Deducibility constraints *

Sergiu Bursuc[1], Hubert Comon-Lundh[1,2], and Stéphanie Delaune[1]

[1] LSV, CNRS & ENS Cachan & INRIA project SECSI
[2] AIST, Tokyo

Abstract. In their work on tractable deduction systems, D. McAllester and later D. Basin and H. Ganzinger have identified a property of inference systems (the locality property) that ensures the tractability of the Entscheidungsproblem.
On the other hand, deducibility constraints are sequences of deduction problems in which some parts (formulas) are unknown. The problem is to decide their satisfiability and to represent the set of all possible solutions. Such constraints have also been used for deciding some security properties of cryptographic protocols.
In this paper we show that local inference systems (actually a slight modification of such systems) yield not only a tractable deduction problem, but also decidable deducibility constraints. Our algorithm not only allows to decide the existence of a solution, but also gives a representation of all solutions.

1 Introduction

Deciding whether a given statement can be derived from hypotheses, using a set of formal inference rules, is one of the famous issues in proof theory, known as the *Entscheidungsproblem*. This is undecidable for first-order logic and untractable for propositional logic. There are however several formal proof systems for which the problem is tractable, for instance Horn propositional logic, but also the so-called Dolev-Yao intruder deduction rules. D. McAllester [13] observed that *any* inference system which is *local* yields a tractable Entscheidungsproblem. D. Basin and H. Ganzinger [1] proved that locality is equivalent to a saturation property of the set of inference rules.

The Dolev-Yao inference system is local (and saturated with respect to the subterm ordering). This is why deciding whether a message can be computed by an attacker from a finite set of messages can be performed in polynomial (actually linear) time in this formal proof system. Now, if we consider an active attacker, we need not only to solve the Entscheidungsproblem, but a more general problem in which some statements and proofs are unknown: this corresponds to the attacker's choices. This was formalized in [14], using *deducibility constraints*. There are many historical examples of deducibility constraints in mathematics: Fermat gave a proof of his famous theorem, in which parts were missing; filling the holes amounts to solve a deducibility constraint (in formal arithmetic).

* This work has been partially supported by the ANR-07-SESU-002 AVOTÉ

The starting point of the present work is the problem of lifting the results of D. McAllester, D. Basin and H. Ganzinger from deducibility to deducibility constraints. For the security protocols, this corresponds to moving from a passive adversary to an active adversary.

We consider formal inference systems without AC-symbols (as in [1]). We also assume that there is a single unary predicate symbol (this corresponds to the attacker knowledge in security protocols). Then we prove that, for any inference system that is saturated in some suitable way (we call it *good*), the deducibility constraints are decidable. Actually, we prove more: we provide with a constraint simplification algorithm that yields *solved forms*. This allows not only to decide the existence of a solution, but also to represent all solutions. Such a feature is used in [7] for deciding trace properties such as authentication and key cycles in security protocols, and also in [11] for deciding game-theoretic security properties such as abuse-freeness. Our results generalize [7] to any good inference system. Finally, we claim that our transformation rules are simple: we simply guess the last inference step and reflect this on the constraint solving. The difficult part is then the design of a complete and terminating strategy.

As in [1], an advantage of our approach is the ability to complete the original inference system; if the inference system is not good (hence we cannot apply directly our results), we may run a saturation procedure, that will yield a good inference system. Of course, such a procedure may not terminate, in which case we can still use our results on the limit (infinite) inference system, but getting an effective algorithm for solving deducibility constraints requires more work, which is out of the scope of this paper.

Related work. The present work is a strict extension of [7]: we consider a class of inference systems instead of a particular one. The intruder theories that are described by a subterm convergent rewriting system can also be casted as good inference systems (but the converse is false). Hence, as far as trace properties are concerned, we also generalize [2, 4]. Note however that [2] also considers equivalence properties, that are not covered (yet) by our work. There are also several examples of formal (intruder) proof systems that yield decidable deducibility constraints [16, 8, 6, 5, 10]. All these works consider AC-symbols and are incomparable with our results.

