procedure* to a set of equations $\mathcal{H}$ yields a (possibly infinite) set of equations $\mathcal{O}$ such that: (i) the congruence relations $=_\mathcal{O}$ and $=_\mathcal{H}$ are equal on $\mathrm{T}(\mathcal{F})$. (ii) the ordered rewrite relation $\rightarrow_\mathcal{O}$ is convergent (*i.e.* terminating and confluent) on $\mathrm{T}(\mathcal{F})$.

We shall say that $\mathcal{O}$ is an *o-completion* of $\mathcal{H}$. Since the rewrite system $\rightarrow_\mathcal{O}$ is convergent on ground terms, we can define $(t){\downarrow}_\mathcal{O}$ as the unique normal form of the ground term t for $\rightarrow_\mathcal{O}$. A ground term t is in *normal form*, or *normalised*, if $t = (t){\downarrow}_\mathcal{O}$. Given a ground substitution σ we denote by $(\sigma){\downarrow}_\mathcal{O}$ the substitution with the same support such that for all variables $x \in \mathrm{Supp}(\sigma)$ we have $x(\sigma){\downarrow}_\mathcal{O} = (x\sigma){\downarrow}_\mathcal{O}$. A substitution σ is *normal* if $\sigma = (\sigma){\downarrow}_\mathcal{O}$.

The proofs of the next sequence of lemmas can be found in [10] and thus is omitted here.

Lemma 1 *Assume that $g = d \in R$, $z \in \mathrm{Var}(d) \setminus \mathrm{Var}(g)$, and $s \rightarrow_R s'$ with $s = s[p \leftarrow g\tau]$, $s' = s[p \leftarrow d\tau]$, $g\tau > d\tau$. Let us define the substitution σ such that for all variables $x \neq z$ we have $x\sigma = x\tau$ and $z\sigma = c_{\min}$. Then we have also: $s \rightarrow_R s[p \leftarrow d\sigma]$*

Lemma 2 *If $\mathcal{H}$ is a consistent equational theory then for any equation $g = d$ in a presentation of $\mathcal{H}$ with $g \neq d$ if there exists a substitution τ such that $g\tau > d\tau$ then g is not a variable.*

3.3   Unification problems

We define when a substitution σ satisfies a set of equations $\mathcal{S}$.

Definition 1 *(Unification systems) Let $\mathcal{H}$ be a set of equational axioms on $\mathrm{T}(\mathcal{F}, \mathcal{X})$. An $\mathcal{H}$-Unification system $\mathcal{S}$ is a finite set of couples of terms in $\mathrm{T}(\mathcal{F}, \mathcal{X})$ denoted by $(u_i \overset{?}{=} v_i)_{i \in \{1,\ldots,n\}}$. The ground substitution σ satisfies $\mathcal{S}$, denoted by $\sigma \models \mathcal{S}$, iff for all $i \in \{1, \ldots, n\}$ $u_i\sigma =_\mathcal{H} v_i\sigma$.*

4   Modelling service execution

4.1   A model for secure Web Services

Let us first briefly review the situation we want to model. Some simple Web services are akin to functions in a library. Their execution is triggered by the reception of a request from a client, to which they immediately respond. These services are advertised by a WSDL specification that defines among other things the contents of the input and output messages. It is also possible to specify some cryptographic protection for integrity, confidentiality and authenticity of a request by means of a published security policy. This policy will constrain acceptable requests by mandating that some parts have to be signed, encrypted or integrity protected. These parts are expressed either by identifiers or by XPath expressions, and we say that these services are *protected*. Finally, some WS standards, *e.g.* BPEL, WS-SecureConversation, and others, permit to express sequences of simple service invocations, which we call *workflows*. We call totally ordered sequences *workflow executions*, as they also correspond to traces of service workflows.

The analysis of Web services is thus very similar to the one of cryptographic protocols. A protected service workflow is a composition of *roles* (client, service,...), and a workflow execution is akin to a protocol execution. *Agents* impersonate the roles in a workflow execution.

This similarity shall not, however, hide the differences at the level of messages. In the case of cryptographic protocols, the patterns of admissible messages are fixed, and known by the intruder, whereas security policies just express constraints on the presence of some particular subterms at some position in a message. Moreover, and since services are in most cases automatically generated, the verification of a message is likely to be independent of the construction of a response. Finally, some implementation may return only one node in a hedge when several ones correspond to an XPath expression, thereby allowing for XML injection attacks. The work described in this paper focuses on security policies expressing paths of subterms from the root (envelope) of the message, leaving general XPath constraints to future work.

Example 1 *A client A invokes a service B by sending* $\text{se}(h(order(x,y,z)), SK_A) \cdot order(x,y,z)$, *where x, y and z are instantiated parameters of the client. Then it receives a response r which is parsed to check that it contains the nodes* $\text{se}(z, PK_A)$ *and* $\text{se}(u,v)$ *at its root, with* $v = PK_x$ *and* $z = h(order(x,y,z))$.

The intruder and the insecurity problem. In the Dolev-Yao model [13], attacks on protocols are modelled by the addition of a malicious participant, called the intruder that controls the network. It can intercept, block and/or redirect all messages sent by honest agents. It can also masquerade its identity and take part in the protocol under the identity of an honest participant. Its control of the communication network is modelled by considering that all messages sent by honest agents are sent directly to the intruder and that all messages received by the honest agents are always sent by the intruder. Besides the control on the net, the intruder has specific rules to deduce new values and compute messages. From the intruder's point of view a finite execution of a protocol is therefore the interleaving of a finite sequence of messages it has to send and a finite sequence of messages it receives (and add to its knowledge). Therefore the intruder is simply an additional role that runs concurrently with the honest participants.

The protocol is insecure if some secret knowledge is revealed during the execution of the protocol, which is modelled by adding to the protocol a last step that reveals the secret. The insecurity problem amounts to finding a sequence of actions of the intruder such that its composition with the (extended) protocol is executable.

4.2 Deduction systems

We give a formal model for roles and the execution of roles (including the intruder).

4.2.1   Deduction rules

We model messages as ground terms and deductions as rewrite rules on sets of messages representing the knowledge of an agent. Each role derives new messages from a given (finite) set of messages by applying deduction rules. Furthermore, these derivations are considered *modulo* the equational congruence $=_{\mathcal{H}}$ generated by the equational axioms satisfied by the function symbols of the signature $\mathcal{F}$.

Definition 2 *An deduction rule is a rule $d : t_1, \ldots, t_n \rightarrow t$ with terms $t_1, \ldots, t_n, t$ such that for every ground substitution σ:*

$$\mathrm{Const}((t\sigma)\downarrow) \subseteq \cup_{i=1}^{n}\mathrm{Const}((t_i\sigma)\downarrow) \cup \mathrm{C_{spe}}$$

This condition is very similar to the *well-definedness* condition of [19, 15]. When the equational theories are regular (a.k.a. non-erasing or variable-preserving), this condition holds iff $\mathrm{Var}(t) \subseteq \mathrm{Var}(t_1, \ldots, t_n)$.

Example 2 *$x, y \rightarrow \langle x, y \rangle$ is a deduction rule since the symbol function $\langle \_, \_ \rangle$ is free. This rules allows an agent (who can be the intruder) to construct a new pair from two values already known.*

4.2.2   Deduction systems and derivations

Deduction rules are the basic ingredients for deduction systems.

Definition 3 *A deduction system $\mathcal{D}$ is a triple $\langle \mathcal{F}, \mathcal{R}, \mathcal{H} \rangle$ where $\mathcal{F}$ is a signature, $\mathcal{R}$ is a set of deduction rules and $\mathcal{H}$ is a set of equations between terms in $\mathrm{T}(\mathcal{F}, \mathcal{X})$. For each deduction rule $d : t_1, \ldots, t_n \rightarrow t \in \mathcal{R}$, the set $\mathrm{GI}(d)$ denotes the set of ground instances of the rule d modulo $\mathcal{H}$:*

$$\mathrm{GI}(d) = \{l \rightarrow r \,|\, \exists \sigma, \text{ ground substitution on } \mathcal{F}, \ l =_{\mathcal{H}} t_1\sigma, \ldots, t_n\sigma \text{ and } r =_{\mathcal{H}} t\sigma\}$$

The set of rules $\mathrm{GI}_{\mathcal{D}}$ is defined as the union of the sets $\mathrm{GI}(d)$ for all $d \in \mathcal{R}$.

Example 3 *Let $\mathcal{F} = \{\langle \_, \_ \rangle, \mathrm{se}(\_, \_), \cdot, \mathrm{c_{min}}, a, b, \ldots, n_a, n_b, \ldots, K_a, K_b, \ldots\}$, $\mathcal{R}_{\mathcal{I}} = \{x, y \rightarrow \langle x, y \rangle ; x, y \rightarrow \mathrm{se}(x, y); \mathrm{se}(x, y), y \rightarrow x; \langle x, y \rangle \rightarrow x; \langle x, y \rangle \rightarrow y; x, y \rightarrow x \cdot y, a(x_1, \ldots, x_n) \cdot y \rightarrow a(x_1, \ldots, x_n)\}$, $\mathcal{H} = \{x \cdot y = y \cdot x, x \cdot (y \cdot z) = (x \cdot y) \cdot z\}$, where $a(\_)$ stands for any n-ary symbol of $\mathcal{F}$ different from $\cdot$. Then the deduction system $\mathcal{D} = \langle \mathcal{F}, \mathcal{R}_{\mathcal{I}}, \mathcal{H} \rangle$ describes the classical Dolev-Yao model with the addition of an associative-commutative operation $\cdot$ and of the rules that allow to build multisets with $\cdot$ or to extract parts of a multiset.*

Each deduction rule $l \rightarrow r$ in $\mathrm{GI}_{\mathcal{D}}$ defines an deduction relation $\rightarrow_{l \rightarrow r}$ between finite sets of terms: given two finite sets of terms E and F we define $E \rightarrow_{l \rightarrow r} F$ if and only if $l \subseteq E$ and $F = E, r$. We denote $\rightarrow_{\mathcal{D}}$ the union of the relations $\rightarrow_{l \rightarrow r}$ for all $l \rightarrow r$ such that $l \rightarrow r \in \mathrm{GI}(d)$ for some $d \in \mathcal{R}$, and by $\rightarrow_{\mathcal{D}}^{*}$ the transitive closure of $\rightarrow_{\mathcal{D}}$. We simply denote by $\rightarrow$ the relation $\rightarrow_{\mathcal{D}}$ when there is no ambiguity about the deduction system $\mathcal{D}$.

Definition 4 *A derivation D of length $n \geq 0$, is a sequence $E_0 \rightarrow_{\mathcal{D}} E_1 \rightarrow_{\mathcal{D}} \cdots \rightarrow_{\mathcal{D}}$ E_n where $E_0, \ldots E_n$ are finite set of ground terms such that $E_i = E_{i-1}, t_i$ for every $i \in \{1, \ldots, n\}$. The term t_n is called the* goal *of the derivation.*

Example 4 *The previous deduction system has a derivation:*

$$\{K_a, \langle \mathrm{se}(s, K_a), a \rangle, a, b\} \quad \rightarrow_{\mathcal{D}} \{K_a, \langle \mathrm{se}(s, K_a), a \rangle, \mathrm{se}(s, K_a), a, b\}$$
$$\rightarrow_{\mathcal{D}} \{K_a, \langle \mathrm{se}(s, K_a), a \rangle, \mathrm{se}(s, K_a), s, a, b\}$$

that describes the discovering of a secret s by an intruder that knows the encryption key K_a, identity of agents and intercepts a message $\langle \mathrm{se}(s, K_a), a \rangle$. The ground deduction rules employed are $\langle \mathrm{se}(s, K_a), a \rangle \rightarrow \mathrm{se}(s, K_a)$ and $\mathrm{se}(s, K_a), K_a \rightarrow s$

The set of terms that can be derived from E is $\mathrm{Der}_{\mathcal{D}}(E) = \{t \mid \exists F \text{ s.t. } E \rightarrow_{\mathcal{D}}^* F \text{ and } t \in F\}$. If there is no ambiguity on the deduction system $\mathcal{D}$ we write $\mathrm{Der}(E)$ instead of $\mathrm{Der}_{\mathcal{D}}(E)$. A derivation is *without stutter* if for all $i, j \in \{1, \ldots, n\}$, $t_i \in E_j$ implies $i \geq j$.

The set of equations $\mathcal{H}$ yields an ordered rewriting relation that is used to normalize terms. Let $\mathcal{O}$ be an o-completion of $\mathcal{H}$. The next result will allow us to restrict ourselves to deductions and derivations with normalized terms:

Lemma 3 *Let E be a set of ground terms and let t be a ground term. Then there is a deduction $E \rightarrow F$ iff there is a deduction $(E)\downarrow_{\mathcal{O}} \rightarrow (F)\downarrow_{\mathcal{O}}$.*

From now on, we assume that all the deduction rules generate terms that are normalized by $\rightarrow_{\mathcal{O}}$ and that the goal and the initial set are in normal form for $\rightarrow_{\mathcal{O}}$.

4.3 Symbolic derivation

Given a deduction system $\langle \mathcal{F}, \mathcal{R}, \mathcal{E} \rangle$, a role applies rules in $\mathcal{R}$ to construct the response of step i. Moreover a role may test equalities to check the well-formedness of a message. Hence the activity of a role can be expressed by a fixed symbolic derivation:

Definition 5 *(Symbolic Derivation) A symbolic derivation for deduction system $\langle \mathcal{F}, \mathcal{R}, \mathcal{E} \rangle$ is a tuple $(\mathcal{V}, \mathcal{S}, \mathcal{K}, \mathrm{I}N, \mathrm{O}UT)$ where $\mathcal{V}$ is a sequence of variables, $(x_i)_{i \in Ind}$, indexed by a linearly ordered set $(Ind, <)$, $\mathcal{K}$ is a set of ground term (the initial knowledge) $\mathrm{I}N$, $\mathrm{O}UT$ are disjoint subsets of Ind and $\mathcal{S}$ is a set of equations such that for all $x_i \in \mathcal{V}$ one of the following holds:*

- *$i \in \mathrm{I}N$*

- *There exists a ground term $t \in \mathcal{K}$ and an equation $x_i \stackrel{?}{=} t$ in $\mathcal{S}$;*

- *There exists a rule $l_1, \ldots, l_m \rightarrow r \in \mathcal{R}$ such that $\mathcal{S}$ contains the equations $x_i \stackrel{?}{=} r$ and $x_{\alpha_j} \stackrel{?}{=} l_j$ for $j \in \{1, \ldots, m\}$ with $\alpha_j < i$.*

A symbolic derivation is closed *if* $\text{IN} = \text{OUT} = \emptyset$, *and may be simply denoted by* $(\mathcal{V}, \mathcal{S}, \mathcal{K})$. *A substitution* σ *satisfies a closed symbolic derivation if* $\sigma \models_{\mathcal{E}} \mathcal{S}$.

