Deciding knowledge in security protocols under equational theories

Martin Abadi, Véronique Cortier

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Deciding knowledge in security protocols under equational theories

Martín Abadi and Véronique Cortier

N° 5169

Avril 2004

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--5169--FR+ENG

INRIA
LORRAINE

Deciding knowledge in security protocols under equational theories

Martín Abadi* and Véronique Cortier **

Thème 2 — Génie logiciel
et calcul symbolique
Projet Cassis

Rapport de recherche n° 5169 — Avril 2004 — 22 pages

Abstract: The analysis of security protocols requires precise formulations of the knowledge of protocol participants and attackers. In formal approaches, this knowledge is often treated in terms of message deducibility and indistinguishability relations. In this paper we study the decidability of these two relations. The messages in question may employ functions (encryption, decryption, etc.) axiomatized in an equational theory. Our main positive results say that, for a large and useful class of equational theories, deducibility and indistinguishability are both decidable in polynomial time.

Key-words: decision, knowledge, cryptographic protocols, deduction, equivalence.

* Martín Abadi's work was partly supported by the National Science Foundation under Grants CCR-0204162 and CCR-0208800.

** Véronique Cortier's work was partly supported by the RNTL project PROUVE-03V360 and the European project AVISPA IST-2001-39252.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Décision de problèmes de connaissance pour des protocoles de sécurité sous des théories équationelles

Résumé : L'analyse des protocoles de sécurité requiert une formulation précise de la notion de connaissance pour les participants au protocole et pour l'attaquant. Dans les approches formelles, cette connaissance est en général décrite en terme de déduction de messages ou d'indistinguabilité. Dans cet article, nous étudions la décidabilité de ces deux notions. Les messages en question peuvent utiliser des fonctions (chiffrement, déchiffrement, etc.) axiomatisées dans une théorie équationnelle. Notre principal résultat positif est que, pour une large et utile classe de théories équationelles, la déduction et l'indistinguabilité sont toutes les deux décidables, en temps polynomial.

Mots-clés : décision, connaissance, protocoles cryptographiques, déduction, équivalence.

Deciding knowledge in security protocols under equational theories

Martín Abadi[1] and Véronique Cortier[2]

[1] Computer Science Department, University of California at Santa Cruz, USA
[2] Loria, INRIA & CNRS, Nancy, France

1 Introduction

Understanding security protocols often requires reasoning about the knowledge of legitimate protocol participants and attackers. As a simple example, let us consider a protocol in which A sends to B a message that consists of a secret s encrypted under a pre-arranged shared key k. One may argue that, after processing this message, B knows s. More interestingly, one may also argue than an attacker with bounded computing power that does not know k but eavesdrops on the communications between A and B and sees the message does not learn s.

Accordingly, formal methods for the analysis of security protocols rely on definitions of the knowledge of protocol participants and attackers. In those methods, the knowledge of an attacker is used to determine what messages the attacker can send at each point in time—it can send only messages it knows. Moreover, security guarantees can be phrased in terms of the knowledge of the attacker. For example, a guarantee might be that, at the end of a protocol run, the attacker does not know a particular key, or that the attacker does not know whether a certain ciphertext contains the plaintext "true" or "false". For such applications, although the attacker is typically an active entity that can learn by conducting experiments, the definition of knowledge focuses on a particular point in a protocol execution.

Many formal definitions explain the knowledge of an attacker in terms of message deduction (e.g., [16, 18, 20, 19]). Given a set of messages S and another message M, one asks whether M can be computed from S. The messages are represented by expressions, and correspondingly the computations allowed are symbolic manipulations of those expressions. Intuitively these computations can rely on any step that an eavesdropper who has obtained the messages in S can perform on its own in order to derive M. For example, the eavesdropper can encrypt and decrypt using known keys, and it can extract parts of messages.

Despite its usefulness in proofs about protocol behaviors, the concept of message deduction does not always provide a sufficient account of knowledge, and it is worthwhile to

* Martín Abadi's work was partly supported by the National Science Foundation under Grants CCR-0204162 and CCR-0208800.
** Véronique Cortier's work was partly supported by the RNTL project PROUVE-03V360 and the European project AVISPA IST-2001-39252.

consider alternatives. For instance, suppose that we are interested in a protocol that transmits an encrypted boolean value, possibly a different one in each run. We might like to express that this boolean value remains secret by saying that no attacker can learn it by eavesdropping on the protocol. On the other hand, it is unreasonable to say that an attacker cannot deduce the well-known boolean values "true" and "false". Instead, we may say that the attacker cannot distinguish an instance of the protocol with the value "true" from one with the value "false". More generally, we may say that two systems are equivalent when an attacker cannot distinguish them, and we may then express security guarantees as equivalences. The use of equivalences is common in computational approaches to cryptography (e.g., [15]), and it also figures prominently in several formal methods (e.g., [3, 17, 1]).

Two systems that output messages that an attacker can tell apart are obviously distinguishable. Conversely, in order to establish equivalences between systems, an important subtask is to establish equivalences between the messages that the systems generate (for example, between the encrypted boolean values). These equivalences may be called static equivalences, because they consider only the messages, not the dynamic processes that generate them. Bisimulation proof techniques can reduce process equivalences to static equivalences plus fairly standard bisimulation conditions [1] (see also [2, 8]).

In this paper we study the decidability of message deduction and static equivalence. We define a relation $\phi \vdash M$ that means that M can be deduced from ϕ, and a relation $\varphi \approx_s \psi$ that means that φ and ψ are statically equivalent; here ϕ, φ, and ψ are all essentially lists of messages, each with a name, represented by formal expressions. For generating these messages, we allow the application of a wide array of functions—pairing, projections, various flavors of encryption and decryption, digital signatures, one-way hash functions, etc.. Indeed, our results do not make any assumption on any particular cryptographic system beyond fairly general hypotheses on the form of the equational theory that is used for defining the properties of the cryptographic operations. Our main positive results assume only that the equational theory is defined by a convergent rewriting system with a finite number of rules of the form $M \to N$ where N is a proper subterm of M or a constant symbol. Such theories, which we call convergent subterm theories, appear frequently in applications. For them, we obtain that both $\phi \vdash M$ and $\varphi \approx_s \psi$ are decidable, in fact in polynomial time. For other equational theories, even decidable ones, we show that $\phi \vdash M$ and $\varphi \approx_s \psi$ can be undecidable. Moreover, we establish that $\vdash$ can be reduced to $\approx_s$ (not too surprisingly), but that the converse does not hold.

The problem of deciding knowledge is particularly important in the context of algorithms and tools for automated protocol analysis. Often, special techniques are introduced for particular sets of cryptographic operations of interest, on a case-by-case basis. For example, the classic Dolev-Yao result deals with a fixed, limited suite of public-key operations [14]; more recent decidability results deal with exclusive-or and modular exponentiation (e.g., [11, 10, 9]); many variants and combinations that arise in practice have not yet been explored. On the other hand, other algorithms and tools (e.g., [5–7]) allow much freedom in the choice of cryptographic operations but their analysis of the knowledge of the attacker is not always guaranteed to terminate. Decidability results under general equa-

tional theories have been rare. The most relevant previous work is that of Comon-Lundh and Treinen [12], who have studied the decidability of the deduction problem for a class of equational theories incomparable with ours. (For example, they allow the homomorphism property $\mathsf{enc}(\langle u, v\rangle, k) = \langle \mathsf{enc}(u,k), \mathsf{enc}(v,k)\rangle$ but not the inverse property $I(I(x)) = x$.) Simultaneously with our work (but independently), Delaune and Jacquemard [13] have shown that the deduction problem is decidable against an active attacker and under an equational theory which is included in ours. But none of them considered static equivalence.

The next section, section 2, introduces notations and definitions. Section 3 compares $\vdash$ and $\approx_s$. Section 4 focuses on convergent subterm theories and gives our main decidability results. Section 5 concludes and discusses the possible use of our results for automated analysis of security protocols. The main technical details appear in the appendix.

2 Basic definitions

Next we review definitions from previous work. In section 2.1 we give the syntax of expressions (following [1]). In section 2.2 we explain a representation for the information available to an observer who has seen messages exchanged in the course of a protocol execution. In section 2.3 and 2.4 we present the relations $\vdash$ and $\approx_s$, which (as explained in the introduction) provide two formalizations of the knowledge that the observer has on the basis of that information.

2.1 Syntax

A *signature* Σ consists of a finite set of function symbols, such as enc and pair, each with an arity. Let $\mathsf{ar}(\Sigma)$ be the maximal arity of a function symbol in Σ. A function symbol with arity 0 is a constant symbol.