Structure of the paper. In Section 2, we introduce good inference systems and their properties. Then, in Section 3, we introduce deducibility constraints. In Section 4, we provide with a set of constraint transformation rules, that is parametrized by any good inference system and that we prove both sound and complete: the solutions of the constraint are the same as the solutions of the solved forms that are obtained by applying the transformation rules. In Section 5, we give a complete and terminating strategy. Due to a lack of space, the proofs are given in [3].

2 Preliminaries

In what follows, we assume that $\mathcal{F}$ is a (ranked) alphabet of *function symbols*. *Terms* are built on this set of function symbols and a set of variables $\mathcal{X}$. *Ground terms* are terms without variables. For any term t (and, by extension, set of terms or any formal expression), $var(t)$ is the set of variable symbols occurring in t and $st(t)$ denotes the set of *subterms* of t defined as usual.

2.1 Inference systems

We use a natural deduction style for inference systems. An *inference rule* consists in a finite set of terms $\{u_1, \ldots, u_n\}$, the *premises*, and a term u, the *conclusion* such that $var(u) \subseteq var(\{u_1, \ldots, u_n\})$. It is displayed

$$\frac{u_1 \quad \cdots \quad u_n}{u}$$

It may also be convenient to use a deducibility predicate symbol I, in which case the inference rules are simply Horn clauses $I(u_1), \ldots, I(u_n) \to I(u)$.

Example 1. Consider the signature $\mathcal{F} = \{\mathtt{enc}/3, \mathtt{pub}/1, \mathtt{priv}/1, \langle\,,\,\rangle/2\}$. The symbols $\mathtt{enc}$ and $\langle\,,\,\rangle$ represent respectively probabilistic encryption and pairing, $\mathtt{pub}$ (resp. $\mathtt{priv}$) represents the public key (resp. private key) construction. A possible set of "Dolev-Yao" inference rules for public-key encryption is:

$$(\mathsf{E})\ \frac{x \quad y \quad z}{\mathtt{enc}(x,y,z)} \qquad (\mathsf{D})\ \frac{\mathtt{enc}(\mathtt{pub}(y),x,z) \quad \mathtt{priv}(y)}{x} \qquad (\mathsf{K})\ \frac{x}{\mathtt{pub}(x)}$$

$$(\mathsf{P})\ \frac{x \quad y}{\langle x,y\rangle} \qquad (\mathsf{Proj}_1)\ \frac{\langle x,y\rangle}{x} \qquad (\mathsf{Proj}_2)\ \frac{\langle x,y\rangle}{y}$$

Other relevant examples of inference systems are obtained by adding signature schemes, hash functions, symmetric encryption ...

A *proof*, with hypotheses H and conclusion t is a tree, whose nodes are labeled with terms and such that, if a node is labeled s and its sons are labeled $s_1, \ldots, s_n$, then either $n = 0$ (this is a leaf node), and $s \in H$, or else there is an inference rule whose premises are $u_1, \ldots, u_n$ and conclusion is u and a substitution θ such that $u\theta = s$ and, for every i, $u_i\theta = s_i$. We write $H \vdash t$ when there exists a proof with hypotheses H and conclusion t.

We let $\mathtt{step}(\pi)$ be the set of terms labeling the proof π and $\mathtt{leaves}(\pi)$ be the multiset of the terms that labels the leaves of π. If π is a proof, we let $\mathtt{last}(\pi)$ be the last inference step in π, $\mathtt{premises}(\pi)$ be the proofs of the premises of $\mathtt{last}(\pi)$ and $\mathtt{conc}(\pi)$ be its conclusion. More formally,

$$\text{if} \quad \pi = \frac{\pi_1 \quad \cdots \quad \pi_n}{u} \quad \text{then} \quad \begin{cases} \mathtt{last}(\pi) = \dfrac{\mathtt{conc}(\pi_1) \quad \cdots \mathtt{conc}(\pi_n)}{u} \\ \mathtt{premises}(\pi) = \{\pi_1, \ldots, \pi_n\} \\ \mathtt{conc}(\pi) = u \end{cases}$$