To improve readability in the examples, we will replace every index i in a set of indexes I by the variable x_i that is associated to it. In the same way we will write

Example 5 *The role A of $\mathcal{P}$ played by agent a can be described by the symbolic deduction system* $\langle \mathcal{F}, \mathcal{R}_B, \mathcal{H} \rangle$ *with $\mathcal{F}$ and $\mathcal{H}$ as before and*

$$\mathcal{R}_A = \{ c_{\min} \rightarrow se(h(order(x,y,z)), SK_A) \cdot order(x,y,z)$$
$$se(h(order(x,y,z)), PK_A) \cdot se(accesscode(z), SK_x) \rightarrow ok \}$$

and a symbolic derivation for this role is:

$$\mathcal{V} = \{ x_1^A, y_1^A, x_2^A, y_2^A \}, In = \{ x_1^A, x_2^A \}, Out = \{ y_1^A, y_2^A \}$$
$$\mathcal{K} = \{ a, b, K_a, K_b, pub(K_a), pub(K_b), OK \}$$
$$\mathcal{S} = \{ \; x_1^A = c_{\min}, y_1^A = se(h(order(x,y,z)), SK_a) \cdot order(x,y,z),$$
$$x_2^A = se(h(order(x,y,z)), PK_a) \cdot se(accesscode(z), SK_x), y_2^A = ok \}$$

The role B of $\mathcal{P}$ played by agent b can be described by the symbolic deduction $\langle \mathcal{F}, \mathcal{R}_B, \mathcal{H} \rangle$ *with $\mathcal{F}$ and $\mathcal{H}$ as before and*

$$\mathcal{R}_B = \{ se(h(order(x,y,z)), SK_b) \cdot order(x,y,z) \rightarrow se(h(order(x,y,z)), PK_a) \cdot se(accesscode(z), SK_x) \}$$

and a symbolic derivation for this role is:

$$\mathcal{V} = \{ x_1^B, y_1^B \}, \; In = \{ x_1^B \}, Out = \{ y_1^B \},$$
$$\mathcal{K} = \{ a, b, K_a, K_b, pub(K_a), pub(K_b), OK \},$$
$$\mathcal{S} = \{ x_1^B = se(h(order(x,y)), K_b) \cdot order(x,y), y_1^B = se(h(order(x,y)), K_a) \cdot OK, \}$$

For simplicity, we have modeled the behaviour of the roles in an abstract way that hides the actual computation that are performed. In our framework, we can also give a much more detailed description of the protocol, that describes all this computations: we add to each role the deduction rules for constructing pairs, encryption, decryption,... Once this is achieved we can describe the construction of the first message sent by A by describing the construction of each part. For instance, the subterm $se(h(order(x,y,z)), SK_a)$ *is built using an instance of the rule* $x, y \rightarrow se(x, y)$.

In order to compose two symbolic derivations we will connect bijectively (or identify) some input variables of one derivation with some output variables of the other and vice-versa. This connection should be compatible with the variable orderings inherited from each component, in a way that is detailed in the following definition:

Definition 6 *(Composition of Symbolic Derivations) Let* $\mathcal{SD}_1 = (\mathcal{V}_1, \mathcal{S}_1, \mathcal{K}_1, \text{IN}_1, \text{OUT}_1)$ *and* $\mathcal{SD}_2 = (\mathcal{V}_2, \mathcal{S}_2, \mathcal{K}_2, \text{IN}_2, \text{OUT}_2)$ *be two symbolic derivations, with disjoint sets of variables and index sets* $(Ind_1, <_1)$ *and* $(Ind_2, <_2)$ *respectively. Let* I_1, I_2, O_1, O_2 *be subsets of*

$\text{IN}_1, \text{IN}_2, \text{OUT}_1, \text{OUT}_2$ *respectively. and assume that there is an order-preserving bijection* ϕ *from* $I_1 \cup I_2$ *to* $O_1 \cup O_2$ *such that* $\phi(I_1) = O_2$ *and* $\phi(I_2) = O_1$. *A composition of two symbolic derivations along the sets* I_1, O_1, I_2, O_2 *is a symbolic derivation*

$$(\mathcal{V}, \phi(\mathcal{S}_1 \cup \mathcal{S}_2), \mathcal{K}_1 \cup \mathcal{K}_2, (\text{IN}_1 \cup \text{IN}_2) \setminus (I_1 \cup I_2), (\text{OUT}_1 \cup \text{OUT}_2) \setminus (O_1 \cup O_2))$$

where ϕ *has been extended to a substitution on terms, by taking* $\phi(x_i) = x_j$ *when* $\phi(i) = j$ *where* $\mathcal{V}$ *is a sequence of variables indexed by* $Ind = (Ind_1 \setminus I_1) \cup (Ind_2 \setminus I_2)$, *ordered by a linear extension of the transitive closure of the relation:*

$$<_1 \cup <_2 \cup \{(u, v) \mid v = \phi(w) \text{ and } u <_1 w \text{ or } u <_2 w\}$$

and such that the variable of index i *in* $\mathcal{V}$ *is equal to the variable of index* i *in* $\mathcal{V}_1$ *if* $i \in Ind_1$, *and to the variable of index* i *in* $\mathcal{V}_2$ *if* $i \in Ind_2$.

It can be checked that the composition of two symbolic derivations is indeed a symbolic derivation. The composition of more than two symbolic derivations can be defined in the same way.

Example 6 *Let us consider the symbolic derivation of the previous example, and let us assume that the variables are ordered as* $y_1^A, x_1^A, y_1^B, y_2^B, x_2^A, y_2^A$.

The normal execution of the protocol $\mathcal{P}$ *corresponds to the composition of the two symbolic derivations of the previous example along* $I_1 = \{x_2^A\}, O_1 = \{y_1^A\}, I_2 = \{x_1^B\}, O_2 = \{y_1^B\}$

where the following new equations are added:

$y_1^A = x_1^B$ *the first message emitted by* A *is the first message received by* B

$y_1^B = x_2^A$ *the first message emitted by* B *is the second message received by* A

4.4   The formal statement of protocol insecurity

Informally we call *protocol insecurity problem* the question whether the intruder can build a symbolic derivation that can be composed with the protocol so that the secret gets revealed to the intruder. We can always extend the protocol (and the associated symbolic derivation) by a dummy step where the secret is received in clear by a honest agent that already knows it (hence it has been sent in clear, hence the intruder was able to derive it). This extended protocol can be completely executed iff the initial protocol was insecure. This shows that we can reduce the insecurity probleme to the problem whether the composition of a symbolic derivation and a symbolic intruder derivation admits a solution.

We now give a formal definition of the insecurity problem we consider:

Insecurity Problem

Input:	a symbolic derivation $\mathcal{C}_h$ for $\langle \mathcal{F}, \mathcal{R}, \mathcal{E} \rangle$ (protocol), a set of terms $\mathcal{K}_i$ (intruder knowledge)	
Output:	SAT iff there exists a symbolic derivation $\mathcal{C}_i = (\mathcal{V}_i, \mathcal{S}_i, \mathcal{K}_i, \text{IN}_i, \text{OUT}_i)$ for $\langle \mathcal{F}, \mathcal{R}, \mathcal{E} \rangle$, a closed composition $\mathcal{C}_a$ of $\mathcal{C}_i$ and $\mathcal{C}_h$, and a substitution σ such that σ satisfies $\mathcal{C}_a$.	

In the rest on the paper we shall consider an equational theory which the union of disjoints theories. For technical reasons, this leads us to consider a slightly more general problem:

Ordered Satisfiability

Input:	a symbolic derivation $\mathcal{C}_h$ for $\langle \mathcal{F}, \mathcal{R}, \mathcal{E} \rangle$ (protocol), a set of terms $\mathcal{K}_i$ (intruder knowledge), X the set of all variables and C the set of all free constants occurring in $\mathcal{C}_h$ and a linear ordering $\prec$ on $X \cup C$.
Output:	SAT iff there exists a symbolic derivation $\mathcal{C}_i = (\mathcal{V}_i, \mathcal{S}_i, \mathcal{K}_i, \text{IN}_i, \text{OUT}_i)$ for $\langle \mathcal{F}, \mathcal{R}, \mathcal{E} \rangle$, a closed composition $\mathcal{C}_a$ of $\mathcal{C}_i$ and $\mathcal{C}_h$, and a substitution σ such that σ satisfies $\mathcal{C}_a$ and: $\forall c \in C, x \prec c$ implies $c \notin \text{Sub}_{\text{syn}}(x\sigma)$

4.5   Comparison with pattern-matching and equational models

The model that we propose for wokflow analysis shares strong similarities with two models already proposed for cryptographic protocols: the *pattern-matching* model and the *equational* model. We discuss the relationship between these models and our proposition. In each model, protocols are a sequence of receive/send actions and use cryptographic primitives and pairing functions. They differ in the way messages are analyzed and the properties of cryptographic primitives and algebraic properties of operations are used.

The pattern-matching model.   Here the pattern-matching is used to retrieve the parts of the messages received. For instance, assume that a role A must execute the send/receive sequence $\text{se}(x, y) \rightarrow \text{se}(x, K_{pub(B)})$. Let $\text{se}(N_B, K)$ be the actual message received by A. Then the pattern-matching algorithm computes the $x = N_B$ component independently of the fact that A knows the key K or not (possibly K is a key that A will learn later on). This model can be seen as a significative improvement on the original Dolev-Yao model [13] since it permits to get rid of the equational properties of explicit destructors, leading to the so-called *pattern-matching model*.

This model has been extended to handle algebraic properties of operation like $\oplus$ or modular exponentiation but this may leads to unrealistic protocols: design a role that receives a message $x \oplus y$ and send back x. Setting $x = y$, the role is set to receive 0 and to send back any message x. Syntactic ad-hoc conditions has been designed to avoid such situation (see [8] for the exclusive-or), but the first generic condition is the notion of *well-defined* protocol defined in [19]. Although this approach is sound, it doesn't really address why these protocols are not plausible. In the $\oplus$ example, there is simply no way for the role to extract a value x from $x \oplus y$ using the usual operations that a role performs. Our models describes this property by forbidding deduction rules like $x \oplus y \rightarrow x$ (this rule doesn't satisfy the definition requirements). But, we shall not use a rule $\text{RECV}(g^y).\text{SEND}(y)$ as a deduction rule: it is well-defined but it corresponds to the resolution of a modular logarithm problem, which is not a plausible operation for a role. Our model is more precise than the pattern matching model. Conversely, an induction on derivation shows that any symbolic

derivation is an execution of a well-defined protocol execution in the pattern matching model. This shows that we can lift the decidability results already stated in that case to our model. The combination result that we prove in Section 9 provides a the combined analysis of independent operators, which greatly simplifies the proofs and allows a modular implementation of dedicated protocol analysis methods. No such modular appraoch exists for well-defined protocols. Futhermore, we are not aware of any general decision procedure for the well-definedness of protocols, when a simple syntactic analysis of each rule in set of derivation rule is enough in our framework. We also strongly conjecture that the decidability of ground reachability problems suffices to decide whether an object is a symbolic derivation in the case of an infinite number of deduction rules.

The equational model. The original Dolev and Yao model [13] used explicit constructors (*e.g.* for encryption $se(,),...)$ and destructors (*e.g.* for decryption $sd(,))$, whith the equational theories describing the properties of these operations. A role uses explicit destructors to extract the relevant parts of the message instead of using pattern-matching for this purpose. For instance, the action of retrieving the encrypted part of the message $m = se(x, K)$ is modelled by the equation $x = sd(m, K)$. The executability of step i of the protocol is guaranteed by the satisfiability of the system of equations. Futhermore, the equational theory is enriched by the equations stating that a destructor f_d^i is the inverse of a constructor f_c:

$$f_d^i(f_c(x_1, \ldots, x_n)) = x_i$$

In the case of the Dolev-Yao intruder, the addition of unification constraints [9] gives *exactly* the same solutions as in the syntactic Dolev-Yao case. Indeed, it suffices to replace the pattern-matching protocol step $\text{RECV}(se(x, K)).\text{SEND}(x)$ by the equational protocol step $\text{RECV}(y).\text{SEND}(sd(y, K))$ with the unification constraint $y = sd(x, K)$. This equivalence stems from the fact $sd(sd(x, K), K)$ is equal, without any ambiguity and after rewriting, to x. The assumption in the deterministic model presented in [9] is that any agent operation can be specified with standard rewrite rules.

In web services application, this assumption is no longer valid since the specification of operations on a message in XML syntax may be ambiguous. For instance, a typical specification can be *get the son of label a of the hedge x* which is not uniquely determined when the hedge x contain several nodes with the label a. This has been already already identified as XML rewriting attacks (see [16] for a proposed counter-measure).

To model the previous operation by a destructor, say f_a, we must have an equation $f_a(a(x) \cdot y') = a(x)$, and apply f_a on y. But, for a term $y = a(x_1) \cdot a(x_2)$, we have both $f_a(y) = a(x_1)$ and $f_a(y) = a(x_2)$, hence $a(x_1) = a(x_2)$: all terms heade by a are undistinguishable. This is an overapproximation which precludes any reasonable analysis of web services.

We believe that the simplest way to reduce this problem is the one adopted in this paper. The fact that it keeps the intuitive destructor model of expliciting the operations performed by all agents has an additional and not to be under-estimated advantage: the reader can easily check that in the specific case where all deduction rules are of the shape $\text{Var}(t) \rightarrow t$,

it is possible to remove all intermediary steps (keeping when necessary the equations) in a symbolic derivation and to replace them by "contexts" on top of the initial knowledge and received variables. In other words, when deduction rules are of the shape $\mathrm{Var}(t) \to t$, symbolic derivations can be trivially transformed into equivalent (w.r.t. the set of solutions) deterministic constraint satisfaction problems of [9]. This means that the decidability results known in this context can be transfered directly into the symbolic derivations framework.

Concluding remark. With respect to our current focus on Web Services, the setting presented in this paper permits to import decision procedures already devised in either the pattern-matching model or in the equational model. This permits *e.g* to have for free the decidability of the deduction system that includes modular exponentiation, exclusive-or, and *e.g* the syntactic Dolev-Yao intruder model. This means that in spite of the complexity of a signature containing different encryption operations, some associative and some associative-commutative symbols for hedges and a few free symbols to denote XML nodes, a first decision procedure for the automated analysis of Web Services can be given by providing just decision procedures for associative and AC symbols. The procedure for the former was given in an earlier work [6], so we will just give here a decision for a deduction theory with an AC symbol that denotes a multi-set.