Given a signature Σ, an infinite set of names $\mathcal{N}$, and an infinite set of variables, the set of *terms* is defined by the grammar:

$L, M, N, T, U, V ::=$	terms
$k, \ldots, n, \ldots, s$	name
x, y, z	variable
$f(M_1, \ldots, M_l)$	function application

where f ranges over the function symbols of Σ and l matches the arity of f. Although names, variables, and constant symbols have similarities, we find it clearer to keep them separate. A term is closed when it does not have free variables (but it may contain names and constant symbols). We write $fn(M)$ for the set of names that occur in the term M. We use meta-variables u, v, w to range over names and variables. The *size* $|T|$ of a term T is defined by $|u| = 1$ and $|f(T_1, \ldots, T_l)| = 1 + \sum_{i=1}^{l} |T_i|$. The *DAG-size* $|T|_{\mathsf{DAG}}$ is the number of distinct subterms of T.

We equip the signature Σ with an equational theory E, that is, an equivalence relation on terms that is closed under substitutions of terms for variables and closed under application

of contexts. We write $M =_E N$ when the equation $M = N$ is in E. We use the symbol $==$ to denote syntactic equality of terms.

2.2 Assembling terms into frames

After a protocol execution, an attacker may know a sequence of messages $M_1, \ldots, M_l$. This means that it knows each message but it also knows in which order it received the messages. So it is not enough for us to say that the attacker knows the set of terms $\{M_1, \ldots, M_l\}$. Furthermore, we should distinguish those names that the attacker had before the execution from those that were freshly generated and which may remain secret from the attacker; both kinds of names may appear in the terms.

In the applied pi calculus [1], such a sequence of messages is organized into a *frame* $\nu\widetilde{n}\sigma$, where $\widetilde{n}$ is a finite set of names (intuitively, the fresh names), and σ is a substitution of the form:

$$\{{}^{M_1}/_{x_1}, \ldots, {}^{M_l}/_{x_l}\} \quad \text{with} \quad dom(\sigma) \stackrel{\text{def}}{=} \{x_1, \ldots, x_l\}.$$

The variables enable us to refer to each M_i, for example for keeping track of their order of transmission. We always assume that our substitutions are cycle-free. The size of a frame $\phi = \nu\widetilde{n}\{{}^{M_1}/_{x_1}, \ldots, {}^{M_l}/_{x_l}\}$ is $|\phi| \stackrel{\text{def}}{=} \sum_{i=1}^{l} |M_i|$.

2.3 Deduction

Given a frame ϕ that represents the information available to an attacker, we may ask whether a given term M may be deduced from ϕ. This relation is written $\phi \vdash M$ (following Schneider [20]). It is axiomatized by the rules:

$$\frac{}{\nu\widetilde{n}\sigma \vdash M} \text{ if } \exists x \in dom(\sigma) \text{ s.t. } x\sigma = M \qquad \frac{}{\nu\widetilde{n}\sigma \vdash s} \, s \notin \widetilde{n}$$

$$\frac{\phi \vdash M_1 \quad \cdots \quad \phi \vdash M_k}{\phi \vdash f(M_1, \ldots, M_k)} \, f \in \Sigma \qquad \frac{\phi \vdash M \quad M =_E M'}{\phi \vdash M'}$$

Since the deducible messages depend on the underlying equational theory, we write $\vdash_E$ when E is not clear from the context. Intuitively, the deducible messages are the messages of ϕ and its free names, closed by equality in E and closed by application of functions. We have the following characterization of deduction:

Proposition 1. *Let M be a closed term and $\nu\widetilde{n}\sigma$ be a frame. Then $\nu\widetilde{n}\sigma \vdash M$ if and only if there exists a term ζ s.t. $fn(\zeta) \cap \widetilde{n} = \emptyset$ and $\zeta\sigma =_E M$.*

As an example, we consider the equational theory of pairing and symmetric encryption. The signature is $\Sigma_1 = \{\mathsf{pair}, \mathsf{enc}, \mathsf{fst}, \mathsf{snd}, \mathsf{dec}\}$. As usual, we write $\langle x, y\rangle$ instead of $\mathsf{pair}(x, y)$. The theory E_1 is defined by:

$$\mathsf{fst}(\langle x, y\rangle) = x \qquad \mathsf{snd}(\langle x, y\rangle) = y \qquad \mathsf{dec}(\mathsf{enc}(x, y), y) = x.$$

Let $\phi \stackrel{\text{def}}{=} \nu k, s\{\mathsf{enc}(s, k)/x, k/y\}$. Then $\phi \vdash k$ and $\phi \vdash s$. Furthermore, we have $k =_{E_1} y\phi$ and $s =_{E_1} \mathsf{dec}(x, y)\phi$.

2.4 Static equivalence

Deduction does not always suffice for expressing the knowledge of an attacker, as discussed in the introduction. For example, consider $\phi_1 \stackrel{\text{def}}{=} \nu k\{\mathsf{enc}(0,k)/x, k/y\}$ and $\phi_2 \stackrel{\text{def}}{=} \nu k\{\mathsf{enc}(1,k)/x, k/y\}$, where $0, 1 \in \Sigma$ are constant symbols. The attacker can deduce the same set of terms from these two frames since it knows 0 and 1. But it could tell the difference between these two frames by checking whether the decryption of x with y produces 0 or 1.

Definition 1. *We say that two terms M and N are equal in the frame φ, for the equational theory E and write $(M =_E N)\varphi$, if and only if $\varphi = \nu\widetilde{n}.\sigma$, $M\sigma =_E N\sigma$, and $\{\widetilde{n}\} \cap (fn(M) \cup fn(N)) = \emptyset$ for some names $\widetilde{n}$ and substitution σ.*

In our example, we have $(\mathsf{dec}(x_1, x_2) =_{E_1} 0)\phi_1$ but not $(\mathsf{dec}(x_1, x_2) =_{E_1} 0)\phi_2$.

Definition 2. *We say that two closed frames φ and ψ are* statically equivalent*, and write $\varphi \approx_s \psi$, when $dom(\varphi) = dom(\psi)$ and when, for all terms M and N, we have $(M =_E N)\varphi$ if and only if $(M =_E N)\psi$.*

In our example, $\phi_1 \not\approx_s \phi_2$ but $\nu k\{\mathsf{enc}(0,k)/x\} \approx_s \nu k\{\mathsf{enc}(1,k)/x\}$.
We write $\approx_{sE}$ when E is not clear from the context.

3 Comparison of the deduction and the static-equivalence problems

We compare equality, deduction, and static equivalence from the point of view of decidability. There is little hope that deduction or static equivalence would be decidable when equality itself is not. (We note however that, for some artificial, especially designed equational theories, deduction may be decidable while equality is undecidable.) Therefore, we focus on equational theories for which equality is at least decidable.

3.1 $\vdash$ may be undecidable

Unfortunately, the decidability of equality is not sufficient for the decidability of deduction and static equivalence. As evidence, let us consider the decidable equational theory E_2 defined by:

$$\begin{aligned} x \cdot (y \cdot z) &= (x \cdot y) \cdot z \\ [x_1, y_1] \cdot [x_2, y_2] &= [x_1 \cdot x_2, y_1 \cdot y_2] \\ f([x \cdot y, x \cdot y]) &= f([x, x]) \end{aligned}$$

According to these equations, the symbol $\cdot$ is associative and distributes over the symbol $[\,]$, and any term of the form $f(M, M)$ can be collapsed into any term $f(M', M')$ where M' is a prefix of M. This equational theory enables us to encode the Post Correspondence Problem (PCP) into the deduction problem.

Proposition 2. *The deduction problem for E_2 ($\vdash_{E_2}$) is undecidable.*

The PCP is: given a finite number of pairs of words $(u_i, v_i)_{1 \leq i \leq n}$ on the alphabet $A \subset \mathcal{N}$, does there exists a sequence $s_1 \cdots s_k \in \{1..n\}^*$ such that: $u_{s_1} \cdots u_{s_k} = v_{s_1} \cdots v_{s_k}$? We map the PCP input $(u_i, v_i)_{1 \leq i \leq n}$ to the substitution $\sigma = \{[u_i, v_i]/x_i\}$. Then we can verify that there exists a solution to the PCP if and only if there exists a letter $a \in A$ such that $(\nu\Sigma)\sigma \vdash_{E_2} T([a, a])$.

3.2 $\vdash$ reduces to $\approx_s$

Next we show that deduction may be reduced to static equivalence. For this purpose, we add the familiar equation $\mathsf{dec}(\mathsf{enc}(x, y), y) = x$. (We have not studied what happens without this equation, since it is so common in applications.)