Example 2. Consider the following proof tree π. Actually, π is a proof in the Dolev-Yao inference system presented in Example 1.

$$\dfrac{\mathtt{enc}(\mathtt{pub}(k), a, r) \quad \dfrac{\langle \mathtt{priv}(k), a\rangle}{\mathtt{priv}(k)}\,(\mathsf{Proj}_1)}{a}\,(\mathsf{D})$$

We have that $\mathtt{premises}(\pi) = \{\mathtt{enc}(\mathtt{pub}(k), a, r), \dfrac{\langle \mathtt{priv}(k), a\rangle}{\mathtt{priv}(k)}\}$, $\mathtt{conc}(\pi) = a$, $\mathtt{last}(\pi) = \dfrac{\mathtt{enc}(\mathtt{pub}(k), a, r)\ \mathtt{priv}(k)}{a}$, $\mathtt{leaves}(\pi) = \{\mathtt{enc}(\mathtt{pub}(k), a, r), \langle \mathtt{priv}(k), a\rangle\}$.

2.2 Good inference systems

In the following definitions, we introduce our notion of saturation. Informally, if there is a proof such that some intermediate step is too large (we call this a bad proof and the large step is called a bad pattern), then there must be a simpler proof of the same statement.

If $\mathsf{R} = \dfrac{s_1 \quad \cdots \quad s_n}{s_0}$ is an inference rule, we let $\mathtt{Max}(\mathsf{R})$ be the multiset of the maximal terms s_i, w.r.t. the subterm ordering $\lhd$.

Definition 1 (bad proof / pattern). *A* bad proof *is a proof π of the form:*

$$\dfrac{u_1 \quad \cdots \quad u_n \qquad \dfrac{v_1 \quad \cdots \quad v_m}{u_{n+1}}\,\mathsf{R}_1 \qquad u_{n+2} \quad \cdots \quad u_{n+k}}{v}\,\mathsf{R}_2$$

such that $\mathsf{R}_1 = \dfrac{s_1 \quad \cdots \quad s_m}{s}$, $\mathsf{R}_2 = \dfrac{t_1 \quad \cdots \quad t_{n+k}}{t}$, $s \in \mathit{Max}(\mathsf{R}_1)$ *and* $t_{n+1} \in \mathit{Max}(\mathsf{R}_2)$. *A* bad pattern *in a proof π is a subproof of π of the form:*

$$\dfrac{\pi_2^1 \quad \cdots \quad \pi_2^{i-1} \qquad \dfrac{\pi_1^1 \quad \cdots \quad \pi_1^m}{\mathit{conc}(\pi_2^i)}\,\mathsf{R}_1 \qquad \pi_2^{i+1} \quad \cdots \quad \pi_2^n}{v}\,\mathsf{R}_2$$

such that the following proof is a bad proof.

$$\dfrac{\mathit{conc}(\pi_2^1) \ \cdots \ \mathit{conc}(\pi_2^{i-1}) \qquad \dfrac{\mathit{conc}(\pi_1^1) \ \cdots \ \mathit{conc}(\pi_1^m)}{\mathit{conc}(\pi_2^i)} \qquad \mathit{conc}(\pi_2^{i+1}) \ \cdots \ \mathit{conc}(\pi_2^n)}{v}$$

If $\pi = \mathsf{R}\theta$ is an instance of an inference rule R and $\mathtt{Max}(\mathsf{R}) = \{s_1, \ldots, s_k\}$, then $\mu(\pi)$ is the multiset $\{s_1\theta, \ldots, s_k\theta\}$. If π is a proof, $\mu(\pi)$ is defined as the multiset of $\mu(\pi')$ for all inference steps π' of π. Formally, if $\mathtt{premises}(\pi) = \{\pi_1, \ldots, \pi_n\}$, we have that:

$$\mu(\pi) = \mu(\pi_1) \uplus \cdots \uplus \mu(\pi_n) \uplus \mu(\mathtt{last}(\pi)).$$

Multisets are ordered using the multiset extensions of their elements: if $\succeq$ is an ordering, we let $\succeq_m$ be its multiset extension.