5   Union of equational theories

From now on, we shall assume that $\mathcal{F}$ is the disjoint union of two signatures $\mathcal{F}_1$ and $\mathcal{F}_2$, and we assume that $\mathcal{E}_1$ (resp. $\mathcal{E}_2$) is a consistent equational theory on $\mathcal{F}_1$ (resp. $\mathcal{F}_2$). We denote by $\mathcal{E}$ the union of the theories $\mathcal{E}_1$ and $\mathcal{E}_2$. For instance $\mathcal{F}_1 = \{\langle, \rangle, \mathrm{se}(,)\}$ and $\mathcal{E}_1 = \emptyset$, $\mathcal{F}_2 = \{\oplus, 0\}$ and $\mathcal{E}_2 = \{x \oplus 0 = x, x \oplus x = 0, x \oplus y = y \oplus x, x \oplus (y \oplus z) = (x \oplus y) \oplus z\}$.

5.1   Definitions for the union of two signatures

We recall that C is an infinite set of free constants, that $\mathcal{X}$ is an infinite set of variables and that $\mathrm{T}(\mathcal{F}_1, \mathcal{X})$ (resp. $\mathrm{T}(\mathcal{F}_2, \mathcal{X})$) denotes the set of terms on $\mathcal{F}_1 \cup \mathrm{C}$ (resp. $\mathcal{F}_2 \cup \mathrm{C}$). We recall that the ordering $<$ is a simplification ordering on $\mathrm{T}(\mathcal{F}, \mathcal{X})$ total on $\mathrm{T}(\mathcal{F})$ such that the constant $\mathrm{c}_{\min}$ is the minimal element for $<$.

   A term t in $\mathrm{T}(\mathcal{F}_1, \mathcal{X})$ (resp. in $\mathrm{T}(\mathcal{F}_2, \mathcal{X})$) is called a *pure* 1-term (resp. a pure 2-term). We denote by $\mathrm{Sign}(\cdot)$ the function that associates to each term $t \notin \mathrm{C} \cup \mathcal{X}$ the signature ($\mathcal{F}_1$ or $\mathcal{F}_2$) of its root symbol. For $t \in \mathrm{C} \cup \mathcal{X}$ we define $\mathrm{Sign}(t) = \bot$, with $\bot$ a new symbol. The term s is *alien* to u if $\mathrm{Sign}(s) \neq \mathrm{Sign}(u)$. We now introduce a notion of subterm values of a term t. Intuitively these subterm values are syntactic subterms of t that are either equal to t or a strict maximal alien syntactic subterm of a subterm value of t. This notion permits to simplify the reasoning on alien subterms of a term.

Definition 7 *(factors) The set of* factors *of a term t is denoted* $\mathrm{Factors}(t)$ *and is the set of maximal syntactic strict subterms of t that are either alien to t or atoms.*

Example 7 *Consider $\mathcal{F}_1 = \{\cdot, a, b, c\}$ and $\mathcal{F}_2 = \{f\}$ where f has arity 1. Then*

$$\left\{ \begin{array}{rcl} \text{Factors}(f(f(a)) \cdot (b \cdot c)) & = & \{f(f(a)), b, c\} \\ \text{Factors}(f(f(f(b) \cdot c))) & = & \{f(b) \cdot c\} \\ \text{Factors}(a) & = & \emptyset \end{array} \right.$$

Note that if t itself is an atom, its set of factors is empty. We now define the notion of *subterm values.*

Definition 8 *(Subterms) Given a term t, the set of its* subterm values *is denoted by* $\text{Sub}(t)$ *and is defined recursively by* $\text{Sub}(t) = \{t\} \cup \bigcup_{u \in \text{Factors}(t)} \text{Sub}(u)$.

By extension, for a set of terms E, the set $\text{Sub}(E)$ is defined as the union of the subterms values of the elements of E.

Example 8 *Consider $\mathcal{F}_1$ and $\mathcal{F}_2$ as in Example 7. Then*

$$\left\{ \begin{array}{rcl} \text{Sub}(f(f(a)) \cdot (b \cdot c)) & = & \{f(f(a)) \cdot (b \cdot c), f(f(a)), a, b, c\} \\ \text{Sub}(f(f(f(b) \cdot c))) & = & \{f(f(f(b) \cdot c)), f(b) \cdot c, f(b), b, c\} \\ \text{Sub}(a) & = & \{a\} \end{array} \right.$$

This shows the difference with the notion of syntactic subterms.

> In the rest of this paper and unless otherwise indicated, *the notion of subterm will refer to subterm values.*

Applying unfailing completion to $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$, it is easy to notice [2] that the set of generated equations R is the disjoint union of the two systems R_1 and R_2 also obtained by applying unfailing completion procedures to $\mathcal{E}_1$ and to $\mathcal{E}_2$ respectively. (Since $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$ and the $\mathcal{E}_i$ are assumed to be consistent *i.e.* the identity $x =_{\mathcal{E}_i} y$ does not hold in either theory, the critical pair generation will produce only pure equations.) We denote $(t)\!\downarrow$ the normal form of a term t for the rewrite system $\rightarrow_R$.

The abstraction of alien subterms, i.e. their replacement by constants, is used extensively in the last part of the paper. We assume that we can associate to every term $t \in \text{T}(\mathcal{F})$ a new free constant c_t

Definition 9 *(Abstraction) For $i = 1, 2$, the function $ab_i(t)$ is defined by:*

- $ab_i(f(t_1, \ldots, t_n) = f(ab_i(t_1), \ldots, ab_i(t_n))$ *if $f \in \mathcal{F}_i$,*

- $ab_i(f(t_1, \ldots, t_n) = c_{f(t_1, \ldots, t_n)}$ *if $f \in \mathcal{F}_j, j \neq i$,*

- $ab_i(c) = c$ *if $c \in \text{C}$ otherwise $ab_i(c) = \text{c}_{\min}$*

The abstraction $\text{Abs}_i(t)$ of the term t is defined by $ab_i((t)\!\downarrow)$

Let s, v be two terms, then the *replacement* $\delta_{s,v}$ is defined by $t\delta_{s,v} = t[\Pi \leftarrow v]$ where Π is the set of positions p such that s occurs at position p in t as a subterm value. By definition δ_s denotes $\delta_{s,\text{c}_{\min}}$.

5.2   Normalisation and replacements

This section gives several technical lemmas that prove that normalization of a term will not introduce new factors except $c_{\min}$ (provided that the factors are already normalized). These Lemmas are used to prove the properties of derivations given in Section 6.

Lemma 4 *Let $t \in \mathrm{T}(\mathcal{F})$ be a ground term such that all factors of t are in normal form. Then either $(t)\!\downarrow\, \in C_{\mathrm{spe}} \cup \mathrm{Factors}(t)$ or $\mathrm{Sign}(t) = \mathrm{Sign}((t)\!\downarrow)$ and $\mathrm{Factors}((t)\!\downarrow) \subseteq C_{\mathrm{spe}} \cup \mathrm{Factors}(t)$.*

PROOF. Several cases may occur:

- If t is in normal form the result holds.

- If t is a constant, then $t \to t'$ and $\mathrm{Factors}(t') = \{r_1, \dots, r_n\}$ implies that $t \to t''$ is obtained from t' by replacing each factor r_i by $c_{\min}$ (the rewriting is still allowed because $r_i > c_{\min}$ for $i = 1, \dots, n$).

- If t is not in normal form, we prove that there is a ground term t' such that $t \to t'$ and either $t' \in C_{\mathrm{spe}} \cup \mathrm{Factors}(t)$ or else (i) $\mathrm{Factors}(t') \subseteq \mathrm{Factors}(t) \cup C_{\mathrm{spe}}$ (ii)$\mathrm{Sign}(t) = \mathrm{Sign}(t')$. If there exists such t', then there is a sequence of rewriting steps $t \to t_1 \to t_2 \dots$ such that for all t_i, either $t_i \in C_{\mathrm{spe}} \cup \mathrm{Factors}(t)$ or else $\mathrm{Factors}(t') \subseteq C_{\mathrm{spe}} \cup \mathrm{Factors}(t)$. Since R is ground normalizing, the sequence is finite and the last term is $(t)\!\downarrow$, which proves the result.

  Let $t \to t'$ be a rewrite step that uses the equation $l = r$ and substitution σ such that $l\sigma > r\sigma$. Let $t = C[r_1, \dots, r_n]$ with $r_1, \dots, r_n$ be the factors of t. Either $t' \in C_{\mathrm{spe}}$ or $t' \in factor t$ and we are done, or $\mathrm{Sign}(t') = \mathrm{Sign}(t)$. Since the factors are in normal form, the rewriting is performed at a position p in C with a substitution σ. Since the equation $l = r$ is pure and in the same theory as C, we get that any factor of t and t' occurs at a position below the position of a variable of r or l. Figure 5.2 shows how we can replace an occurrence of a factor r_{new} occurring in t' but not in t by $c_{\min}$ to get another rewrite step such that all occurrences of r_{new} introduced by $y\sigma$ are replaced by $c_{\min}$. The ordering condition for rewriting still holds since $c_{\min}$ is the smallest term and since the ordering is stable under context ($u < v \Rightarrow C[u] < C[v]$).

$\square$

Lemma 5 *Let t be a term such that its factors are in normal form, let $s \notin C_{\mathrm{spe}}$ be a ground term in normal form, let v be a ground term. If $t \to_R t'$ then $s \neq t'$ or $\mathrm{Sign}(s) \neq \mathrm{Sign}(t)$ implies $t\delta_{s,v} =_R t'\delta_{s,v}$.*

PROOF. The assumption $t \to_R t'$ implies that t is not in normal form and thus $s \neq t$. Occurrences of u as a subterm in t are thus only at the level of factors or below it. Let π be the position in t at which the equation $l = r \in R$ is applied with a substitution τ. Let $\tau' = \tau\delta_{s,v}$ and let us prove that $t\delta_{s,v} =_{l=r} t'\delta_{s,v}$

By Lemma 1 the term l is not a variable. Thus we have the equality $\mathrm{Sign}(l) = \mathrm{Sign}(t)$. Since l is pure by construction of R the factors of $l\tau$ are factors of t and thus $l\tau'$ is a

$x\sigma' = x\sigma$ for x occurring in l

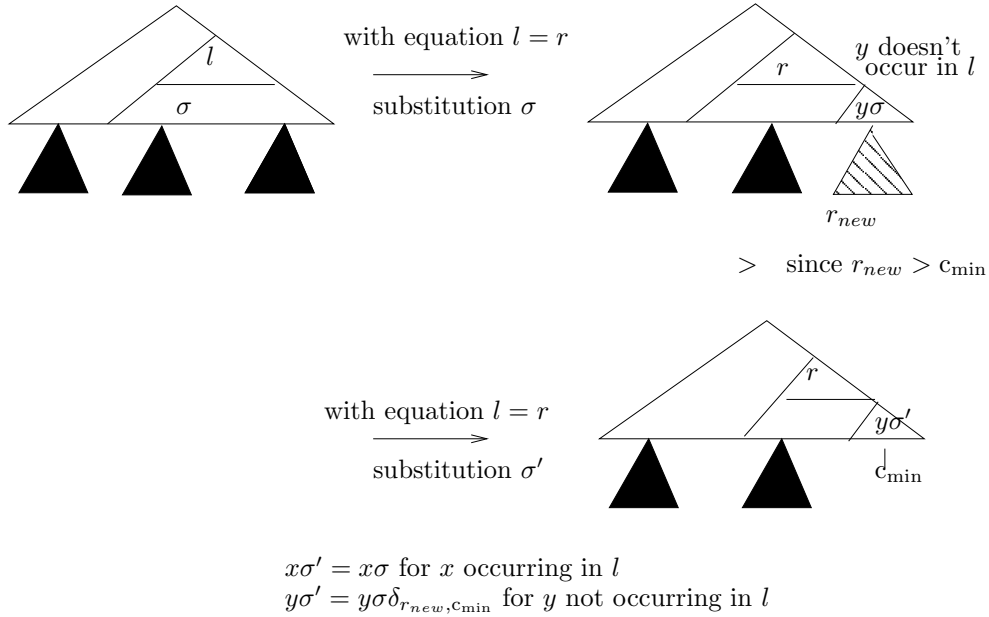$y\sigma' = y\sigma\delta_{r_{new},c_{min}}$ for y not occurring in l

Figure 1: Normalization doesn't yield new factors

syntactic subterm of $t\delta_{s,v}$. Thus it suffices to prove that the hypotheses on s and t imply $t\delta_{s,v}[l\tau' \leftarrow r\tau'] = t[l\tau \leftarrow r\tau]\delta_{s,v}$.

If $s \neq t'$ the replacement on the right-hand side of the equation only occurs at the position of factors of t' or below. It thus occurs on factors of t or of τ and the equality holds.

If $s = t'$ then we have $\mathrm{Sign}(s) \neq \mathrm{Sign}(t)$ by hypothesis and thus $\mathrm{Sign}(t') \neq \mathrm{Sign}(t)$. Since $s \notin \mathrm{C_{spe}}$ by hypothesis this implies that $s \in \mathrm{Factors}(t)$ by Lemma 4. We easily see that the only possible case is that $r \in \mathrm{Var}(l)$, $r\tau = s$ and $\pi = \epsilon$. In this case we have $t'\delta_{s,v} = v = r\tau'$ and thus the Lemma is true. $\qquad\square$

Lemma 6 *Let t be a ground term and $s \notin \mathrm{C_{spe}}$ be a term in normal form such that the following condition $Cond(t,s)$ is satisfied:*

(i) If $s = (t)\downarrow$ then $\mathrm{Sign}(s) \neq \mathrm{Sign}(t)$ and

(ii) If $s = (r)\downarrow$ for some $r \in \mathrm{Sub}(t) \setminus \{t\}$ such that $r \neq s$, then s is a free constant.

Let v be a ground term. If t is reducible then there exists t' such that $t \rightarrow_R t'$, $t\delta_{s,v} =_R t'\delta_{s,v}$, and $Cond(t',s)$ holds if t' is reducible.

PROOF.

- First, we prove that replacement and rewriting commute.

  Since t is reducible, there exists t' such that $t \rightarrow t'$ by rewriting some subterm $u \in Sub(t)$ into u'. Let us choose u minimal for the subterm ordering. This implies that all factors of u are in normal form. Two cases occur:

  - $u = t$. Either $s \neq u'$ or $s = u'$. Since s is in normal form, $u = t \rightarrow u' = s$ implies $s = (t)\downarrow$ and condition (i) of $Cond(t, s)$ yields $\text{Sign}(t) \neq \text{Sign}(s)$. Therefore either $s \neq t'$ or $\text{Sign}(t) \neq \text{Sign}(s)$, and Lemma 5 is applicable, yielding that $t\delta_{s,v} = t'\delta_{s,v}$.

  - $u \neq t$. Since u is reducible, $u \neq s$ and condition (ii) of $Cond(t, s)$ yields that s is a free constant. Then $\text{Sign}(s) \neq \text{Sign}(u)$. Lemma 5 is applicable to u, yielding that $u\delta_{s,v} =_R u'\delta_{s,v}$.