Proposition 3. *Let E be an equational theory over some signature Σ. Let $0, 1$ be two constants,* dec *and* enc *be two binary function symbols that are not in Σ.*
We define $\Sigma' \stackrel{def}{=} \Sigma \uplus \{0, 1, \mathsf{enc}, \mathsf{dec}\}$ and $E' \stackrel{def}{=} E \uplus \{\mathsf{dec}(\mathsf{enc}(x, y), y) = x\}$. Let $\phi = \nu\widetilde{n}\{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}\}$ be a frame and M be a term. Then $\phi \vdash_E M$ if and only if

$$\nu\widetilde{n}\{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}, ^{\mathsf{enc}(0,M)}/_{x_{l+1}}\} \not\approx_{sE'} \nu\widetilde{n}\{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}, ^{\mathsf{enc}(1,M)}/_{x_{l+1}}\}.$$

We derive that if $\approx_s$ is decidable for $E \cup \{\mathsf{dec}(\mathsf{enc}(x, y), y) = x\}$, then $\vdash$ is decidable for E (with at most the same complexity).

3.3 $\approx_s$ does not reduce to $\vdash$ in general

The converse is not true: $\vdash$ may be decidable while $\approx_s$ is not. Indeed, we can encode an undecidable problem into the static equivalence problem in such a way that the deduction problem remains decidable.

Proposition 4. *There exists an equational theory such that $\approx_s$ is undecidable while $\vdash$ is decidable.*

We consider the following construction: Given two deterministic Turing machines $M_1 = (Q, A, q_0, Q_f, \delta_1)$ and $M_2 = (Q, A, q_0, Q_f, \delta_2)$ with the same control states, where $\delta_1, \delta_2 : Q \times A \to Q \times A \times \{L, R\}$, we construct the machine $\mathcal{M}(M_1, M_2) = (Q, A, q_0, Q_f, \delta)$ where $\delta : \{1, 2\} \times Q \times A \to Q \times A \times \{L, R\}$ such that $\delta(1, q, a) = \delta_1(q, a)$ and $\delta(2, q, a) = \delta_2(q, a)$. At each step, the machine $\mathcal{M}(M_1, M_2)$ plays a transition of either M_1 or M_2. Since the machines M_1 and M_2 are deterministic, a run of the machine $\mathcal{M}(M_1, M_2)$ on a word w may be described by a word s of $\{1, 2\}^*$, which gives the list of choices made by $\mathcal{M}(M_1, M_2)$ at each step. $\mathcal{M}(M_1, M_2), w \stackrel{s}{\to}$ denotes the machine (with its current tape) after the sequence of choices s on the word w. We assume that the local control state is written on the tape.

Proposition 5. *The following problem is undecidable.*
Input: *Two machines* $\mathcal{M}(M_1, M_2)$ *and* $\mathcal{M}(M_1', M_2')$ *and a word* w *of* A^*.
Output: *Does the following property hold for* $\mathcal{M}(M_1, M_2)$ *and* $\mathcal{M}(M_1', M_2')$*: for any sequences* $s_1, s_2 \in \{1,2\}^*$, $\mathcal{M}(M_1, M_2), w \xrightarrow{s_1}$ *and* $\mathcal{M}(M_1, M_2), w \xrightarrow{s_2}$ *have the same tape if and only if* $\mathcal{M}(M_1', M_2'), w \xrightarrow{s_1}$ *and* $\mathcal{M}(M_1', M_2'), w \xrightarrow{s_2}$ *have the same tape?*

We reduce this undecidable problem to the $\approx_s$ problem under an equational theory E_3 such that $\vdash$ remains decidable. The intuitive idea of our encoding is that a frame ϕ represents a machine of the form $\mathcal{M}(M_1, M_2)$, a term M represents a sequence of choices such that $M\phi$ represents the tape of the machine (and the number of choices) after this sequence of choices. Then, for two "machines" ϕ and ϕ', it is undecidable whether there exists two sequences of choices M_1, M_2 such that $(M_1 =_{E_3} M_2)\phi$ and $(M_1 \neq_{E_3} M_2)\phi'$, *i.e.*, whether $\phi \approx_s \phi'$.

On the other hand, it is possible to decide whether there exists a sequence of choices M such that $M\phi =_{E_3} N$ (*i.e.*, whether $\phi \vdash N$) for a given term N. Indeed, the term N contains the number of choices, so it is sufficient to test any sequence of choices of length equal to this number of choices.

4 Deciding knowledge under convergent subterm theories

In order to obtain decidability results for both $\vdash$ and $\approx_s$, we restrict attention to *subterm theories*, defined by a finite set of equations of the form $M = N$ where N is a proper subterm of M or a constant symbol. In section 4.1, we motivate and introduce a convergence condition on subterm theories. Convergent subterm theories are quite common in applications, as we illustrate with examples in section 4.2. We present our main decidability results for these theories in section 4.3.

4.1 Convergence

The definition of subterm theories is almost vacuous on its own. Even equality may be undecidable for subterm theories. Any equational theory defined by a finite set of equations with variables $M = M'$ can be encoded as a subterm theory, with the two equations:

$$\mathsf{Whichever}(M, M') = M \qquad \mathsf{Whichever}(M, M') = M'$$

for each original equation $M = M'$. In light of this encoding, we should add the assumption that, by orienting the equations that define a subterm theory from left to right, we obtain a convergent rewriting system:

Definition 3. *A subterm equational theory* E*, defined by a finite set of equations* $\bigcup_{i=1}^{n}\{M_i = N_i\}$*, is a* convergent subterm theory *if the set of rewriting rules* $r(E) \stackrel{def}{=} \bigcup_{i=1}^{n}\{M_i \to N_i\}$ *is convergent. We write* $u \to_E v$ *if* u *may be rewritten to* v *using a rule of* $r(E)$.

We use $\to$ instead of $\to_E$ when the equational theory is clear from the context.

4.2 Examples

Important destructor-constructor rules like those for pairing, encryption, and signature may be expressed in subterm theories (typically convergent ones):

$$\begin{array}{ll} \mathsf{fst}(< x, y >) = x & \mathsf{dec}(\mathsf{enc}(x, y), y) = x \\ \mathsf{snd}(< x, y >) = y & \mathsf{check}(x, \mathsf{sign}(x, \mathsf{sk}(y)), \mathsf{pk}(y)) = \mathsf{ok} \end{array}$$

Additional examples may be found in previous work (e.g., [1, 7]). Convergent subterm theories also enable us to capture sophisticated but sensible properties, as in:

$$\begin{array}{ll} E_4 = & \{I(I(x)) = x, I(x) \times x = 1, x \times I(x) = 1\} \\ E_5 = & \{h(h(x)) = x\} \\ E_6 = & \{\mathsf{enc}(\mathsf{enc}(x, y), y) = x\}. \end{array}$$

The theory E_4 models an inverse function. The theory E_5 models a hash function that is idempotent on small inputs (since the hash of a hash gives the same hash). The theory E_6 represents an encryption function that also decrypts: the encryption of a plaintext, twice with the same key, returns the plaintext.

4.3 Decidability results

For convergent subterm theories, $\vdash$ and $\approx_s$ become decidable. Let E be a fixed convergent subterm theory, defined by $\bigcup_{i=1}^{n}\{M_i = N_i\}$ Let $c_E = \max_{1 \leq i \leq m}(|C_i|)$.

Theorem 1. *For any frames ϕ and ϕ', for any term M, we can decide $\phi \vdash M$ and $\phi \approx_s \phi'$ in polynomial time in $|\phi|$, $|\phi'|$, and $|M|$.*

The end of this section is devoted to the proof of the theorem; the appendix contains additional detail.

Step 1 of the proof: saturating a frame ϕ. We first associate with each frame ϕ the set of subterms of messages in ϕ that may be deduced from ϕ by applying only small contexts. We prove that this set can be computed in polynomial time. In addition, we show that each term in this set has a "representation" whose DAG-size is polynomial.