Definition 2 (good inference system). *An inference system is* good *if there is a total well-founded extension $\prec$ of the subterm ordering $\lhd$ such that, for any bad proof π, there is a proof π' of $\mathtt{leaves}(\pi) \vdash \mathtt{conc}(\pi)$ with $\mathtt{leaves}(\pi') \subseteq \mathtt{leaves}(\pi)$ (multiset inclusion), $\mu(\pi')\ (\prec_m)_m\ \mu(\pi)$, and $\mu(\pi') \neq \mu(\pi)$.*

Example 3. The Dolev-Yao inference system described in Example 1 is good. Indeed, all bad proofs are of one of the following forms

$$\frac{\dfrac{\mathtt{pub}(u_1)\quad u_2\quad u_3}{\mathtt{enc}(\mathtt{pub}(u_1), u_2, u_3)}\quad \mathtt{priv}(u_1)}{u_2} \qquad \frac{\dfrac{u_1 \quad u_2}{\langle u_1, u_2\rangle}}{u_i}\quad i = 1, 2$$

Obviously, for all such π there is a smaller, trivial, proof π' of $\mathtt{leaves}(\pi) \vdash \mathtt{conc}(\pi)$ such that $\mathtt{leaves}(\pi') \subseteq \mathtt{leaves}(\pi)$ and $\mu(\pi')\ (\prec_m)_m\ \mu(\pi)$ for any total well-founded extension $\prec$ of the subterm ordering.

Now, we give another example in which the inference system is no longer finite. We consider blind signatures, as described in [9], that are used in some e-voting protocols. The inference system is not good. However, we may complete it and get an infinite, yet recursive, good inference system.

Example 4. We add the following rules to the system of Example 1:

$$(\mathsf{S})\ \frac{x \quad y}{\mathtt{sign}(x, y)} \qquad (\mathsf{C})\ \frac{\mathtt{sign}(x, y)}{x}$$

$$(\mathsf{B})\ \frac{x \quad y}{\mathtt{blind}(x, y)} \qquad (\mathsf{UB}_1)\ \frac{\mathtt{blind}(x, y)\quad y}{x} \qquad (\mathsf{UB}_2)\ \frac{\mathtt{sign}(\mathtt{blind}(x, y), z)\quad y}{\mathtt{sign}(x, z)}$$

Because of the rule (UB_2), the system not good. The following proof π is bad:

$$\frac{\dfrac{\mathtt{sign}(\mathtt{blind}(\mathtt{blind}(x, x_1), x_2), y) \quad x_2}{\mathtt{sign}(\mathtt{blind}(x, x_1), y)} \quad x_1}{\mathtt{sign}(x, y)}$$

There is no other proof π' of $\mathtt{leaves}(\pi) \vdash \mathtt{sign}(x, y)$ such that $\mathtt{leaves}(\pi') \subseteq \mathtt{leaves}(\pi)$. Thus, for any total well-founded extension $\prec$ of $\lhd$, there is no proof π' of $\mathtt{leaves}(\pi) \vdash \mathtt{sign}(x, y)$ such that $\mathtt{leaves}(\pi') \subseteq \mathtt{leaves}(\pi)$ and $\mu(\pi')\ (\prec_m)_m\ \mu(\pi)$. However, we may add all shortcuts that correspond to bad proofs. Let $b_n(x, x_1, \ldots, x_n)$ be defined by $b_1(x, x_1) = \mathtt{blind}(x, x_1)$ and $b_{n+1}(x, x_1, \ldots, x_{n+1}) = \mathtt{blind}(b_n(x, x_1, \ldots, x_n), x_{n+1})$. We add the following rules (for every $n \geq 1$) and the resulting system is a good inference system.

$$\frac{\mathtt{sign}(b_n(x, x_1, \ldots, x_n), y)\quad x_1\ \ldots\ x_n}{\mathtt{sign}(x, y)}$$