    Let p_u be the position of the subterm u in t (hence of the subterm u' in t'). We prove that there is no subterm $r' \in \text{Sub}(t')$ at a position q (strict) prefix of p_u such that $r' = s$. Otherwise, there exists a subterm r in t at position p such that $r \rightarrow s$. Since the rewriting occurs at position p_u we must have $\text{Sign}(r) = \text{Sign}(r') = \text{Sign}(s)$. The case $r = t$ and $p = \epsilon$ is prevented by condition (i) and the case $r \neq t$ is prevented by condition (ii) since it implies that s is a free constant (hence $\text{Sign}(r) \neq \text{Sign}(s)$).

    Finally, if s occurs at position p in t' with p not a prefix nor a suffix of p_u, we have a corresponding occurrence of s at the same position in t.

    Therefore $t\delta_{s,v} = t'\delta_{s,v}$.

- Then we prove now that $Cond(t', s)$ holds or that t' is irreducible.

  - We prove that $s = (t')\downarrow$ implies that $\text{Sign}(t') \neq \text{Sign}(s)$ (condition (i) of $Cond(t', s)$). If the rewriting step doesn't occur at root position, then $\text{Sign}(t) = \text{Sign}(t')$. Therefore $\text{Sign}(t') \neq \text{Sign}(s)$.

    If the rewriting occurs at root position, then there exists a rewrite rule $l \rightarrow r$ such that $t = l\sigma$ and $t' = r\sigma$. Then $\text{Sign}(r\sigma) = \text{Sign}(l\sigma)$ unless r is a variable x, and $x\sigma$ is an alien subterm of $l\sigma$. Either $s = x\sigma$ which is irreducible, or $s \neq x\sigma$. In this latter case, since $s = (t')\downarrow = (x\sigma)\downarrow$, by condition (ii) of $Cond(t, s)$, we get that s is a free constant. Since $x\sigma \neq s$, the term $x\sigma$ is reducible and $\text{Sign}(x\sigma) \neq \text{Sign}(s)$.

  - We prove condition (ii) of $Cond(t', s)$. If $r' \in \text{Sub}(t')$ is a subterm of t' at position p such that $r' \neq t$ and r' is not in normal form, then there exists a subterm $r \in \text{Sub}(t)$ such that $r \rightarrow r'$. Therefore $s = (r')\downarrow$ implies $s = (r)\downarrow$ and s is a free constant by condition (ii) of $Cond(t, s)$.

  $\square$

Lemma 7 *Let t be a term, σ be a normal ground substitution, and $s \notin \text{C}_{\text{spe}}$ be a term in normal form such that the following conditions hold:*

1. $s = (t\sigma)\!\downarrow$ *implies* $\mathrm{Sign}(s) \neq \mathrm{Sign}(t)$ *or* t *is a variable;*

2. $s = (t_{|p}\sigma)\!\downarrow$ *for some position* $p \neq \epsilon$ *of a non-variable subterm in* t, *then* s *is a free constant and* $s \notin \mathrm{Sub}(t);$

3. $s = x\sigma$ *for some variable* x *at position* p *in* t, *then* p *is the position of a subterm in* $t\sigma$.

Then the equality $(t\sigma)\delta_{s,v} =_R t(\sigma\delta_{s,v})\!\downarrow$ *holds for any ground term* v.

PROOF. Let t, s, and σ be two terms and a substitution satisfying the conditions of the lemma. Let $\sigma' = (\sigma\delta_{s,v})\!\downarrow$.

First, if t is a variable, notice that σ in normal form implies the resulting equation becomes $t\sigma\delta_{s,v} =_R t\sigma\delta_{s,v}$, which is trivial. Thus we can assume that t is not a variable and thus that $\mathrm{Sign}(t)$ is defined.

Claim 1 *For all non-variable subterm* u *of* t *one has* $u\sigma \neq s$.

PROOF OF THE CLAIM. By contradiction assume there exists $u \in \mathrm{Sub}(t) \setminus \mathrm{Var}(t)$ such that $u\sigma = s$. If $u = t$ we have $t\sigma = s$. Since t is not a variable, this implies $\mathrm{Sign}(t) = \mathrm{Sign}(s)$. Furthermore, since s is in normal form, this implies that $t\sigma$ is also in normal form, which contradicts the first condition. Thus a candidate u such that $u\sigma$, and therefore $(u\sigma)\!\downarrow$, is equal to s is in $\mathrm{Sub}(t) \setminus (\mathrm{Var}(t) \cup \{t\})$. The second condition implies that s must be a free constant not occurring in t, which contradicts the fact that u is not a variable and $u\sigma = s$. $\Diamond$

Claim 2 $(t\sigma)\delta_{s,v} = t(\sigma\delta_{s,v})$.

PROOF OF THE CLAIM. Let p be a position in t such that $(t\sigma)_{|p} = s$. If p is the position of a variable in t, then by the third condition p is the position of a subterm in $t\sigma$ and s is replaced by v at position p on both sides.
If p is not the position of a variable in t, then by the definition of the subterm relation it is a subterm position in t iff it is a subterm position in $t\sigma$. Thus if s is replaced by v at position p in $t\sigma$, there exists a non-variable *subterm* u of t such that $u\sigma = s$, which contradicts Claim 1.
Thus all replacements on the left-hand side are applied at the level at or below a variable. The equality follows. $\Diamond$

Let us now prove the Lemma. Applying a bottom-up normalisation of $t(\sigma\delta_{s,v})$ stopping at the level of variables of t yields $t(\sigma\delta_{s,v}) =_R t(\sigma\delta_{s,v})\!\downarrow$. By Claim 2 this implies $(t\sigma)\delta_{s,v} =_R t(\sigma\delta_{s,v})\!\downarrow$, and we conclude the proof. $\square$

Lemma 8 *Let* t *be a term,* σ *be a normal ground substitution, and* $s \notin \mathrm{C}_{\mathrm{spe}}$ *be a term in normal form such that the following conditions hold:*

1. $s = (t\sigma)\!\downarrow$ *implies* $\mathrm{Sign}(s) \neq \mathrm{Sign}(t)$ *or* t *is a variable;*

2. $s = (t_{|p}\sigma)\!\downarrow$ *for some position* $p \neq \epsilon$ *of a non-variable subterm in* t, *then* s *is a free constant and* $s \notin \mathrm{Sub}(t)$;

3. $s = x\sigma$ *for some variable* x *at position* p *in* t, *then* p *is the position of a subterm in* $t\sigma$.

Then the equality $((t\sigma)\!\downarrow\delta_{s,v})\!\downarrow = (t(\sigma\delta_{s,v})\!\downarrow)\!\downarrow$ *holds for any ground term* v.

PROOF. If $t\sigma$ is in normal form, then Lemma 7 and the convergence of R permit to conclude. Otherwise, the conditions on t, s, and σ imply, with $t' = t\sigma$, the ones of Lemma 6. This permits to find a sequence t_i of terms such that $t_0 = t\sigma$, $t_i \to_R t_{i+1}$ and $t_i\delta_{s,v} =_R t_{i+1}\delta_{s,v}$. Since R is convergent, this sequence is finite. Let t_n be the final term of a maximal sequence. By maximality t_n is in normal form, and thus equal to $(t\sigma)\!\downarrow$. Applying Lemma 6 along the normalisation sequence yields $(t\sigma)\delta_{s,v} =_R (t\sigma)\!\downarrow\delta_{s,v}$, and thus by Lemma 7 and the convergence of R we have $((t\sigma)\!\downarrow\delta_{s,v})\!\downarrow = (t\sigma')\!\downarrow$.   □

Lemma 9 *Let* t *be a pure term,* σ *be a normal substitution,* $s \notin \mathrm{C}_{\mathrm{spe}}$ *be a term in normal form with* $\mathrm{Sign}(s) \neq \mathrm{Sign}(t)$. *Then for any ground term* v *one has* $(t(\sigma\delta_{s,v})\!\downarrow)\!\downarrow = ((t\sigma)\!\downarrow\delta_{s,v})\!\downarrow$.

PROOF. Since t is pure and $\mathrm{Sign}(s) \neq \mathrm{Sign}(t)$, if there exists a variable x at position p in t such that $x\sigma = s$, then p is the position of a subterm in $t\sigma$. Since $\mathrm{Sign}(s) \neq \mathrm{Sign}(t)$ we have (trivially) that $(t\sigma)\!\downarrow = s$ implies $\mathrm{Sign}(s) \neq \mathrm{Sign}(t)$. Thus we can apply Lemma 8, thus yielding the desired results.

□

Lemma 10 *Let* t *be a term such that its factors are in normal form, and let* u *and* v *be two ground terms in normal form with* $u \notin \{(t)\!\downarrow\} \cup \mathrm{C}_{\mathrm{spe}}$ *or* $\mathrm{Sign}(u) \neq \mathrm{Sign}(t)$. *Then* $(t\delta_{u,v})\!\downarrow = ((t)\!\downarrow\delta_{u,v})\!\downarrow$.

PROOF. It suffices to decompose t into a pure term t' and a normal substitution σ and to apply Lemma 9.   □

Lemma 11 *Let* t *be a term, let* $c \notin \mathrm{C}_{\mathrm{spe}}$, *then for all term* v,

$$((t)\!\downarrow)\delta_{c,v} =_R t\delta_{c,v}$$

PROOF. Consider the term t' constructed from t by replacing every occurrence of c by a variable x and the substitution σ mapping x to c. The lemma is then a direct consequence of Lemma 8.   □

Let U be a set of terms. A ground term s is said to be *bound* by σ to the term t in U if there exists $t \in U$ such that $(t\sigma)\!\downarrow = s$. A ground term s which is not bound to any term in U is said to be *free* in U. The following lemma permits us to define a new substitution after the replacement of a free subterm of the solution σ.

Lemma 12 *Let t be a term and σ be a normalised substitution. Let $s \notin \mathrm{C_{spe}}$ in $\mathrm{Sub}(t\sigma)$ be such that s is free in $\mathrm{Sub}(t)$ for σ. Then $((t\sigma)\!\downarrow\!\delta_{s,v})\!\downarrow\, = (t(\sigma\delta_{s,v}))\!\downarrow$.*

PROOF. Since s is free in $\mathrm{Sub}(t)$ for σ, there is no subterm u of t, including t and its variables, such that $(u\sigma)\!\downarrow\, = s$. We can thus apply Lemma 8 to obtain the desired result. $\square$

The proof for next Lemma can be found in [10].

Lemma 13 *For all normal substitutions σ, for all terms t and for all $s \in \mathrm{Sub}((t\sigma)\!\downarrow)$ one of the following holds:*

- $s \in \mathrm{C_{spe}}$;

- *There is $u \in \mathrm{Sub}(m)$ such that $(u\sigma)\!\downarrow\, = s$ and $\mathrm{Sign}(u) = \mathrm{Sign}(s)$;*

- *There exists $x \in \mathrm{Var}(t)$ such that $s \in \mathrm{Sub}(x\sigma)$.*

6 Union of deduction systems

In this section, we revisit deductions rules and their properties when the signature is a disjoint union.

6.1 Properties of derivations

Given a set of deduction rules $\mathcal{R}$, we define $\langle \mathcal{R} \rangle$ to be the minimal set of deduction rules such that

(i) $\mathcal{R} \subseteq \langle \mathcal{R} \rangle$ and

(ii) for all $d : t_1, \ldots, t_n \to t \in \langle \mathcal{R} \rangle$, $d' : t_1, \ldots, t'_m \to t' \in \langle \mathcal{R} \rangle$, for all $i \in \{1, \ldots, m\}$ and for all unifier σ of t and t'_i modulo $\mathcal{H}$, the deduction rule: $(t_1, \ldots, t_n, (t'_1, \ldots, t'_m \setminus t'_i) \to t')\sigma$ is in $\langle \mathcal{R} \rangle$.

The rule constructed in (ii) is still a deduction rule since d, d' are deduction rules. Hence deduction rules in $\langle \mathcal{R} \rangle$ are built by composing deduction rules in $\mathcal{R}$ iteratively. By induction on the length of derivations, we can then prove that the deduction systems $\mathcal{D} = \langle \mathcal{F}, \mathcal{R}, \mathcal{H} \rangle$ and $\mathcal{D}' = \langle \mathcal{F}, \langle \mathcal{R} \rangle, \mathcal{H} \rangle$ define the same sets of derivable terms, i.e. for all E we have $\mathrm{Der}_{\mathcal{I}}(E) = \mathrm{Der}_{\mathcal{I}'}(E)$.

We consider now the union of two deduction systems $\mathcal{D}_1 = \langle \mathcal{F}_1, \mathcal{R}_1, \mathcal{E}_1 \rangle$ and $\mathcal{D}_2 = \langle \mathcal{F}_2, \mathcal{R}_2, \mathcal{E}_2 \rangle$, and we are interested in the derivations using $\to_{\mathcal{D}_1} \cup \to_{\mathcal{D}_2}$.

Since $\langle \mathcal{R}_1 \cup \mathcal{R}_2 \rangle = \langle \langle \mathcal{R}_1 \rangle \cup \langle \mathcal{R}_2 \rangle \rangle$, the set of terms derivable with $\langle \mathcal{R}_1 \cup \mathcal{R}_2 \rangle$ is the same set as the set of terms derivable with $\langle \mathcal{R}_1 \rangle \cup \langle \mathcal{R}_2 \rangle$. For technical reason, we use the latter set to define the union of deduction systems.

Definition 10 *The* union of the two deduction systems $\mathcal{D}_1 = \langle \mathcal{F}_1, S_1, \mathcal{E}_1 \rangle$ *and* $\mathcal{D}_2 = \langle \mathcal{F}_2, S_2, \mathcal{E}_2 \rangle$ *is the deduction system* $\mathcal{D} = \langle \mathcal{F}_1 \cup \mathcal{F}_2, \langle S_1 \rangle \cup \langle S_2 \rangle, \mathcal{E}_1 \cup \mathcal{E}_2 \rangle$ *and is denoted by* $\mathcal{D}_1 \cup \mathcal{D}_2$.

From now we assume that deduction steps refer to the deduction system $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$.

We extend the function $\mathrm{Sign}(\_)$ to rules by setting $\mathrm{Sign}(d) = \mathcal{F}_i$ (i=1,2) if all terms in d are pure $\mathcal{F}_i$-terms.

Lemma 14 *Let* $l \to r$ *be a rule in* $\mathrm{GI}(d)$ *with* $d : t_1, \ldots, t_n \to t$ *and* $s \notin \mathrm{C}_{\mathrm{spe}}$ *alien to some* t_i *or* t. *Then* $(l\delta_s)\!\downarrow \to (r\delta_s)\!\downarrow$ *is also a rule in* $\mathrm{GI}(d)$.