Definition 4. *Let $\phi = \nu\widetilde{n}\{M_1/x_1, \cdots, M_k/x_k\}$ be a frame. Let $\mathsf{st}(\phi)$ be the set of subterms of the M_i's. The saturation $\mathsf{sat}(\phi)$ of ϕ is the minimal set such that:*

1. *for every $1 \leq i \leq k$, $M_i \in \mathsf{sat}(\phi)$,*
2. *if $M_1, \ldots, M_k \in \mathsf{sat}(\phi)$ and $C[M_1, \ldots, M_k] \rightarrow M$, where $|C| \leq c_E$ and $M \in \mathsf{st}(\phi)$ then $M \in \mathsf{sat}(\phi)$,*
3. *if $M_1, \ldots, M_k \in \mathsf{sat}(\phi)$ and $f(M_1, \ldots, M_k) \in \mathsf{st}(\phi)$, then $f(M_1, \ldots, M_k) \in \mathsf{sat}(\phi)$.*

Proposition 6. *Let ϕ be a frame, $\phi = \nu\widetilde{n}\sigma$.*

1. *The set $\mathsf{sat}(\phi)$ can be computed in time $\mathcal{O}(|\phi|^{\max(\mathsf{ar}(\Sigma), c_E)+1})$.*

2. *For every* $M \in \mathsf{sat}(\phi)$, *there exists a term* ζ_M *such that* $fn(\zeta_M) \cap \widetilde{n} = \emptyset$, $|\zeta_M|_{\mathsf{DAG}} \leq (c_E+1)|\phi|^2$ *and* $\zeta_M\sigma =_E M$. *The term* ζ_M *is called a* recipe *of* M *and is chosen arbitrarily between the possible terms verifying the above properties.*

The set $\mathsf{sat}(\phi)$ is obtained by saturating the set $\{M_1, \ldots, M_k\}$ by applying the rules 2 and 3 of definition 4. Since $\mathsf{sat}(\phi) \subset \mathsf{st}(\phi)$, this set is saturated in at most $|\phi|$ steps. At each step, we have to compute:

- Every term of the form $C[M_1, \ldots, M_k]$, where $|C| \leq c_E$ and the M_i's are already in the set, and check if it is an instance of some $C_l[M_l]$. Thus we need at most $\mathcal{O}(c_E|\phi|^{c_E})$ computations.
- Every term $f(M_1, \ldots, M_k)$ that is also in $\mathsf{st}(\phi)$. Thus we have to construct at most $|\Sigma||\phi|^{\mathsf{ar}(\Sigma)}$ terms.

Since each step requires at most $\mathcal{O}(|\phi|^{\max(\mathsf{ar}(\Sigma),c_E)})$ computations and since there are at most $|\phi|$ steps, $\mathsf{sat}(\phi)$ may be computed in time $\mathcal{O}(|\phi|^{\max(\mathsf{ar}(\Sigma),c_E)+1})$. For the second part of proposition 6, we already know by proposition 1 that each term M of $\mathsf{sat}(\phi)$ has a recipe ζ_M such that $fn(\zeta) \cap \widetilde{n} = \emptyset$ and $\zeta_M\sigma =_E M$. By construction of $\mathsf{sat}(\phi)$, these recipes may be chosen so that:

1. $\zeta_M = x_i$ if $\sigma(x_i) = M$,
2. $\zeta_M = C[\zeta_{M_1}, \ldots, \zeta_{M_k}]$ with $M_i \in \mathsf{sat}(\phi)$ if M is obtained by the rule 2,
3. $\zeta_M = f(\zeta_{M_1}, \ldots, \zeta_{M_k})$ with $M_i \in \mathsf{sat}(\phi)$ if M is obtained by the rule 3.

Since there are at most $|\mathsf{sat}(\phi)| \leq |\phi|$ recipes, the maximal DAG-size of a recipe of a term in $\mathsf{sat}(\phi)$ is $(c_E+1)|\phi|^2$.

Step 2 of the proof: Introducing a finite set of equalities to characterize a frame. With each frame ϕ, we associate a finite number of equalities $\mathsf{Eq}(\phi)$ such that two frames are equivalent if and only if they satisfy the equalities from each other's set: ϕ' satisfies the equalities $\mathsf{Eq}(\phi)$ and ϕ satisfies the equalities $\mathsf{Eq}(\phi')$. We assume fixed the set of recipes corresponding to the terms of $\mathsf{sat}(\phi)$.

Definition 5. *Let* $\phi = \nu\widetilde{n}\sigma$ *be a frame. The set* $\mathsf{Eq}(\phi)$ *is the set of equalities*

$$C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] = C_2[\zeta_{M'_1}, \ldots, \zeta_{M'_l}]$$

such that $(C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] =_E C_2[\zeta_{M'_1}, \ldots, \zeta_{M'_l}])\phi$, $|C_1|, |C_2| \leq c_E$, *and the* M_i *and* M'_i *are in* $\mathsf{sat}(\phi)$. *If* ϕ' *is a frame such that* $(M =_E N)\phi'$ *for every* $(M = N) \in \mathsf{Eq}(\phi)$, *we write* $\phi' \models \mathsf{Eq}(\phi)$.

Two crucial lemmas show that it is sufficient to consider these equalities:

Lemma 1. *Let* $\phi = \nu\widetilde{n}\sigma$ *and* $\phi' = \nu\widetilde{n'}\sigma'$ *be two frames such that* $\phi' \models \mathsf{Eq}(\phi)$. *For every context* $C_1[x_1, \ldots, x_k]$, $C_2[x_1, \ldots, x_l]$ *such that* $(fn(C_1) \cup fn(C_2)) \cap (\widetilde{n} \cup \widetilde{n'}) = \emptyset$, *for every* $M_i, M'_i \in \mathsf{sat}(\phi)$, *if* $C_1[M_1, \ldots, M_k] == C_2[M'_1, \ldots, M'_l]$ *(syntactic equality), then* $(C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] =_E C_2[\zeta_{M'_1}, \ldots, \zeta_{M'_l}])\phi'$.

Lemma 2. *Let $\phi = \nu\widetilde{n}\sigma$ be a frame. For every context C_1 s.t. $fn(C_1) \cap \widetilde{n} = \emptyset$, for every $M_i \in \mathsf{sat}(\phi)$, for every term T such that $C_1[M_1, \ldots, M_k] \rightarrow_E T$, there exist a context C_2 s.t. $fn(C_2) \cap \widetilde{n} = \emptyset$, and terms $M'_i \in \mathsf{sat}(\phi)$, such that $T == C_2[M'_1, \ldots, M'_l]$ and for every frame $\phi' \models \mathsf{Eq}(\phi)$ such that $\phi' = \nu\widetilde{n'}\sigma'$ and $(fn(C_1) \cup fn(C_2)) \cap \widetilde{n'} = \emptyset$, $(C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] =_E C_2[\zeta_{M'_1}, \ldots, \zeta_{M'_l}])\phi'$.*

How these lemmas are used to prove the decidability of deduction and static equivalence is explained in steps 3 and 4 of the proof, respectively.

Step 3 of the proof: decidability of $\vdash$. Here we show that any message deducible from a frame ϕ is actually a context over terms in $\mathsf{sat}(\phi)$.

Proposition 7. *Let $\phi = \nu\widetilde{n}\sigma$ be a frame, M be a term and $M\downarrow$ its normal form. $\phi \vdash M$ if and only if there exist $C[x_1, \ldots, x_k]$, $M_1, \ldots, M_k \in \mathsf{sat}(\phi)$ such that $fn(C) \cap \widetilde{n} = \emptyset$ and $M\downarrow == C[M_1, \ldots, M_k]$.*

Proof. If $M\downarrow == C[M_1, \ldots, M_k]$ with $fn(C) \cap \widetilde{n} = \emptyset$, then $M =_E C[\zeta_{M_1}, \ldots, \zeta_{M_k}]\sigma$, by construction of the ζ_{M_i}'s. Thus, by proposition 1, $\phi \vdash M$. Conversely, if $\phi \vdash M$, then by proposition 1, there exists ζ such that $fn(\zeta) \cap \widetilde{n} = \emptyset$ and $M =_E \zeta\sigma$. Thus $M\downarrow == (\zeta\sigma)\downarrow$. Applying recursively lemma 2, we obtain that $(\zeta\sigma)\downarrow == C[M_1, \ldots, M_k]$ for some $M_1, \ldots, M_k \in \mathsf{sat}(\phi)$ and C such that $fn(C) \cap \widetilde{n} = \emptyset$.

We derive that $\phi \vdash M$ can be decided by checking whether M is of the form $C[M_1, \ldots, M_k]$ with $M_i \in \mathsf{sat}(\phi)$. As for complexity, given a term M, $M\downarrow$ is computed in time $\mathcal{O}(|M|)$. Once $\mathsf{sat}(\phi)$ is computed, checking if there exist $C[x_1, \ldots, x_k], M_1, \ldots, M_k \in \mathsf{sat}(\phi)$ such that $fn(C) \cap \widetilde{n} = \emptyset$ and $M\downarrow == C[M_1, \ldots, M_k]$ may be done in time $\mathcal{O}(|M||\phi|)$. We conclude that $\phi \vdash M$ is decidable in time $\mathcal{O}(|M||\phi|^{\max(\mathsf{ar}(\Sigma), c_E)+1})$.

Step 4 of the proof: decidability of $\approx_s$.

Proposition 8. *Let ϕ and ϕ' be two frames.*

$$\phi \approx_s \phi' \quad \text{if and only if} \quad \phi \models \mathsf{Eq}(\phi') \text{ and } \phi' \models \mathsf{Eq}(\phi).$$

Proof. By definition of static equivalence, if $\phi \approx_s \phi'$ then $\phi \models \mathsf{Eq}(\phi')$ and $\phi' \models \mathsf{Eq}(\phi)$. Assume now that $\phi' \models \mathsf{Eq}(\phi)$ and consider M, N such that there exist $\widetilde{n}, \sigma$ such that $\phi = \nu\widetilde{n}\sigma$, $(fn(M) \cup fn(N)) \cap \widetilde{n} = \emptyset$ and $(M =_E N)\phi$. Then $M\sigma =_E N\sigma$, so $(M\sigma)\downarrow == (N\sigma)\downarrow$. Let $T \stackrel{\text{def}}{=} (M\sigma)\downarrow$. Applying recursively lemma 2, we obtain that there exist $M_1, \ldots, M_k \in \mathsf{sat}(\phi)$ and C_M such that $fn(C_M) \cap \widetilde{n} = \emptyset$ and

$$T == C_M[M_1, \ldots, M_k] \text{ and } M\sigma' =_E C_M[\zeta_{M_1}, \ldots, \zeta_{M_k}]\sigma'.$$

Since $T == (N\sigma)\downarrow$, we obtain similarly that there exist $M'_1, \ldots, M'_l \in \mathsf{sat}(\phi)$ and C_N such that $fn(C_N) \cap \widetilde{n} = \emptyset$ and

$$T == C_N[M'_1, \ldots, M'_l] \text{ and } N\sigma' =_E C_N[\zeta_{M'_1}, \ldots, \zeta_{M'_l}]\sigma'.$$

Moreover, since $C_M[M_1, \ldots, M_k] == C_N[M'_1, \ldots, M'_l]$, we derive from lemma 1 that $C_M[\zeta_{M_1}, \ldots, \zeta_{M_k}]\sigma' =_E C_N[\zeta_{M'_1}, \ldots, \zeta_{M'_l}]\sigma'$, thus $(M =_E N)\phi'$. Conversely, if $(M =_E N)\phi'$ and $\phi \models \mathsf{Eq}(\phi')$, we can prove that $(M =_E N)\phi$. We conclude $\phi \approx_s \phi'$.

Therefore, given ϕ and ϕ', to decide whether $\phi \approx_s \phi'$ we have to construct $\mathsf{sat}(\phi)$ and $\mathsf{sat}(\phi')$. This can be done in time $\mathcal{O}(\max(|\phi|, |\phi'|)^{\max(\mathsf{ar}(\Sigma), c_E)+1})$ by proposition 6. For each term M of $\mathsf{sat}(\phi)$ or $\mathsf{sat}(\phi')$, the term ζ_M has a polynomial DAG-size. Then, for any contexts C_1, C_2 such that $|C_1|, |C_2| \leq c_E$, for any $M_i, M'_i \in \mathsf{sat}(\phi)$, we check whether $(C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] =_E C_2[\zeta_{M'_1}, \ldots, \zeta_{M'_l}])\phi$ and $(C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] =_E C_2[\zeta_{M'_1}, \ldots, \zeta_{M'_l}])\phi'$. There are at most $(c_E|\phi|^{c_E})^2$ equalities in $\mathsf{Eq}(\phi)$. Each term of the form $C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}]\phi$ has a polynomial DAG-size. The equality of two terms represented by DAGs can be checked in polynomial time: we do not need to expand the DAGs to test for equality. We conclude that $\phi \approx_s \phi'$ can be decided in polynomial time in $|\phi|$ and $|\phi'|$.

5 Conclusion

This paper investigates decidability questions for message deducibility and static equivalence, two formal representations for knowledge in the analysis of security protocols. This investigation yields a few somewhat negative results, for example that static equivalence cannot always be reduced to message deducibility. On the other hand, the main results are strong, positive ones: both message deducibility and static equivalence are decidable in polynomial time under a large and useful class of equational theories.

These positive results suggest some directions for further research in protocol analysis. In the general case of infinite-state protocols, our algorithms could be integrated into analysis tools; substantial work on optimizations may however be required. For finite-state protocols, various security properties are decidable under specific equational theories (e.g., [4]). Perhaps our results can serve as the starting point for a generalization to a broad class of equational theories. This generalization may be easy if one restricts attention to passive attackers (eavesdroppers): since the capabilities of eavesdroppers are limited to deducing and comparing messages, our decidability results may apply fairly directly. The case with active attackers is clearly more difficult and interesting; as mentioned in the introduction, Delaune and Jacquemard proved that the deduction problem remains decidable for a subclass of convergent subterm theories. It remains to study how our work could be extended to equivalences.

Acknowledgments

We are grateful to Michael Rusinowitch for helpful discussions.

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, January 2001.

2. M. Abadi and A. D. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4):267–303, Winter 1998.
3. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, Jan. 1999.
4. R. M. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In C. Palamidessi, editor, *CONCUR 2000: Concurrency Theory (11th Int. Conference)*, volume 1877 of *LNCS*, pages 380–394. Springer Verlag, Aug. 2000.
5. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96. IEEE Computer Society, June 2001.
6. B. Blanchet. From secrecy to authenticity in security protocols. In M. Hermenegildo and G. Puebla, editors, *9th Int. Static Analysis Symposium (SAS'02)*, volume 2477 of *LNCS*, pages 342–359. Springer Verlag, Sept. 2002.
7. B. Blanchet. Automatic proof of strong secrecy for security protocols. In *IEEE Symposium on Security and Privacy*, page To appear, May 2004.
8. M. Boreale, R. De Nicola, and R. Pugliese. Proof techniques for cryptographic processes. In *Proceedings of the Fourteenth Annual IEEE Symposium on Logic in Computer Science*, pages 157–166, July 1999.
9. Y. Chevalier, R. Kuester, M. Rusinowitch, and M. Turani. Deciding the security of protocols with Diffie-Hellman exponentiation and products in exponents. In *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference*, volume 2914 of *LNCS*, pages 124–135. Springer Verlag, 2003.
10. Y. Chevalier, R. Kuester, M. Rusinowitch, and M. Turani. An NP decision procedure for protocol insecurity with xor. In *Proceedings of the 18th Annual IEEE Symposium on Logic In Computer Science (LICS'03)*, pages 261–270, 2003.
11. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proceedings of the 18th Annual IEEE Symposium on Logic In Computer Science (LICS'03)*, pages 271–280, 2003.
12. H. Comon-Lundh and R. Treinen. Easy intruder deductions. In *Research Report LSV-03-8*, 2003.
13. S. Delaune and F. Jacquemard. Narrowing-based constraint solving for the verification of security protocols. In *Research Report LSV-04-8*, April 2004. 24 pages.
14. D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(12):198–208, Mar. 1983.
15. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, Apr. 1984.
16. R. Kemmerer, C. Meadows, and J. Millen. Three system for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, Spring 1994.
17. P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proceedings of the Fifth ACM Conference on Computer and Communications Security*, pages 112–121, 1998.
18. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055 of *LNCS*, pages 147–166. Springer Verlag, 1996.
19. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1–2):85–128, 1998.

20. S. Schneider. Security properties and CSP. In *IEEE Symposium on Security and Privacy*, pages 174–187, 1996.

Appendix

A Impossibility of reducing $\approx_s$ to $\vdash$ in general

Here we give the missing proofs of section 3.3.