PROOF. Let σ be a ground normal substitution such that $(t_1\sigma, \ldots, t_n\sigma)\!\downarrow = l$ and $(t\sigma)\!\downarrow = r$. Let us prove that for $u \in \{t_1, \ldots, t_n, t\}$ one has $((u\sigma)\!\downarrow\delta_s)\!\downarrow = (u(\sigma\delta_s))\!\downarrow$. This suffices to find a ground instance $(l\delta_s)\!\downarrow \to (r\delta_s)\!\downarrow$ in GI(d).

If u is a variable then σ in normal form implies $(u\sigma)\!\downarrow = u\sigma$ and thus $((u\sigma)\delta_s)\!\downarrow = ((u\sigma)\!\downarrow\delta_s)\!\downarrow$.

Otherwise, and since the rule is pure, the assumption implies that $\mathrm{Sign}(u) \neq \mathrm{Sign}(s)$. Since σ is a normal substitution, and since for $u \in \{t_1, \ldots, t_n, t\}$ the term u is pure, the factors of $u\sigma$ are in normal form.

These two previous facts and Lemma 10 imply that for all $u \in \{t_1, \ldots, t_n, t\}$ one has $((u\sigma)\!\downarrow\delta_s)\!\downarrow = ((u\sigma)\delta_s)\!\downarrow$. Since s is alien to the term u we also have $(u\sigma)\delta_s = u(\sigma\delta_s)$. By applying a bottom-up normalisation we have $u(\sigma\delta_s) =_{\mathcal{E}} u(\sigma\delta_s)\!\downarrow$. Putting together these equalities we obtain a substitution $\sigma' = (\sigma\delta_s)\!\downarrow$ such that the rule d instantiated by σ' is $(l\delta_s)\!\downarrow \to (r\delta_s)\!\downarrow$. □

The proof of the two next lemmas rely on the property of ordered rewriting shown in Figure 2.

The next definition is needed to replace each different factor by a different free constant without introducing occurrence of a free constants in a factor. Let $r_1, \ldots, r_n$ be a sequence of terms, $c_1, \ldots, c_n$ be a sequence of constants that don't occur in the r_i's, $c_i \neq \mathrm{c}_{\mathrm{min}}$, we define $\delta_{\vec{r},\vec{c}}$ as the composition $\delta_{r_{i_1}, c_{i_1}} \ldots \delta_{r_{i_n}, c_{i_n}}$ where for each j, r_{i_j} is maximal for the subterm ordering in the sequence $r_{i_{j_1}}, \ldots, r_{j_n}$. The notation $\delta_{\vec{c},\vec{r}}$ defines the inverse operation $\delta_{c_{i_n}, r_{i_n}} \ldots \delta_{r_{i_1}, c_{i_1}}$.

In figure 2, the rewriting step computing s from $(s\delta_{\vec{r},\vec{c}})\!\downarrow\delta_{\vec{c},\vec{r}}$ exists since R is ground normalizing. The fact that each factor of $(s\delta_{\vec{r},\vec{c}})\!\downarrow$ is in $\{c_1, \ldots, c_n\}$ is a consequence of Lemma 4.

Lemma 15 *Let* $s_1, \ldots, s_n \to s \in \mathrm{GI}(d)$ *be a ground normalized instance of a deduction rule* d, *and let* $\mathrm{Sign}(s) \neq \mathrm{Sign}(d)$. *If* $s \notin \{s_1, \ldots, s_n\} \cup \mathrm{C}_{\mathrm{spe}}$, *then there exists* $i \in \{1, \ldots, n\}$ *such that* $\mathrm{Sign}(s_i) = \mathrm{Sign}(d)$ *and* $s \in \mathrm{Factors}(s_i)$.

$$s = (t\sigma)\downarrow = ((s\delta_{\vec{r},\vec{c}})\downarrow \delta_{\vec{c},\vec{r}})\downarrow \qquad\qquad s\delta_{\vec{r},\vec{c}}$$

t σ C r_1 r_p C c_1 c_p

$=_R$ $*$ $r = r_{j_m} \in \{r_1,\dots,r_p\}$

C' r_{i_1} r_{i_l} C' c_{i_1} c_{i_l}

$=_R$

$\dots$

$=_R$

$(s\delta_{\vec{r},\vec{c}})\downarrow \delta_{\vec{c},\vec{r}} =$ $(s\delta_{\vec{r},\vec{c}})\downarrow$

r_{j_1} r_{j_k} c_{j_1} c_{j_k}

$\{r_{j_1},\dots,r_{j_k}\} \subseteq \{r_1,\dots,r_p\}$ $\{c_{j_1},\dots,c_{j_k}\} \subseteq \{c_1,\dots,c_p\}$
because inclusion of sets of free constants
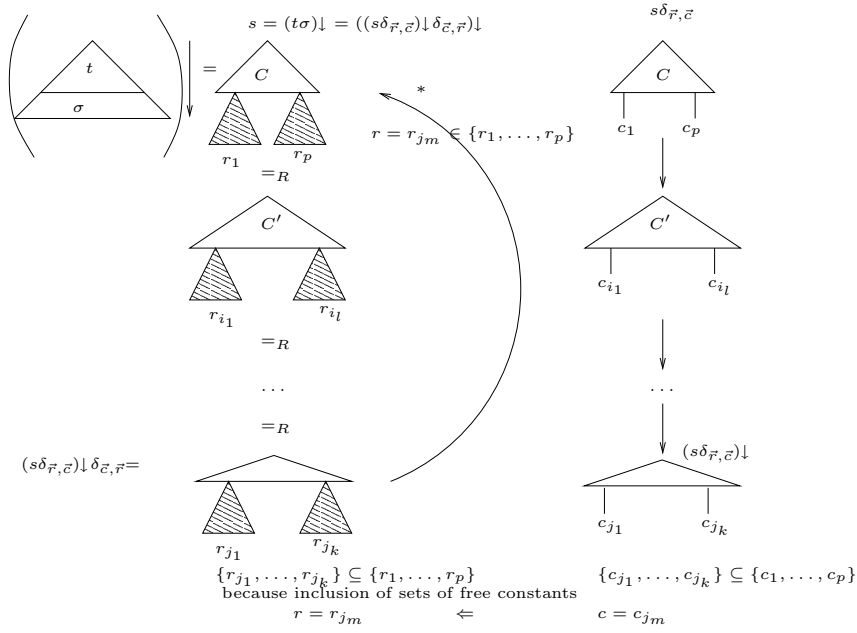$r = r_{j_m}$ $\Leftarrow$ $c = c_{j_m}$

Figure 2: Replacement of subterms by constants and normalization

PROOF. Without loss of generality assume $\mathrm{Sign}(d) = 1$. Let $t_1,\dots,t_n \to t$ be the rule applied, and σ be the ground normal substitution with which it is applied. Let $\vec{S}$ be the set of maximal subterms alien to d not in C_{spe} appearing in the rule, and $\vec{C_S}$ be the set of corresponding abstraction constants resulting from the application of $\mathrm{Abs}_1(\_)$ on the s_i and s. Finally, let $\delta = \delta_{\vec{S},\vec{C_S}}$ (as defined above) and $\delta^{-1} = \delta_{\vec{C_S},\vec{S}}$. Let $u \in \{t_1,\dots,t_n,t\}$.

Since the abstraction constants are away from $\mathrm{Const}(\sigma)$, we have:

- $((u\sigma)\downarrow\delta)\downarrow = (u(\sigma\delta)\downarrow)\downarrow$ by Lemma 12

- $((u\sigma)\delta)\downarrow = ((u\sigma)\downarrow\delta)\downarrow$ by Lemma 7

- $(((u\sigma)\delta)\delta^{-1})\downarrow = (((u\sigma)\downarrow\delta)\downarrow\delta^{-1})\downarrow$ by Lemma 11 and thus

$$(u\sigma)\downarrow = (((u\sigma)\downarrow\delta)\downarrow\delta^{-1})\downarrow$$

Now by Lemma 1 we have:

$$
\begin{aligned}
\mathrm{Const}((((u\sigma)\downarrow\delta)\downarrow\delta^{-1})\downarrow) \cup C_{\mathrm{spe}} \quad &\subseteq \quad \mathrm{Const}(((u\sigma)\downarrow\delta)\downarrow\delta^{-1}) \cup C_{\mathrm{spe}}\\
&= \quad \mathrm{Const}(((u\sigma)\downarrow\delta)\downarrow)\delta^{-1} \cup C_{\mathrm{spe}}\\
&\subseteq \quad \mathrm{Const}((u\sigma)\downarrow\delta)\delta^{-1} \cup C_{\mathrm{spe}}\\
&= \quad (\mathrm{Const}((u\sigma)\downarrow)\delta)\delta^{-1} \cup C_{\mathrm{spe}}\\
&= \quad \mathrm{Const}((u\sigma)\downarrow) \cup C_{\mathrm{spe}}
\end{aligned}
$$

and therefore
$$\mathrm{Const}(((u\sigma){\downarrow}\delta){\downarrow}\delta^{-1}) \cup \mathrm{C_{spe}} = \mathrm{Const}((u\sigma){\downarrow}) \cup \mathrm{C_{spe}}$$

Thus:
$$\begin{aligned}
\mathrm{Const}(((u\sigma){\downarrow}\delta){\downarrow}\delta^{-1})\delta \cup \mathrm{C_{spe}} &= \mathrm{Const}((u\sigma){\downarrow})\delta \cup \mathrm{C_{spe}} \\
\rightarrow \quad (\mathrm{Const}(((u\sigma){\downarrow}\delta){\downarrow})\delta^{-1})\delta \cup \mathrm{C_{spe}} &= \mathrm{Const}((u\sigma){\downarrow})\delta \cup \mathrm{C_{spe}} \\
\rightarrow \quad \mathrm{Const}(((u\sigma){\downarrow}\delta){\downarrow}) \cup \mathrm{C_{spe}} &= \mathrm{Const}((u\sigma){\downarrow}\delta) \cup \mathrm{C_{spe}}
\end{aligned}$$

Finally, and since we have:

- $\mathrm{Const}((t(\sigma\delta){\downarrow}){\downarrow}) \subseteq \cup_{i=1}^{n}\mathrm{Const}((t_i(\sigma\delta){\downarrow}){\downarrow})\cup \mathrm{C_{spe}}$ by the constraint on ground instances of deduction rules

- Since $((u\sigma){\downarrow}\delta){\downarrow} = (u(\sigma\delta){\downarrow}){\downarrow}$ for $u \in \{t_1, \ldots, t_n, t\}$ this implies

$$\mathrm{Const}(((t\sigma){\downarrow}\delta){\downarrow}) \subseteq \cup_{i=1}^{n}\mathrm{Const}(((t_i\sigma){\downarrow}\delta){\downarrow}) \cup \mathrm{C_{spe}}$$

- and thus $\mathrm{Const}(((t\sigma){\downarrow}\delta) \cup \mathrm{C_{spe}} \subseteq \cup_{i=1}^{n}\mathrm{Const}((t_i\sigma){\downarrow}\delta) \cup \mathrm{C_{spe}}$

By definition of δ this implies that the set of maximal subterms of $(t\sigma){\downarrow}$ alien to d is included in the union of the sets of maximal subterms of the $(t_i\sigma){\downarrow}$ alien to d.

If s is itself alien to d, we can assume that s is not on the left-hand side of the rule instance, for otherwise the rule is necessarily a stutter. Thus there exists a t_i such that s is a maximal alien subterm of $(t_i\sigma){\downarrow}$ and $s \neq (t_i\sigma){\downarrow}$. Thus s is a factor of $(t_i\sigma){\downarrow}$ and $\mathrm{Sign}((t_i\sigma){\downarrow}) = \mathrm{Sign}(d)$. $\qquad\square$

Lemma 16 *Let $s_1, \ldots, s_n \rightarrow s \in \mathrm{GI}(d)$ be a ground normalized instance of an deduction rule and let $\mathrm{Sign}(s) = \mathrm{Sign}(d)$. Then for all $r \in \mathrm{Factors}(s)$ either $r \in \{s_1, \ldots, s_n\} \cup \mathrm{C_{spe}}$ or there exists i such that $\mathrm{Sign}(s_i) = \mathrm{Sign}(d)$ and $r \in \mathrm{Factors}(s_i)$.*

PROOF. The proof proceeds in the same way as the proof of Lemma 15 but for the concluding step, which is left to the reader. $\qquad\square$

6.2   Properties of one-step deductions

First we prove some properties that are *local* to a given deduction in a derivation. To begin with, we prove that, informally, if the set of subterms changes after a deduction of a term s by a rule in GI(d), then the only change is the addition of s with $\mathrm{Sign}(d) = \mathrm{Sign}(s)$.

Lemma 17 *Let E and F be two finite sets of normalized terms and let $E \rightarrow_{\mathrm{GI}(d)} F$ be a deduction and assume that $\mathrm{Sub}(E) \cup \mathrm{C_{spe}} \neq \mathrm{Sub}(F) \cup \mathrm{C_{spe}}$. Then $F = E, s$, with $\mathrm{Sub}(F) = \mathrm{Sub}(E) \cup \{s\}$ and $\mathrm{Sign}(d) = \mathrm{Sign}(s)$.*

PROOF. Let s be the term deduced by the deduction rule. By contradiction let us first assume that $\mathrm{Sign}(s) \neq \mathrm{Sign}(d)$. By Lemma 15 this implies that $s \in \mathrm{Sub}(E)$. Since $\mathrm{Sub}(F) = \mathrm{Sub}(E) \cup \mathrm{Sub}(s)$ this contradicts $\mathrm{Sub}(E) \neq \mathrm{Sub}(F)$, and thus $\mathrm{Sign}(s) = \mathrm{Sign}(d)$.

In this case, by Lemma 16, all factors of s are in $\mathrm{Sub}(E) \cup \mathrm{C_{spe}}$. Thus by definition of the subterm relation, we have $(\mathrm{Sub}(F) \cup \mathrm{C_{spe}}) \setminus (\mathrm{Sub}(E) \cup \mathrm{C_{spe}}) = s$. $\qquad\square$

The properties of $\langle \mathcal{R} \rangle$ allow to state the following lemma:

Lemma 18 *Let D be a derivation and $l \to r \in \mathrm{GI}(d)$ a rule applied in D. Then for each $s \in l$ if there is a rule $l_s \to s \in \mathrm{GI}(d')$ applied in D then we can assume d and d' are rules of different deduction systems $\mathcal{D}_1$ or $\mathcal{D}_2$.*

PROOF. By contradiction. Let $\mathcal{D}$ be the set of derivations for which the lemma does not hold and let D be in $\mathcal{D}$ with a minimal number of rules application that does not satisfy the lemma. Let $l \to r \in \mathrm{GI}(d)$ be the first of these rules. For any term $s \in l$ such that there exists a rule $l_s \to s \in \mathrm{GI}(d')$ in D we do the following construction:

Since d and d' are in the same deduction system wlog we can assume $d, d' \in \langle \mathcal{R}_1 \rangle$, and since the result of the ground instance d' is used in the left-hand side of the ground instance of d, there exists a unifier between the result of d' and a term in the left-hand side of d. Thus by construction $\langle \mathcal{R}_1 \rangle$ contains a rule that has a ground instance:

$$l_s, (l \setminus s) \to r$$

By iterating this construction on l we build a rule matching the criterion. This contradicts the minimality of D in $\mathcal{D}$. Thus $\mathcal{D}$ is empty. □

The following lemma is a direct consequence of Lemma 17 that will be used throughout this paper.