Proposition 9 (proposition 5). *The following problem is undecidable.*
Input: *Two machines $\mathcal{M}(M_1, M_2)$ and $\mathcal{M}(M_1', M_2')$ and a word w of A^*.*
Output: *Does the following property* **(P)** *hold for $\mathcal{M}(M_1, M_2)$ and $\mathcal{M}(M_1', M_2')$: for any sequences $s_1, s_2 \in \{1,2\}^*$, $\mathcal{M}(M_1, M_2), w \xrightarrow{s_1}$ and $\mathcal{M}(M_1, M_2), w \xrightarrow{s_2}$ have the same tape if and only if $\mathcal{M}(M_1', M_2'), w \xrightarrow{s_1}$ and $\mathcal{M}(M_1', M_2'), w \xrightarrow{s_2}$ have the same tape?*

Proof. The halting problem of a deterministic Turing machine can be reduced to this problem. Given any deterministic Turing machine $M = (Q, A, q_0, Q_f, \delta)$, we construct the deterministic Turing machine $\mathcal{T}(M) = (Q, A \uplus \{c_o\}, q_0, Q_f, \delta')$, where we modify the transitions for the final states:

$$\begin{cases} \delta'(q,a) = \delta(q,a) & \forall a \in A, q \notin Q_f \\ \delta'(q,a) = (q, c_0, L) & \forall a \in A, q \in Q_f. \end{cases}$$

Then $\mathcal{M}(M, \mathcal{T}(M)), w \xrightarrow{s_1}$ and $\mathcal{M}(M, \mathcal{T}(M)), w \xrightarrow{s_2}$ have the same tape for any sequences $s_1, s_2 \in \{1,2\}^*$ if and only if M does not reach its final state on w.

Now, let M_0 be any fixed deterministic Turing machine. For any sequences $s_1, s_2 \in \{1,2\}^*$, $\mathcal{M}(M_0, M_0), w \xrightarrow{s_1}$ and $\mathcal{M}(M_0, M_0), w \xrightarrow{s_2}$ have the same tape. We deduce that M reaches its final state on w if and only $\mathcal{M}(M, \mathcal{T}(M))$ and $\mathcal{M}(M_0, M_0)$ satisfy the property (P).

To encode this undecidability result into $\approx_s$, we consider the equational theory E_3 displayed in figure 1. By orienting the equations from left to right, we obtain convergent rewriting rules such that $M =_{E_3} M'$ if and only if $M \downarrow == M' \downarrow$ where $M \downarrow$ is the normal form of M for these rewriting rules. Intuitively, we consider terms of the form $h(w_1, q, w_2, s^n(0))$, where w_1 represents the tape before the machine's head, w_2 represents the tape after the machine's head, q is the control state and $s^n(0)$ is a counter representing the number of rules which have been applied. A term $[(q, a \to q_1, a_1, D_1), (q, a \to q_2, a_2, D_2)]$ represents a couple of rules of two Turing machine. Then the term

$$\mathsf{Apply}(i, [(q, a \to q_1, a_1, D_1), (q, a \to q_2, a_2, D_2)], h(w_1, q, w_2, s^n(0))),$$

where $i \in \{1,2\}, D_1, D_2 \in \{L, R\}$, represents the application of the rule number 1 or 2 (depending on i) on the tape $h(w_1, q, w_2, s^n(0))$. The result of this application is given by the equational theory E_3.

Now, to each machine $\mathcal{M}(M_1, M_2)$, we associate the frame $\phi_{\mathcal{M}(M_1,M_2)}$:

$$\nu(A \cup Q)[h(\#, q_0, \#, 0)/x_0] \cup \bigcup_{a \in A, q \in Q} [[(q, a \to \delta_1(q,a)), (q, a \to \delta_2(q,a))]/x_{a,q}] \,.$$

$$\begin{aligned}
\mathsf{Apply}(1,[(x_q,x_1\to x_{q'},x_2,R),y],h(z_1,x_q,x_1\cdot z_2,x')) &= h(z_1\cdot x_2,x_{q'},z_2,s(x'))\\
\mathsf{Apply}(1,[(x_q,x_1\to x_{q'},x_2,R),y],h(z_1,x_q,x_1,x')) &= h(z_1\cdot x_2,x_{q'},\#,s(x'))\\
\mathsf{Apply}(1,[(x_q,x_1\to x_{q'},x_2,L),y],h(z_1\cdot x_3,x_q,x_1\cdot z_2,x')) &= h(z_1,x_{q'},x_3\cdot(x_2\cdot z_2),s(x'))\\
\mathsf{Apply}(2,[y,(x_q,x_1\to x_{q'},x_2,R)],h(z_1,x_q,x_1\cdot z_2,x')) &= h(z_1\cdot x_2,x_{q'},z_2,s(x'))\\
\mathsf{Apply}(2,[y,(x_q,x_1\to x_{q'},x_2,R)],h(z_1,x_q,x_1,x')) &= h(z_1\cdot x_2,x_{q'},\#,s(x'))\\
\mathsf{Apply}(2,[y,(x_q,x_1\to x_{q'},x_2,L)],h(z_1\cdot x_3,x_q,x_1\cdot z_2,x')) &= h(z_1,x_{q'},x_3\cdot(x_2\cdot z_2),s(x'))
\end{aligned}$$

Fig. 1. The equational theory E_3.

Then we can verify that two machines $\mathcal{M}(M_1,M_2)$ and $\mathcal{M}(M_1',M_2')$ verify the property (P) of proposition 5 if and only if $\phi_{\mathcal{M}(M_1,M_2)}\approx_s\phi_{\mathcal{M}(M_1',M_2')}$. We deduce that $\approx_s$ is undecidable for the equational theory E_3. In the same time, $\vdash$ remains decidable. Indeed, to decide whether $\phi\vdash M$, where $\phi=\nu\widetilde{n}\sigma$ it is sufficient to decide if there exists ζ such that $fn(\zeta)\cap\widetilde{n}=\emptyset$ and $\zeta\sigma=_{E_3}M$, *i.e.*, $\zeta\sigma\downarrow==M\downarrow$. Intuitively, for ϕ of the form $\phi_{\mathcal{M}(M_1,M_2)}$ and for M of the form $h(w_1,q,w_2,s^n(0))$, we are looking for some sequences of choices (represented by ζ) such that the tape of the machine $\mathcal{M}(M_1,M_2)$ after this sequence of choices is equal to M. Since the term M contains the number of rules that have been applied, it is sufficient to test any sequence of choices of length equal to this number of rules, which gives a finite number of sequences to check. This can be generalized to any ϕ and M, proving that $\vdash$ is decidable.

B Decidability results for $\vdash$ and $\approx_s$ for subterm theories

B.1 Proof of lemmas 1 and 2

We introduce the definition of *paths* in a term t.

Definition 6. *The set of* paths *of a term M, denoted by $\mathcal{P}(M)$ is defined recursively by: $\mathcal{P}(u)=\epsilon$ if u is a name or a variable, $\mathcal{P}(f(M_1,\ldots,M_n))=\epsilon\cup\bigcup_{i=1}^n i\cdot\mathcal{P}(M_i)$. The subterm of M at position p, denoted by $M|_p$, is defined by: $M|_\epsilon=M$ and $f(M_1,\ldots,M_n)|_{i\cdot p}=M_i|_p$.*

Lemma 3 (lemma 1). *Let $\phi=\nu\widetilde{n}\sigma$ and $\phi'=\nu\widetilde{n'}\sigma'$ be two frames such that $\phi'\models\mathsf{Eq}(\phi)$. For every context $C_1[x_1,\ldots,x_k]$, $C_2[x_1,\ldots,x_l]$ such that $(fn(C_1)\cup fn(C_2))\cap(\widetilde{n}\cup\widetilde{n'})=\emptyset$, for every $M_i,M_i'\in\mathsf{sat}(\phi)$, if $C_1[M_1,\ldots,M_k]==C_2[M_1',\ldots,M_l']$ (syntactic equality), then*

$$(C_1[\zeta_{M_1},\ldots,\zeta_{M_k}]=_E C_2[\zeta_{M_1'},\ldots,\zeta_{M_l'}])\phi'.$$

This lemma is proved by induction on the sum of the size of C_1 and C_2.
Base case: If $|C_1|,|C_2|\leq c_E(E)$, then the equality

$$(C_1[\zeta_{M_1},\ldots,\zeta_{M_k}]=C_2[\zeta_{M_1'},\ldots,\zeta_{M_l'}])$$

is in $\mathsf{Eq}(\phi)$, so $\phi' \models \mathsf{Eq}(\phi)$ implies $(C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] =_E C_2[\zeta_{M'_1}, \ldots, \zeta_{M'_l}])\phi'$.
Inductive step: If neither C_1 nor C_2 is a variable, then $C_1 == f(C_1^1, \ldots, C_r^1)$, $C_2 == f(C_1^2, \ldots, C_r^2)$ and for every $1 \leq i \leq r$, $C_i^1[M_1, \ldots, M_k] == C_i^2[M'_1, \ldots, M'_l]$. By applying the induction hypothesis, we obtain $(C_i^1[\zeta_{M_1}, \ldots, \zeta_{M_k}] =_E C_i^2[\zeta_{M'_1}, \ldots, \zeta_{M'_l}])\phi'$, thus $(C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] =_E C_2[\zeta_{M'_1}, \ldots, \zeta_{M'_l}])\phi'$.
If C_1 or C_2 is a variable, then let us say $C_1 == f(C_1^1, \ldots, C_r^1)$ and $C_2 == x$. We have $f(C_1^1[M_1, \ldots, M_k], \ldots, C_1^r[M_1, \ldots, M_k]) == M$. For every $1 \leq i \leq r$, let $N_i \stackrel{\text{def}}{=} C_i^1[M_1, \ldots, M_k]$. N_i is a subterm of M, thus is in $\mathsf{st}(\sigma)$. Since each M_j is in $\mathsf{sat}(\phi)$ and by applying recursively rule 3 of definition 4, we get that N_i is in $\mathsf{sat}(\phi)$. Thus $\zeta_{N_i}\sigma =_E N_i$.

- From $N_i == C_i^1[M_1, \ldots, M_k]$ and applying the induction hypothesis, we get $\zeta_{N_i}\sigma' =_E C_i^1[\zeta_{M_1}, \ldots, \zeta_{M_k}]\sigma'$.
- From $M == f(N_1, \ldots, N_r)$, we get $\zeta_M\sigma' =_E f(\zeta_{N_1}, \ldots, \zeta_{N_r})\sigma'$.

Combining these equalities, we get $(\zeta_M =_E C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}])\phi'$.

Lemma 4 (lemma 2). *Let $\phi = \nu\widetilde{n}\sigma$ be a frame. For every context C_1 s.t. $fn(C_1) \cap \widetilde{n} = \emptyset$, for every $M_i \in \mathsf{sat}(\phi)$, for every term T such that $C_1[M_1, \ldots, M_k] \rightarrow_E T$, there exist a context C_2 s.t. $fn(C_2) \cap \widetilde{n} = \emptyset$, and terms $M'_i \in \mathsf{sat}(\phi)$, such that $T == C_2[M'_1, \ldots, M'_l]$ and for every frame $\phi' \models \mathsf{Eq}(\phi)$ such that $\phi' = \nu\widetilde{n'}\sigma'$ and $(fn(C_1) \cup fn(C_2)) \cap \widetilde{n'} = \emptyset$, $(C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] =_E C_2[\zeta_{M'_1}, \ldots, \zeta_{M'_l}])\phi'$.*

An easy case is when the reduction occurs inside one of the M_i: $M_i \rightarrow M'_i$. By construction of $\mathsf{sat}(\phi)$, we know that $M'_i \in \mathsf{sat}(\phi)$. In addition, the equality $(\zeta_{M_i} = \zeta_{M'_i})$ is in $\mathsf{Eq}(\phi)$, thus $(\zeta_{M_i} =_E \zeta_{M'_i})\phi'$. We have $T == C_1[M_1, \ldots, M'_i, \ldots, M_k]$ and $(C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] =_E C_1[\zeta_{M'_1}, \ldots, \zeta_{M'_i}, \ldots, \zeta_{M'_l}])\phi'$.

When the reduction does not occur inside some of the M_i's, we have

$$C_1[M_1, \ldots, M_k] == C_3[C_4[M_1, \ldots, M_k], M_1, \ldots, M_k]$$

and $C_4[M_1, \ldots, M_k]$ is an instance $C_0\theta$ of some context C_0, where $C_0[x_1, \ldots, x_n] \rightarrow C'_0[x_1, \ldots, x_n]$ or $C_0[x_1, \ldots, x_n] \rightarrow a$ is a rewriting rule associated with the theory E (thus $C'_0[x_1, \ldots, x_n]$ is a subterm of $C_0[x_1, \ldots, x_n]$ or a is constant symbol).

We rewrite $C_0\theta$ in $C''_0[C_1^0[M_1, \ldots, M_k], \ldots, C_n^0[M_1, \ldots, M_k], M_1, \ldots, M_k]$ such that C_0 is an instance of $C''_0[x_1, \ldots, x_n, y_1, \ldots, y_l]$ and for every path p in C''_0, leading to a variable x_i, $C_0|_p = x_i$, for every path p in C''_0, leading to a variable y_i, $C_0\theta|_p = M_i \in \mathsf{sat}(\phi)$ (see figure 2): C''_0 is the maximal context such that C_0 and C_4 are instances of C''_0.

We transform again $C_0\theta$ in order to obtain a context of terms of $\mathsf{sat}(\phi)$, such that the context is of size smaller than $c_E(E)$. For each variable x_i of C''_0:

- either this variable is *tested for equality under some y_j*: there exists a path $p = p_1 \cdot p_2$ such that $C''_0|_{p_1} = y_j$ and $C_0|_{p_1 \cdot p_2} = x_i$. In this case, we defined $M''_i \stackrel{\text{def}}{=} x_i\theta$. Then $M''_i == C_i^0[M_1, \ldots, M_k]$ and is a subterm of M_j, thus applying recursively the rule 3 of definition 4, we have $M''_i \in \mathsf{sat}(\phi)$,

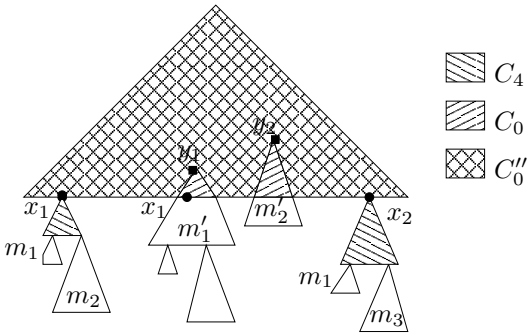


Fig. 2. Transforming $C_0\theta$ into $C''_0[C^0_1[M_1,\ldots,M_k],\ldots,C^0_n[M_1,\ldots,M_k],M'_1,\ldots,M'_l]$.

– either this variable is *unconstrained* in C_0: for every path p in C_0 such that $C_0|_p = x_i$, p is a path in C''_0 and $C''_0|_p = x_i$.

By renaming the variables of C_0 and C''_0, we may assume that $x_1,\ldots,x_r$ are unconstrained in C_0 and $x_{r+1},\ldots,x_n$ are tested for equality under some variables. We obtain that:

$$C_0\theta == C''_0[C^0_1[M_1,\ldots,M_k],\ldots,C^0_r[M_1,\ldots,M_k],M''_{r+1},\ldots,M''_n,M_1,\ldots,M_k],$$

with $M_i, M''_i \in \mathsf{sat}(\phi)$. We have to consider three cases depending on the form of $C'_0\theta$, which is a subterm of $C_0\theta$ or a constant symbol.

– Either $C'_0\theta$ is of the form:

$$C'''_0[C^0_1[M_1,\ldots,M_k],\ldots,C^0_r[M_1,\ldots,M_k],M''_{r+1},\ldots,M''_n,M_1,\ldots,M_k]$$

for some context C'''_0. Since $C_0\theta \to C'_0\theta$ and since the variables $x_1,\ldots,x_r$ are unconstrained, we also have

$$C''_0[a_1,\ldots,a_r,M''_{r+1},\ldots,M''_n,M_1,\ldots,M_k] \to C'''_0[a_1,\ldots,a_r,M''_{r+1},\ldots,M''_n,M_1,\ldots,M_k],$$

where the a_i's a fresh names. Thus

$$(C''_0[a_1,\ldots,a_r,\zeta_{M''_{r+1}},\ldots,\zeta_{M''_n},\zeta_{M_1},\ldots,\zeta_{M_k}] =_E C'''_0[a_1,\ldots,a_r,\zeta_{M''_{r+1}},\ldots,\zeta_{M''_n},\zeta_{M_1},\ldots,\zeta_{M_k})\phi.$$

Since this equality belongs to $\mathsf{Eq}(\phi)$, we deduce:

$$(C''_0[a_1,\ldots,a_r,\zeta_{M''_{r+1}},\ldots,\zeta_{M''_n},\zeta_{M_1},\ldots,\zeta_{M_k}] =_E C'''_0[a_1,\ldots,a_r,\zeta_{M''_{r+1}},\ldots,\zeta_{M''_n},\zeta_{M_1},\ldots,\zeta_{M_k})\phi'.$$