Lemma 19 *Let $D : E_0 \to \ldots \to E_n$ be a derivation where the E_i are normalized for $i \in \{1, \ldots, n\}$ and assume there exists $s \in \mathrm{Sub}(E_i) \setminus (\mathrm{Sub}(E_0) \cup \mathrm{C}_{\mathrm{spe}})$. Then there exists in D a step $E_{j-1} \to_{l_s \to s} E_j$ with $j \leq i$ and $l_s \to s \in \mathrm{GI}(d)$ with $\mathrm{Sign}(d) = \mathrm{Sign}(s)$.*

PROOF. Consider the minimal indice j such that $s \in \mathrm{Sub}(E_j)$. By hypothesis we have $j > 0$ and $j \leq i$. Moreover by minimality of j we have $\mathrm{Sub}(E_j) \neq \mathrm{Sub}(E_{j-1})$. Since $s \notin \mathrm{C}_{\mathrm{spe}}$ Lemma 17 implies that $E_j = E_{j-1}, s$, and that if $E_{j-1} \to_{l_s \to s} = E_j$ with $l_s \to s \in \mathrm{GI}(d)$ then $\mathrm{Sign}(d) = \mathrm{Sign}(s)$. □

6.3 Well-formed derivations

A derivation $E_0 \to_{\mathcal{I}} E_1 = E_0, t_1 \to_{\mathcal{D}} \cdots \to_{\mathcal{I}} E_n = E_{n-1}, t_n$ of deduction system $\mathcal{D}$ is *well-formed* if for all $i \in \{1, \ldots, n\}$ we have $t_i \in \mathrm{Sub}(E_0, t_n) \cup \mathrm{C}_{\mathrm{spe}}$. In other words every message generated by an intermediate step either occurs in the goal, in the initial set of messages or is a special constant.

In the next lemma we assume that E and t are in normal form, that $\mathrm{c}_{\mathrm{min}} \in E$ and that all terms produced during a derivation are in normal form.

Lemma 20 *A derivation of minimal length starting from E of goal t is well-formed.*

PROOF. By contradiction. Let n be the length of D, and let $\{t_1, \ldots, t_n\}$ be such that $t = t_n$ and

$$D : E \to E, t_1 \to E, t_1, t_2 \to \ldots \to E, t_1, \ldots, t_n$$

Since D is minimal, it is without stutter. Let us assume that the property doesn't hold, and let i be the maximal indice such that $t_i \notin \mathrm{Sub}(E_0, t_n) \cup C_{\mathrm{spe}}$. This implies $t_i \neq t_n$ and thus $i < n$. Since the derivation is without stutter, Lemma 19 yields that $t_i \notin \mathrm{Sub}(E_0) \cup C_{\mathrm{spe}}$ implies $t_i \notin \mathrm{Sub}(E_{i-1}) \cup C_{\mathrm{spe}}$. Let $L_i \to t_i \in \mathrm{GI}(d)$ such that $E_{i-1} \to_{L_i \to t_i} E_i$. By Lemma 17 this implies $\mathrm{Sign}(d) = \mathrm{Sign}(t_i)$.

By minimality of D, the term t_i must be used in the left-hand side of a subsequent step in the derivation (otherwise the step producing t_i can be avoided). Let Ω be the set of step indices i where t_i has to be used in left-hand side. More precisely, let

$$\Omega = \{h \,|\, E_{h-1} \to E_h \in D \text{ and } E_{h-1} \setminus \{t_i\} \not\to E_h \setminus \{t_i\}\}$$

Ω is non-empty from our previous assumption. Let j be the minimum element of Ω and let $l_j \to t_j \in \mathrm{GI}(d')$ be the j-th deduction rule in derivation D. Note that $j > i$ and thus by maximality of i we have $t_j \in \mathrm{Sub}(E_0, t_n)$. Therefore $t_i \notin \mathrm{Sub}(t_j)$. Moreover $j \in \Omega$ implies $t_i \in l_j$. By Lemma 18 we can assume d and d' are not in the same theory. Since $\mathrm{Sign}(d) = \mathrm{Sign}(t_i)$ we have $\mathrm{Sign}(t_i) \neq \mathrm{Sign}(d')$. Thus by Lemma 14 there exists a rule $(l_j \delta_{t_i})\!\downarrow \to (t_j \delta_{t_i})\!\downarrow$ in GI(d'). By minimality of j the only term $r \in E_{j-1}$ such that $t_i \in \mathrm{Sub}(r)$ is t_i itself and thus $(l_j \delta_{t_i})\!\downarrow = l_j, c_{\min} \setminus \{t_i\}$. Since $t_i \notin \mathrm{Sub}(t_j)$ we have $(t_j \delta_{t_i})\!\downarrow = t_j$. Thus there exists in GI(d') a rule $l_j' \to t_j$ with $l_j' \subseteq E_{j-1}$ and $t_i \notin E_j$. This contradicts again $j \in \Omega$, therefore Ω is empty which shows that our initial assumption cannot hold.   □

7 Decidability results

The main result of this paper is a modularity result that can be stated as follows:

Theorem 1 *If the ordered satisfiability problem for closed symbolic derivation is decidable for two deduction systems $\langle \mathcal{F}_1, S_1, \mathcal{E}_1 \rangle$ and $\langle \mathcal{F}_2, S_2, \mathcal{E}_2 \rangle$ with disjoint signatures $\mathcal{F}_1$ and $\mathcal{F}_2$ then the ordered satisfiability problem for closed symbolic derivation is decidable for the deduction system $\langle \mathcal{F}_1 \cup \mathcal{F}_2, \langle S_1 \rangle \cup \langle S_2 \rangle, \mathcal{E}_1 \cup \mathcal{E}_2 \rangle$.*

This result is obtained as a consequence of Algorithm 1 reducing the resolution of a $\mathcal{D}$-symbolic derivation $\mathcal{C}$ to the resolution of $\mathcal{D}_1$ and $\mathcal{D}_2$ symbolic derivations with the addition of ordering constraints among the variables of $\mathcal{C}$ to the already present ordering constraints between variables and constants of $\mathcal{C}$. The proof of the soundness and completeness of this algorithm relies on the results given in Section 9.

In [9] we have defined the *deterministic* symbolic derivations that model the execution of protocols in which a message sent by an honest agent is uniquely determined (up to nonce's values) by the messages received so far. Here we rather introduce symbolic derivations to model protocol instances in which roles are defined by the sequence of deductions applied on

the received messages to build the responses. The difference with the deterministic systems is that now we introduce new variables that keep track of the deductions performed by honest agents when they compose messages at each step.

8 Bound solutions of symbolic derivations

We recall that the deduction system $\mathcal{D}$ is the union of the two deduction systems $\langle \mathcal{F}_1, S_1, \mathcal{E}_1 \rangle$ and $\langle \mathcal{F}_2, S_2, \mathcal{E}_2 \rangle$. In the rest of this section we first prove in Subsection 8.1 that if s is a free subterm of σ then replacing it by $c_{\min}$ in derivations yields new derivations (*i.e.* no deduction power is lost.) Then we prove in that if $\mathcal{C}$ is satisfiable there exists a solution σ of $\mathcal{C}$ which is constructed from the subterms of the input problem (Subsection 8.2).

8.1 Stability of derivations by replacement of free subterms

First we prove that when replacing a free term s in σ by the constant $c_{\min}$ we still obtain a derivation.

The next lemma is a one-step version of Lemma 22.

Lemma 21 *Let G be a finite set of normalised terms with $c_{\min} \in G$, let r and s be two normalised terms with $s \notin C_{\mathrm{spe}}$, let l_r be the rule $r_1, \dots, r_n \to r \in \mathrm{GI}(u)$, l_s be the rule $s_1, \dots, s_m \to s \in \mathrm{GI}(v)$. Assume moreover $\mathrm{Sign}(v) = \mathrm{Sign}(s)$ and:*

$$G \to_{l_s} G, s \to_{l_r} G, r, s$$

Then either $(r\delta_s)\!\downarrow \in (G\delta_s)\!\downarrow$ or:

$$(G\delta_s)\!\downarrow \to_{\mathcal{I}} (G\delta_s)\!\downarrow, s, (r\delta_s)\!\downarrow$$

PROOF. Assume $(r\delta_s)\!\downarrow \notin (G\delta_s)\!\downarrow$ and thus $s \neq r$. By Lemma 18 we can safely assume $\mathrm{Sign}(u) \neq \mathrm{Sign}(v)$. Thus the assumption $\mathrm{Sign}(s) = \mathrm{Sign}(u)$ implies $\mathrm{Sign}(s) \neq \mathrm{Sign}(v)$. The result is then a trivial consequence of Lemma 14. □

Lemma 22 will be applied with s a free term in a solution σ in Lemma 24. It permits us to characterise minimal solutions of a constraint satisfaction problem.

Lemma 22 *Let E and F be finite sets of normalised terms with $c_{\min} \in E$. Let s, t be two normalised terms not in C_{spe} with $s \in \mathrm{Der}(E) \setminus \mathrm{Sub}(E)$ and $t \in \mathrm{Der}(E \cup F)$. We have:*

$$(t\delta_s)\!\downarrow \in \mathrm{Der}(((E \cup F)\delta_s)\!\downarrow)$$

PROOF. By assumption there exists:

- a derivation $D_t : E \cup F \to^* G_{E \cup F} \to G_{E \cup F}, t$ with $t \notin G_{E \cup F}$;

- a derivation $D_s : E \to^* G_E \to G_E, s$ with $s \notin G_E$

and therefore one can construct the derivation:

$$D : E \cup F \to^* F \cup G_E \to F \cup G_E, s \to^* G_E, G_{E \cup F}, s \to G_E, G_{E \cup F}, s, t$$

where the instantiated deduction rules applied up to the deduction of s are from D_s and those applied thereafter are from D_t. Let $t_1, \dots, t_n$ be the sequence of the terms deduced along the derivation D, and i_s be minimal such that $t_{i_s} = s$, and $G_0, \dots, G_n$ be the sequence of knowledge sets.

By contradiction, assume the lemma does not hold, and thus $(t_n \delta_s) \!\downarrow\; \notin \mathrm{Der}(((E \cup F) \delta_s) \!\downarrow)$. This implies that the set of indices i such that $(t_i \delta_s) \!\downarrow\; \notin \mathrm{Der}(((E \cup F) \delta_s) \!\downarrow)$ is not empty. Let i_0 be minimal in this set. Since $s \notin G_E$ and $s \notin \mathrm{Sub}(E)$, Lemma 19 implies that $s \notin \mathrm{Sub}(G_E)$. This has two consequences:

- the rule applied to deduce s in D is from the same signature as s;

- we have $i_0 \geq i_s$, and since $c_{\min} \in E$ we have $i_0 > i_s$.

From this we conclude that $s \in G_{i_0 - 1}$, and that there exists a sequence of two deductions:

$$G_{i_0 - 1} \setminus \{s\} \to_{l_1} G_{i_0 - 1} \to_{l_2} G_{i_0 - 1}, t_{i_0}$$

where l_1 (resp. l_2) is the rule applied in D to deduce s (resp. t_{i_0}). By Lemma 21 this implies that either $(t_{i_0} \delta_s) \!\downarrow\; \in (G_{i_0 - 1} \delta_s) \!\downarrow$ or that there exists a deduction $(G_{i_0 - 1} \delta_s) \!\downarrow\; \to (G_{i_0 - 1} \delta_s) \!\downarrow, (t_{i_0} \delta_s) \!\downarrow$. Both cases contradict the fact that $(t_{i_0} \delta_s) \!\downarrow\; \notin \mathrm{Der}(((E \cup F) \delta_s) \!\downarrow)$, which proves the lemma.

$\square$

8.2   Existence and properties of bound solutions

Let $\mathcal{C}$ be a symbolic derivation and $K_{\mathcal{D}}$ be the initial knowledge of an intruder trying to mount an attack against $\mathcal{C}$. From now we say a ground term is *bound* if it is bound by σ in $\mathrm{Sub}(\mathcal{C} \cup K_{\mathcal{D}})$, unless otherwise specified. In the same way a term is *free* if it is free in $\mathrm{Sub}(\mathcal{C} \cup K_{\mathcal{D}})$. We say a substitution σ is bound if for all variables x in its support all terms in $\mathrm{Sub}(x\sigma)$ are bound by σ in $\mathrm{Sub}(\mathcal{C} \cup K_{\mathcal{D}})$.

The proof of the following lemma relies on the hypothesis above that σ is normal.

Lemma 23 *Let $\mathcal{C}_c = (\mathcal{V}, \mathcal{S}, \mathcal{K})$ be a closed derivation with $\mathcal{V} = (x_1, \dots, x_m)$, and sigma be a ground normal substitution that satisfies $\mathcal{C}_c$. Let $s \notin \mathrm{C}_{\mathrm{spe}}$ be a term such that $s \in \mathrm{Sub}((x_k \sigma) \!\downarrow)$ for some $k \in Ind$. Then there exists*

- *an equation $x_j \overset{?}{=} t$ in $\mathcal{S}$ with t ground and $s \in \mathrm{Sub}(t)$ or*

- *$j \leq_{Ind} k$ such that $(x_j \sigma) \!\downarrow = s$ and x_j is the right-hand side of a rule d with $\mathrm{Sign}(d) = \mathrm{Sign}(s)$.*

PROOF. Let k be minimal in Ind such that $s \in \mathrm{Sub}((x_k\sigma)\downarrow)$. If the first term of the alternative does not hold, and since there is no input variable, by definition of symbolic derivation we have that x_k is the right-hand side of a deduction rule applied on $x_{\beta_1}, \ldots, x_{\beta_l}$ with $\beta_j <_{Ind} i$ for $1 \le j \le l$, and $s \in \mathrm{Sub}(x_k\sigma)$. Let $E = x_{\beta_1}\sigma, \ldots, x_{\beta_l}\sigma$. We have $s \notin \mathrm{Sub}(E) \cup \mathrm{C_{spe}}$ and $E \to E, x_j\sigma$ with $s \in \mathrm{Sub}(x_j\sigma) \setminus \mathrm{C_{spe}}$. We conclude with Lemma 19. $\square$

We now prove that if $\mathcal{C}$ is satisfiable then it is satisfied by a bound substitution. First we prove it is possible to replace one free term s by the minimal constant.

Lemma 24 *If there exists $\mathcal{C}_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$ and σ such that $\sigma \models \mathcal{C} \circ \mathcal{C}_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$ and $x \in \mathrm{Var}(\mathcal{C})$, $s \in \mathrm{Sub}(x\sigma)$ such that s is free in $\mathrm{Sub}(\mathcal{C} \cup \mathcal{K}_{\mathcal{D}})$ for σ, then there exists $\mathcal{C}'_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$ such that $(\sigma\delta_s)\downarrow \models \mathcal{C} \circ \mathcal{C}'_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$*

PROOF. Let $\mathcal{C}$, $\mathcal{K}_{\mathcal{D}}$ and σ be such that there exists a symbolic derivation $\mathcal{C}_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$ with $\sigma \models \mathcal{C} \circ \mathcal{C}_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$ and $s \in \mathrm{Sub}(\sigma)$ is free. Let $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K}, \mathrm{In}, \mathrm{Out})$, and $\mathcal{C}_c = \mathcal{C} \circ \mathcal{C}_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$, and $\sigma' = (\sigma\delta_s)\downarrow$, and:

$$\begin{cases} I & = & (v_i)_{i\in\mathrm{IN}} \\ O_i & = & \{v_j \,|\, j \in \mathrm{OUT} \text{ and } j < i\} \end{cases}$$

Let us first note that $\sigma \models \mathcal{S}$ and s free in $\mathrm{Sub}(\mathcal{S})$ imply by Lemma 12 that $\sigma' \models \mathcal{S}$. Thus to prove the existence of $\mathcal{C}'_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$ such that $\sigma' \models \mathcal{C} \circ \mathcal{C}'_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$ it suffices to prove that for all $i \in \mathrm{IN}$ we have $v_i\sigma' \in \mathrm{Der}(\mathcal{K}_{\mathcal{D}} \cup O_i\sigma')$. Since $\mathcal{C}_c$ is a closed symbolic derivation and $s \notin \mathrm{Sub}(\mathcal{C} \cup \mathcal{K}_{\mathcal{D}})$ by Lemma 23 there exists a variable v_s of $\mathcal{C}_c$ such that $v_s\sigma = s$ and for all variables x of $\mathcal{C}_c$ of rank lower than v_s we have $s \notin \mathrm{Sub}(x\sigma)$. Since s is free in $\mathrm{Sub}(\mathcal{C})$ this variable corresponds to an intruder deduction. Id est, there exists i_s such that $s \notin \mathrm{Sub}(O_{i_s}\sigma \cup \mathcal{K}_{\mathcal{D}})$ and $s \in \mathrm{Der}(O_{i_s}\sigma \cup \mathcal{K}_{\mathcal{D}})$. For $i \ge i_s$, applying Lemma 22 with $E = O_{i_s}\sigma \cup \mathcal{K}_{\mathcal{D}}$, and $F = O_i\sigma \cup \mathcal{K}_{\mathcal{D}}$, and $t = v_i\sigma$ yields $v_i\sigma' \in \mathrm{Der}(O_i\sigma' \cup \mathcal{K}_{\mathcal{D}})$. For $i < i_s$ the result is trivial since in this case $v_i\sigma' = v_i\sigma$ and $O_i\sigma' = O_i\sigma$ by minimality of i_s. Thus there exists a symbolic derivation $\mathcal{C}'_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$ such that $\sigma' \models \mathcal{C} \circ \mathcal{C}'_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$.

$\square$

The proof of next Proposition 1 is a direct consequence of Lemma 24 and exploits the well-foundedness of the order $<$ to prove it is possible to iteratively replace all free subterms.

Proposition 1 *Let $\mathcal{C}$ be a satisfiable symbolic derivation. There exists a normal bound substitution σ such that $\sigma \models \mathcal{C}$.*

PROOF. Consider the set Σ of normal substitutions that satisfy $\mathcal{C}$. By hypothesis Σ is not empty. Let σ be a minimal substitution in Σ for the total ordering $<$ on ground terms extended on substitutions seen as multisets of ground terms. Let us prove σ is bound to $\mathcal{C}$.

By contradiction assume there exists s free in $\mathrm{Sub}(\sigma)$ and let $\sigma' = (\sigma\delta_s)\downarrow$. By Lemma 24 we also have $\sigma' \models \mathcal{C}$. By monotony of $<$, $\mathrm{c_{min}} < s$ implies $\sigma' < \sigma$. By definition of R we have $(\sigma')\downarrow \le \sigma'$. Thus $(\sigma')\downarrow \in \Sigma$ and $(\sigma')\downarrow < \sigma$ which contradicts the minimality of σ. $\square$

Lemma 25 *If σ is a bound substitution such that there exists a symbolic derivation $\mathcal{C}_\mathcal{D}(\mathcal{K}_\mathcal{D})$ with $\sigma \models \mathcal{C} \circ \mathcal{C}_\mathcal{D}(\mathcal{K}_\mathcal{D})$ then there exists $\mathcal{C}_\mathcal{D}(\mathcal{K}_\mathcal{D})$ such that:*

- $\mathrm{Sub}((\mathrm{Sub}(\mathcal{C} \cup \mathcal{K}_\mathcal{D})\sigma)\!\downarrow) = (\mathrm{Sub}(\mathcal{C} \cup \mathcal{K}_\mathcal{D})\sigma)\!\downarrow;$

- *For all derivation variable x of $\mathcal{C}_\mathcal{D}(\mathcal{K}_\mathcal{D})$ one has $x\sigma \in \mathrm{Sub}((\mathcal{C}\sigma \cup \mathcal{K}_\mathcal{D})\!\downarrow)$.*

PROOF. Let us prove the first point. Let $S = (\mathrm{Sub}(\mathcal{C} \cup \mathcal{K}_\mathcal{D})\sigma)\!\downarrow$. We have $S \subseteq \mathrm{Sub}(S)$. The converse inclusion $\mathrm{Sub}(S) \subseteq S$ follows directly from:

$$\mathrm{Sub}((\mathrm{Sub}(\mathcal{C} \cup \mathcal{K}_\mathcal{D})\sigma)\!\downarrow) \subseteq (\mathrm{Sub}(\mathcal{C} \cup \mathcal{K}_\mathcal{D})\sigma)\!\downarrow \cup \mathrm{Sub}(\mathrm{Var}(\mathcal{C} \cup \mathcal{K}_\mathcal{D})\sigma) \cup \mathrm{C}_{\mathrm{spe}}$$

Since σ is bound we have $\mathrm{Sub}(\mathrm{Var}(\mathcal{C} \cup \mathcal{K}_\mathcal{D})\sigma) \subseteq (\mathrm{Sub}(\mathcal{C} \cup \mathcal{K}_\mathcal{D})\sigma)\!\downarrow$ and by hypothesis we have $\mathrm{C}_{\mathrm{spe}} \subseteq \mathrm{Sub}(\mathcal{C})$. The second point is a direct consequence of the first point and of the existence of well-formed derivations (*i.e.* Lemma 20). □

Before presenting the combination algorithm for Theorem 1 let us terminate with lemmas stating how a closed symbolic derivation can be split into symbolic derivations on two disjoint signatures, and how intruder deduction rules may be eliminated and replaced by output/input variables.

The abstraction $\mathrm{Abs}_i(\sigma)$ of a substitution σ in $\mathcal{F}_i$ is the substitution such that for all x in the support of σ, $x\mathrm{Abs}_i(\sigma)$ is the abstraction of $x\sigma$ in $\mathcal{F}_i$. In the statement of next lemma, $\mathrm{Sign}(t)$ designates the signature to which belongs the top symbol of the term t. From a system of equations $\mathcal{S}$, one notes that by introducing new variables one can derive an equisatisfiable homogeneous set of equations $\mathcal{S}_1 \cup \mathcal{S}_2$ such that terms in $\mathcal{S}_i$ are pure $\mathcal{F}_i$-terms for $i = 1, 2$.

The proof idea of Lemma 26 is to "abstract" a solution σ for a symbolic derivation $\mathcal{C}$ in deduction system $\langle \mathcal{F}_1 \cup \mathcal{F}_2, S_1 \cup S_2, \mathcal{E}_1 \cup \mathcal{E}_2 \rangle$ and to apply iteratively Lemma 14 in order to show for $i = 1, 2$ that $\mathrm{Abs}_i(\sigma)$ is a solution of the symbolic derivation in the deduction system $\langle \mathcal{F}_i, S_i, \mathcal{E}_i \rangle$.

Lemma 26 *Let $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K})$ be a closed derivation satisfied by a normal substitution σ for the deduction system $\langle \mathcal{F}_1 \cup \mathcal{F}_2, S_1 \cup S_2, \mathcal{E}_1 \cup \mathcal{E}_2 \rangle$. For $i = 1, 2$ let $\mathcal{S}_i$ denote the purification of $\mathcal{S}$ and:*

$$\mathcal{K}_i = \{\mathrm{Abs}_i(t) \in \mathrm{Sub}(\mathcal{K}) \mid \mathrm{Sign}(t) = i \text{ and } \exists x \in \mathcal{V}, \ x\sigma = t\} \cup \{x \in \mathcal{V} \mid \mathrm{Sign}(x\sigma) \neq i\}$$

Then two closed symbolic derivations $\mathcal{C}_i = (\mathcal{V}, \mathcal{S}_i, \mathcal{K}_i)$ (for $i = 1, 2$) can be computed such that they are satisfied by $\mathrm{Abs}_i(\sigma)$ for intruder $\langle \mathcal{F}_i, S_i, \mathcal{E}_i \rangle$ respectively.

PROOF. By symmetry it suffices to prove that $(\mathcal{V}, \mathcal{S}_1, \mathcal{K}_1)$ is a closed symbolic derivation. Let x be a variable in $\mathcal{V}$ and i be the rank of x. Since $\mathcal{C}$ is closed, by Lemma 23 we have:

- there is an equation $x \stackrel{?}{=} t$ in $\mathcal{S}$ with t ground, or

- x is the result of a rule d with $\mathrm{Sign}(d) = \mathrm{type}(w)$.

If $\text{Sign}(d) = \mathcal{F}_1$ then all equations corresponding to this rule are pure $\mathcal{F}_1$ equations and thus are present in $\mathcal{S}_1$. On the other hand, if $\text{Sign}(d) = \mathcal{F}_2$ then we also have $\text{Sign}(x\sigma) = \mathcal{F}_2$ and thus x is abstracted by a constant, and therefore as a ground term.

Finally the fact that C_i is satisfied by the abstraction on signature $\mathcal{F}_i$ of σ (for $i \in \{1, 2\}$) is a consequence of repeated applications of Lemma 14 to replace maximal alien subterms (factors or the term itself) of the $x\sigma$, for $x \in \mathcal{V}$ by the corresponding abstraction constants. $\square$

Lemma 27 *Let $K_{\mathcal{D}}$ be a finite set of ground terms and $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K}, \text{IN}, \text{OUT})$ be a symbolic derivation such that there exists σ and $\mathcal{C}_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$ with $\sigma \models \mathcal{C} \circ \mathcal{C}_{\mathcal{D}}(K_{\mathcal{D}})$. Let $\mathcal{S}_d \subseteq \mathcal{S}$ be a set of equations corresponding to the application of rule d with lhs indices $I \subseteq Ind$ and rhs indice $i \in Ind$. Then:*

$$\mathcal{C}' = (\mathcal{V}, \mathcal{S} \setminus \mathcal{S}_d, \mathcal{K}, \text{IN} \cup \{i\}, \text{OUT} \cup I)$$

is a symbolic derivation such that there exists $\mathcal{C}'_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$ with $\sigma \models \mathcal{C} \circ \mathcal{C}'_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$.

PROOF. Add $\mathcal{S}_d$ to $\mathcal{C}_{\mathcal{D}}(K_{\mathcal{D}})$ together with additional variables matching the IN and OUT changes. $\square$

9 Deriving pure symbolic derivations

In Algorithm 1, a *partial* symbolic derivation designates a symbolic derivation for which variables in $\mathcal{V}$ whose index is not in IN may have no associated equation in $\mathcal{S}$ (may not be a left-hand side of an equation in $\mathcal{S}$).

Theorem 2 *(Completeness) If $\mathcal{C}$ is satisfiable then Algorithm 1 returns* SAT.

PROOF. Assume $\mathcal{C}$ is satisfiable. By Proposition 1 there exists a normal substitution σ bound by σ in $\text{Sub}(\mathcal{C}) \cup \mathcal{K}_{\mathcal{D}}$ and a symbolic intruder derivation $\mathcal{C}_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$ such that $\sigma \models \mathcal{C} \circ \mathcal{C}_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$. By Lemma 25 we can assume that for all derivation variable x of $\mathcal{C}_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$ one has $x\sigma \in \text{Sub}((\mathcal{C}\sigma \cup \mathcal{K}_{\mathcal{D}})\downarrow)$.

Let $\mathcal{C}_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}}) = (\mathcal{V}_{\mathcal{D}}, \mathcal{S}_D, \mathcal{K}_{\mathcal{D}}, \text{IN}_{\mathcal{D}}, \text{OUT}_{\mathcal{D}})$. Since the composed symbolic derivation is closed, we have $|\text{IN}_{\mathcal{D}}| = |\text{OUT}|$ and $|\text{OUT}_{\mathcal{D}}| = |\text{IN}|$. Re-using already deduced terms, one easily sees that the number of different variables of $\mathcal{V}_{\mathcal{D}}$ corresponding to intruder deduction rules is bounded by $|(\mathcal{C}\sigma \cup \mathcal{K}_{\mathcal{D}})\downarrow|$ and thus by $|\text{Sub}(\mathcal{C} \cup \mathcal{K}_{\mathcal{D}})|$. The number of positions in $\mathcal{V}_{\mathcal{D}}$ equal to a ground term is bounded by $\mathcal{K}_{\mathcal{D}}$, the number of input positions is $|\text{OUT}|$. Finally we allow for $|\text{IN}|$ other positions to enable the intruder to send several times the same term. Thus we can assume

$$|Ind| \leq |\text{Sub}(\mathcal{C}) + |\text{IN}| + |\text{OUT}| + |\mathcal{K}_{\mathcal{D}}|$$

The substitution σ defines an equivalence relation $\equiv_\sigma$ on $\text{Sub}(\mathcal{C} \cup \mathcal{K}_{\mathcal{D}})$ by setting $t \equiv_\sigma s$ iff $(t\sigma)\downarrow = (s\sigma)\downarrow$. Let X be a set of new variables representing the equivalence classes of $\equiv_\sigma$. We also designates σ the extension of σ to X. Let $\mathcal{S}_{\mathcal{D}}$ be the set of all purified equations $t \stackrel{?}{=} x$ for $t \in \text{Sub}(\mathcal{C} \cup \mathcal{K}_{\mathcal{D}})$, where x is the representant of the equivalence class of t. The

---

Algorithm 1 Combination Algorithm

Solve$_{\mathcal{D}}(\mathcal{C}, \prec)$

Input: $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K}, \text{IN}, \text{OUT})$ where $\mathcal{S}$ homogeneous;
A finite set of ground terms $K_{\mathcal{D}}$;
A linear ordering $\prec$ on $\text{Var}(\mathcal{C}) \cup \text{Const}(\mathcal{C})$.