By replacing each a_i by $C_i^0[\zeta_{M_1}, \ldots, \zeta_{M_k}]$, we get:

$$(C_0''[C_1^0[\zeta_{M_1}, \ldots, \zeta_{M_k}], \ldots, C_r^0[\zeta_{M_1}, \ldots, \zeta_{M_k}], \zeta_{M''_{r+1}}, \ldots, \zeta_{M''_n}, \zeta_{M_1}, \ldots, \zeta_{M_k}] =_E$$
$$C_0'''[C_1^0[\zeta_{M_1}, \ldots, \zeta_{M_k}], \ldots, C_r^0[\zeta_{M_1}, \ldots, \zeta_{M_k}], \zeta_{M''_{r+1}}, \ldots, \zeta_{M''_n}, \zeta_{M_1}, \ldots, \zeta_{M_k}])\phi'. \quad (1)$$

In addition, since

$$C_1[M_1, \ldots, M_k] == C_3[C_0''[C_1^0[M_1, \ldots, M_k], \ldots, C_r^0[M_1, \ldots, M_k],$$
$$M''_{r+1}, \ldots, M''_n, M_1, \ldots, M_k], M_1, \ldots, M_k],$$

we deduce, applying lemma 1 and equation 1, that:

$$C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] =_E C_3[C_0'''[C_1^0[\zeta_{M_1}, \ldots, \zeta_{M_k}], \ldots, C_r^0[\zeta_{M_1}, \ldots, \zeta_{M_k}],$$
$$\zeta_{M''_{r+1}}, \ldots, \zeta_{M''_n}, \zeta_{M_1}, \ldots, \zeta_{M_k}, \zeta_{M_1}, \ldots, \zeta_{M_k})\phi',$$

which allows to conclude since

$$T == C_4[C_0'''[C_1^0[M_1, \ldots, M_k], \ldots, C_r^0[M_1, \ldots, M_k],$$
$$M''_{r+1}, \ldots, M''_n, M_1, \ldots, M_k], M_1, \ldots, M_k].$$

– Either $C_0'\theta$ is a subterm M_0 of one of the M_i or M_i''. Since the variables $x_1, \ldots, x_r$ are unconstrained, we also have

$$C_0''[a_1, \ldots, a_r, M''_{r+1}, \ldots, M''_n, M_1, \ldots, M_k] \to M_0,$$

where the a_i's a fresh names. Since $|C_0''| \leq c_E(E)$ and by the rule 2 of definition 4, we have $M_0 \in \mathsf{sat}(\phi)$. Thus

$$(C_0''[a_1, \ldots, a_r, \zeta_{M''_{r+1}}, \ldots, \zeta_{M''_n}, \zeta_{M_1}, \ldots, \zeta_{M_k}] =_E \zeta_{M_0})\phi.$$

Since this equality belongs to $\mathsf{Eq}(\phi)$, we deduce:

$$(C_0''[a_1, \ldots, a_r, \zeta_{M''_{r+1}}, \ldots, \zeta_{M''_n}, \zeta_{M_1}, \ldots, \zeta_{M_k}] =_E \zeta_{M_0})\phi'.$$

We conclude like the previous case.

– Either $C_0'\theta$ is a constant symbol. We conclude like the case above.

B.2 DAG representations

In the core of the article, we claim that given two DAG representations of terms, we can check for equality in polynomial time. We prove this claim in two steps : first we study the complexity of checking for syntactic equality, then we deduce the complexity of checking for equational equality.

Let us formally define what is a DAG representation of a term.

Definition 7 (DAG representation). *A DAG representation of a term is a Direct Acyclic Graph* (V, l, E, v_0)*, where* V *is the set of vertices,* $l : V \to \Sigma$ *a labelling function,* $E \subseteq V \times V \times \{1..\mathsf{ar}(\Sigma)\}$ *the set of edges and* $v_0 \in V$ *the* root *of the graph. In addition, we assume that for every* $v \in V$*, for every integer* i *such that* $0 \leq i \leq \mathsf{arity}(l(v))$*, there exists a unique* v' *(denoted by* $E(v, i)$*) such that* (v, v', i) *is in* E *and that there is not edge of the form* (v, v', i) *for* $i > \mathsf{arity}(l(v))$*.*

The size of R*, denoted by* $|R|$*, is defined by the number of vertices.*

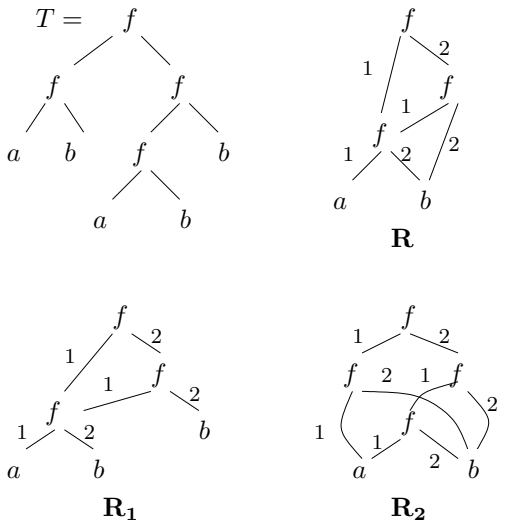
The term represented *by a DAG* (V, l, E, v_0)*, denoted by* $t(V, l, E, v_0)$ *is recursively defined by* $t(V, l, E, v_0) = l(v_0)(t(V, l, E, e(v_0, 1)), \ldots, t(V, l, E, e(v_0, k)))$ *where* $k = \mathsf{arity}(l(v_0))$*.*

A DAG representation (V, l, E, v_0) *is said* minimal *if there is not distinct vertices* v_1 *and* v_2 *such that* $t(V, l, E, v_1) = t(V, l, E, v_2)$

Note that the real memory size needed to represent a DAG R is larger than our definition of size $|R|$ but is polynomial (actually quadratic) in $|R|$. Thus this definition will be sufficient to prove our result.

It is easy to verify that, to each term T, we can associate a unique minimal DAG-representation of T such that its number of vertices is equal to the number of subterms in T (*i.e.*, $|T|_{\mathsf{DAG}}$).

For examples, see figure 3.



The DAG $\mathbf{R}$ is the minimal representation of T but $\mathbf{R_1}$ and $\mathbf{R_2}$ are also DAG representations of T.

Fig. 3. Examples of DAG representations.

Proposition 10. *Given a DAG representation R, we can compute the minimal DAG representation of $t(R)$ in polynomial time in $|R|$. We deduce that checking whether $t(R_1) == t(R_2)$ where R_1, R_2 are two DAG-representations can be done in polynomial time in $|R_1|$ and $|R_2|$.*

Proof. Given a DAG representation R, we check if there exist two distinct vertices v_1 and v_2 (at most $|R|^2$ possibilities) such that $l(v_1) = l(v_2)$ and for every i such that $0 \leq i \leq \mathsf{arity}(l(v_1))$, $E(v_1, i) = E(v_2, i)$. If such v_1, v_2 exist, we suppress v_1 in the set of vertices and replace each occurrence of v_1 in E by v_2. This can be done in time $\mathcal{O}(|R|)$. Thus this step is done is $\mathcal{O}(|R|^2)$.

We repeat this procedure as longer as possible (at most $|R|$ time). We end with the minimal representation of $t(R)$. The total cost of this procedure is at most $\mathcal{O}(|R|^3)$.

Proposition 11. *Given a DAG representation R, we can compute a DAG representation of $R\downarrow$ (where $R\downarrow$ is the normal form of R for some subterm equational theory) in polynomial time in $|R|$. We deduce that checking whether $t(R_1) =_E t(R_2)$ where R_1, R_2 are two DAG-representations can be done in polynomial time in $|R_1|$ and $|R_2|$.*

Proof. (sketch) Let $R = (V, l, E, v_0)$ be a DAG representation. For every rewriting rule of the form $C[x_1, \ldots, x_n] \to C'[x_1, \ldots, x_n]$ or $C[x_1, \ldots, x_n] \to a$ associated with E, we check (from the root) if the pattern C appears in R (at most $|C||R|$ tests). If it is the case (*i.e.*, there exists some $v \in V$ such that $t(V, l, E, v) == C[x_1, \ldots, x_n]\theta$ for some θ), then we associate with every maximal path p in C a vertices v_p obtained from v following the path p. For every p, p', maximal paths in C such that $C|_p == C|_{p'}$, we check whether $t(v_p) == t(v_{p'})$ (syntactic equality). There are at most $|C|$ vertices, thus this can be done in time $\mathcal{O}(|C|^2|R|^3)$ using proposition 10. Then we replace the vertices v by one of the vertices representing $C'[x_1, \ldots, x_n]\theta$ or we add the a vertices representing a.

At each step (except for a constant number of cases), one of the vertices is suppressed, thus this procedure stops after at most $|R|$ steps. We end with a DAG-representation of $R\downarrow$, which is computed in time $\mathcal{O}(|R|^4)$.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399