1: Choose a partial symbolic derivation $\mathcal{C}_{\mathcal{D}} = (\mathcal{V}_{\mathcal{D}}, \mathcal{S}_{\mathcal{D}}, \mathcal{K}_{\mathcal{D}}, \text{IN}_{\mathcal{D}}, \text{OUT}_{\mathcal{D}})$ such that:

- $|\mathcal{V}_{\mathcal{D}}| \leq |\text{Sub}(\mathcal{C})| + |\text{IN}| + |\text{OUT}| + |\mathcal{K}_{\mathcal{D}}|$

- $|\text{IN}_{\mathcal{D}}| = |\text{OUT}|$ and $|\text{OUT}_{\mathcal{D}}| = |\text{IN}|$

- The composition $\mathcal{C} \circ \mathcal{C}_{\mathcal{D}}(\mathcal{K}_{\mathcal{D}})$ is defined

- $\mathcal{S}_{\mathcal{D}}$ is homogeneous, contains equations $x \overset{?}{=} t$ with $t \in \text{Sub}(\mathcal{C}) \cup \mathcal{V}_{\mathcal{D}}$ and $x \in X$, a set of new variables with $|X| \leq |\text{Sub}(\mathcal{C}) \cup \mathcal{V}_{\mathcal{D}}|$, and defines an equivalence relation $\equiv_{\mathcal{S}_{\mathcal{D}}}$ on $\text{Sub}(\mathcal{C}) \cup \mathcal{V}_{\mathcal{D}}$

2: Choose a linear ordering $\prec_X$ on variables in $X \cup \text{Const}(\mathcal{C}) \cup \text{Var}(\mathcal{C})$ extending $\prec$. Let X_1 and X_2 be two disjoint subsets of X
3: Form the closed partial symbolic derivation $\mathcal{C}' = \mathcal{C} \circ \mathcal{C}_{\mathcal{D}}$, and purify it into two pure symbolic derivations $\mathcal{C}_1$ and $\mathcal{C}_2$, where variables of X_1 (resp. X_2) are considered as constants in $\mathcal{C}_2$ (resp. $\mathcal{C}_1$).
4: If **Solve**$_{\mathcal{D}_1}(\mathcal{C}_1, \prec_X)$ and **Solve**$_{\mathcal{D}_2}(\mathcal{C}_2, \prec_X)$ return SAT else FAIL

---

newly introduced variables are employed to purify the equations $t \overset{?}{=} x$ with $t \in \text{Sub}(\mathcal{K}_{\mathcal{D}})$ which are the only non-pure ones. By definition of $\mathcal{S}_{\mathcal{D}}$ one easily sees that the symbolic derivation $\mathcal{C}_\sigma = (\mathcal{V}_{\mathcal{D}}, \mathcal{S}_D \cup \mathcal{S}_{\mathcal{D}}, \mathcal{K}_{\mathcal{D}}, \text{IN}_{\mathcal{D}}, \text{OUT}_{\mathcal{D}})$ can also be composed with $\mathcal{C}$ to obtain a closed symbolic derivation satisfied by σ.

Let us choose in the first step of the algorithm the partial symbolic derivation leaving $\mathcal{S}_D$ aside, *i.e.*:

$$\mathcal{C}_{\mathcal{D}} = (\mathcal{V}_{\mathcal{D}}, \mathcal{S}_{\mathcal{D}}, \mathcal{K}_{\mathcal{D}}, \text{IN}_{\mathcal{D}}, \text{OUT}_{\mathcal{D}})$$

We let $\prec_X$ be any linear ordering compatible with $\prec$ and the subterm relation on $X\sigma$. We let X_1 (resp. X_2) be the subset of X of variables x with $\text{Sign}(x\sigma) = \mathcal{F}_1$ (resp. $\mathcal{F}_2$).

We note that $\mathcal{C} \circ \mathcal{C}_\sigma$ is a closed symbolic derivation. Thus, by Lemma 26 the symbolic derivations $\mathcal{C}_{1,\sigma}$ and $\mathcal{C}_{2,\sigma}$ are also closed symbolic derivations satisfied respectively by $\text{Abs}_1(\sigma)$ and $\text{Abs}_2(\sigma)$. To complete the proof it then suffices to notice that iterated applications of Lemma 27 on $\mathcal{C}_{1,\sigma}$ (resp. $\mathcal{C}_{2,\sigma}$) yield the symbolic derivation $\mathcal{C}_1$ (resp. $\mathcal{C}_2$) guessed by the algorithm which thus is also satisfiable. The ordering $\prec_X$ is satisfied by σ by construction. □

To prove the correction, it suffices to note that the linear ordering $\prec_X$ permits to construct a solution σ from two partial solutions σ_1 and σ_2, and that one obtains deduction rules when replacing free constants by alien terms in normal form.

10   Application to Security Protocols

We refer to [6] for the definition of an intruder on words as well as for the proof of the decidability of its related ordered satisfiability problem. We will present here a similar result for an intruder on multi-sets of terms. We believe that the proof can be carried as is to prove the decidability of ordered satisfiabiliy problems for a deduction system operating on sets, when one allows for the set union and the extraction of a subset.

10.1   Multisets

We consider the signature $\mathcal{F} = \{+, 1\}$ with the following equational theory:

$$\mathcal{E} \quad \begin{cases} x + y &=& y + x \\ x + (y + z) &=& (x + y) + z \\ x + 1 &=& x \end{cases}$$

This is the associative-commutative theory with unit, usually denoted ACU. Notice however that the presence or absence of the last axiom does not affect our decidability result. We consider the deduction rules:

$$\mathcal{R} \quad \begin{cases} x + y &\rightarrow& x \\ x, y &\rightarrow& x + y \\ &\rightarrow& 1 \end{cases}$$

Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R}, \mathcal{E} \rangle$. First let us notice that these are indeed deduction rules since the theory is regular and the variables on the right-hand side of deduction rules also appear on the left-hand side. Let us prove a few results on the deduction with $\mathcal{D}$.

Proposition 2 *Let E and t be respectively a set of terms and a term in normal form modulo $\mathcal{E}$. We have:*

- $\mathrm{Const}(E) \subseteq \mathrm{Der}_{\mathcal{D}}(E)$ *and* $E \subseteq \mathrm{Der}_{\mathcal{D}}(\mathrm{Const}(E))$;

- $\mathrm{Der}_{\mathcal{D}}(E) = \mathrm{Der}_{\mathcal{D}}(\mathrm{Const}(E))$;

- $t \in \mathrm{Der}_{\mathcal{D}}(E)$ *iff* $\mathrm{Const}(t) \subseteq \mathrm{Const}(E)$

PROOF. Let us prove the points sequentially. Let $c \in \mathrm{Const}(E)$ be a constant, and $e \in E$ be a term such that $c \in \mathrm{Const}(e)$. We have therefore $e = e' + c$. Thus in one step we have $E \rightarrow E, c$. Since c is arbitrary, we have $\mathrm{Const}(E) \subseteq \mathrm{Der}_{\mathcal{D}}(E)$. We easily sees that we also have $E \in \mathrm{Der}_{\mathcal{D}}(\mathrm{Const}(E))$.

From this the second point follows by double inclusion. For the last point we have $t \in \mathrm{Der}_{\mathcal{D}}(E)$ is equivalent to $\mathrm{Der}_{\mathcal{D}}(E) = \mathrm{Der}_{\mathcal{D}}(E,t)$, and thus by the second point it is equivalent to $\mathrm{Der}_{\mathcal{D}}(\mathrm{Const}(E)) = \mathrm{Der}_{\mathcal{D}}(\mathrm{Const}(E,t))$. By contradiction assume there exists $c \in \mathrm{Const}(t) \setminus \mathrm{Const}(E)$. Given the constraint on constants in deduction rules we have $c \notin \mathrm{Der}_{\mathcal{D}}(\mathrm{Const}(E))$ whereas c trivially is in $\mathrm{Der}_{\mathcal{D}}(\mathrm{Const}(E,t))$, which contradicts the equality, and thus $t \in \mathrm{Der}_{\mathcal{D}}(E)$ □

The consequence of Proposition 2 is that, as far as sequences of deductions are concerned, only the set of constants appearing or not in the initial knowledge and the goal of a putative derivation are relevant to assess its feasibility. This has the important implication, with respect to decidability, that it is not necessary to know the exact instance of intruder's knowledge (its initial knowledge and the messages in the output of the symbolic derivation up to this point) and of the goal (the next input message of the symbolic derivation) to decide whether a derivation exists. It suffices to know the (guessable) sets of constants of the knowledge and of the goal. Finally, let us notice that unifiability modulo AC or ACU with ordering constraints can be decided by setting, for a variable x in which constants $a_1, \ldots, a_k$ must appear and constants $b_1, \ldots, b_l$ may not appear by replacing in all equations x by $x' + a_1 + \ldots + a_k$ and by setting the ordering constraints $x \prec b_1, \ldots, b_l$. These remarks permit to prove the correcteness of Algorithm 2.

10.2   Algorithm for ordered satisfiability

The completeness of this algorithm is trivial. The correctness is a result of Proposition 2.

Note on the complexity.   The first step can be done in NPTIME. The satisfiability of linear equation systems over positive integers is in NPTIME. The check in the third step can be performed in PTIME. Thus the ordered satisfiability for multi-set deduction systems is in NPTIME. We finally have:

Theorem 3 *Ordered satisfiability of multiset deduction systems is decidable in NPTIME.*

11   Conclusion

Web service XML messages are often vulnerable to rewriting attacks since the associated message format and processing model are quite tolerant to inclusion of new elements. We have developed a verification procedure that accounts for this and moreover can be extended with most existing protocol analysis procedures for algebraic operators. The combination algorithm applies in particular to encryption/decryption operators [17], list with associative concatenation [7], XOR [8], abelian groups [15]. Our framework allows to describe specific implementation of XML/XPath library and we aim at extending this work to take the various security standards for Web services. We also plan to implement it in the AVISPA platform [1].

Algorithm 2 Resolution of ordered statisfiability problems for multi-set deduction systems

Solve$_{\mathcal{D}}(\mathcal{C}, \prec)$

Input:

- $\mathcal{C} = (\mathcal{V}, \mathcal{S}, \mathcal{K}, \text{IN}, \text{OUT})$;

- A finite set of ground terms $K_{\mathcal{D}}$;

- An ordering constraint $\prec$ between $\text{Var}(\mathcal{C})$ and $\text{Const}(\mathcal{C})$.

1: For each variable x in $\mathcal{V}$ choose the set of constants $P_x = \{a_1, \ldots, a_k\}$ that will be present in the guessed solution. All these constants must be lower than x for the ordering $\prec$
2: Check the satisfiability of the unification system $\mathcal{S}$ by transforming it in a set of linear equations over $\mathbb{N}$ by setting, for each variable $v \in \mathcal{V}$:

$$v = \Sigma_{c \in P_v}(\lambda_{v,c} + 1)c$$

3: Check for each variable $v \in \text{IN}$ that

$$P_v \subseteq \text{Const}(\mathcal{K}_{\mathcal{D}}) \cup \bigcup_{\substack{v' \in \text{OUT} \\ v' <_{\mathcal{V}} v}} P_{v'}$$

4: If both checks are successful return SAT else FAIL

References

[1] A. Armando and et al. The AVISPA tool for the automated validation of internet security protocols and applications. In Kousha Etessami and Sriram K. Rajamani, editors, *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer, 2005.

[2] F. Baader and K. U. Schulz. Unification in the union of disjoint equational theories. combining decision procedures. *J. Symb. Comput.*, 21(2):211–243, 1996.

[3] David A. Basin, Sebastian Mödersheim, and Luca Viganò. Algebraic intruder deductions. In Geoff Sutcliffe and Andrei Voronkov, editors, *LPAR*, volume 3835 of *Lecture Notes in Computer Science*, pages 549–564. Springer, 2005.

[4] K. Bhargavan, C. Fournet, A. D. Gordon, and R. Pucella. Tulafale: A security tool for web services. In *Formal Methods for Components and Objects*, volume 3188 of *Lecture Notes in Computer Science*, pages 197–222. Springer, 2003.

[5] I. Cervesato, N. A. Durgin, P. Lincoln, J. C. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *CSFW*, pages 55–69, 1999.

[6] Y. Chevalier and M. Kourjieh. A symbolic intruder model for hash-collision attacks. Technical report, IRIT, 2006. `ftp://ftp.irit.fr/IRIT/LILAC/main.pdf`.

[7] Y. Chevalier and M. Kourjieh. A symbolic intruder model for hash-collision attacks. In *11th Annual Asian Computing Science Conference*, Lecture Notes in Computer Science. Springer, 2006. `ftp://ftp.irit.fr/IRIT/LILAC/main.pdf`.

[8] Y. Chevalier, R. Kuesters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *Proceedings of the Logic In Computer Science Conference, LICS'03*, June 2003.

[9] Y. Chevalier and M. Rusinowitch. Combining intruder theories. In *Proc. of ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 639–651. Springer, 2005.

[10] Y. Chevalier and M. Rusinowitch. Combining intruder theories. Technical report, INRIA, 2005. `http://www.inria.fr/rrrt/liste-2005.html`.

[11] H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In *Proceedings of RTA'05)*, Lecture Notes in Computer Science, Nara, Japan, April 2005. Springer.

[12] N. Dershowitz and J-P. Jouannaud. Rewrite systems. In *Handbook of Theoretical Computer Science, Volume B*, pages 243–320. Elsevier, 1990.

[13] D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.

[14] J. Goubault-Larrecq, M. Roger, and K. N. Verma. Abstraction and resolution modulo ac: How to verify diffie-hellman-like protocols automatically. *J. Log. Algebr. Program.*, 64(2):219–251, 2005.

[15] J. Millen and V. Shmatikov. Symbolic protocol analysis with an abelian group operator or Diffie-Hellman exponentiation. *Journal of Computer Security*, 2005.

[16] Maarten Rits and Mohammad Ashiqur Rahaman. Secure SOAP Requests in Enterprise SOA. In *ACSAC*, 2006.

[17] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc.14th IEEE Computer Security Foundations Workshop*, Cape Breton, Nova Scotia, June 2001.

[18] M. Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *J. Symb. Comput.*, 8(1/2):51–99, 1989.

[19] V. Shmatikov. Decidable analysis of cryptographic protocols with products and modular exponentiation. In *Proceedings of ESOP'04*, volume 2986 of *Lecture Notes in Computer Science*, pages 355–369,. Springer-Verlag, 2004.

[20] F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1), 1999.

[21] C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *16th International Conference on Automated Deduction*, volume 1632 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 1999.