2.3 Some properties of good inference systems

Definition 3 (simple proof). *Let $H_1 \subseteq H_2 \subseteq \cdots \subseteq H_n$. A proof π of $H_i \vdash u$ is* left-minimal *if for any $j < i$ such that $H_j \vdash u$, π is a proof of $H_j \vdash u$. A proof is* simple *if it does not contain any bad pattern and all its subproofs are left-minimal.*

Example 5. Consider the Dolev-Yao inference system given in Example 1. Let $H_1 = \{\mathtt{enc}(\mathtt{pub}(k), a, r), \mathtt{priv}(k), a\}$, $H_2 = H_1 \cup \{\langle a, b\rangle\}$. We have that $H_2 \vdash a$. Indeed, the proofs π_1, π_2 and π_3 described below are witnesses of this fact:

$$\frac{\langle a, b\rangle}{a} \qquad \frac{\mathtt{enc}(\mathtt{pub}(k), a, r) \quad \mathtt{priv}(k)}{a} \qquad \frac{\dfrac{a \quad a}{\langle a, a\rangle}}{a}$$

The proofs π_2 is simple whereas π_1 and π_3 are not. Note that proof of $H_2 \vdash a$ reduced to a leaf is also a simple proof.

Lemma 1. *Consider a good inference system. Let $H_1 \subseteq H_2 \subseteq \cdots \subseteq H_n$ be an increasing sequence of sets of terms and $i \in \{1, \ldots, n\}$. If π is a proof of $H_i \vdash u$ then there is a simple proof of $H_i \vdash u$.*

From now on, we only consider good inference systems. The rules of such systems can be divided in three sets.

- The *composition rules* whose conclusion is the only maximal term. Any rule $I(x_1), \ldots, I(x_n) \to I(\mathsf{f}(x_1, \ldots, x_n))$ is a composition, e.g. (P), (E), (S).
- The *decomposition rules* whose all maximal terms are premises, e.g. (D).
- The *versatile rules* whose both the conclusion and some premises are maximal, e.g. (UB_2).

In what follows, we also assume that:

1. any composition rule has a conclusion $\mathsf{f}(x_1, \ldots, x_n)$ where $x_1, \ldots, x_n$ are variables. This is the case in our application area: each function symbol is either public (and there is such a rule) or private.
2. any versatile rule satisfies the following properties:
   (a) each strict subterm of the conclusion is a subterm of some premise.
   (b) each premise that is not maximal in the rule is a strict subterm of another premise of that rule.

These conditions are satisfied in Examples 1 and 4. Besides these examples, any intruder theory that can be presented by a finite subterm-convergent rewrite system satisfies our hypotheses. These hypotheses might not be necessary for our result, but we use them in our proof.
We now classify the proofs, according to the type of the last proof step. This generalizes the classical composition/decomposition classification:

Lemma 2 (locality). *Let π be a proof of $H \vdash u$ without bad pattern, one of the following occurs:*

- $\mathtt{last}(\pi)$ *is a composition and* $\mathtt{step}(\pi) \subseteq st(H \cup \{u\})$*;*
- π *is reduced to a leaf or* $\mathtt{last}(\pi)$ *is a decomposition and* $\mathtt{step}(\pi) \subseteq st(H)$*;*
- $\mathtt{last}(\pi)$ *is (an instance of) a versatile rule and* $\mathtt{step}(\pi') \subseteq st(H)$ *for any strict subproof* π' *of* π*.*

This is proved by observing that any proof in which a maximal conclusion is also a maximal premise of the next rule can be simplified, according to the definition of good inference systems.

3 Deducibility constraints

The following definition of (deducibility) constraints has been proved to be relevant in the context of security protocols verification (see, e.g. [15, 16, 7]).

Definition 4. *A* constraint system D *is a formula of the form* $\exists \tilde{z}.[C \mid \mathsf{E}]$ *where:*

- $\tilde{z}$ *is a sequence of variables;*
- $\mathsf{E}(D) = \mathsf{E}$ *is a set of equations in solved form, identified to a substitution* θ_{E}*;*
- C *is a conjunction of deducibility constraints* $H_1 \Vdash u_1 \wedge \ldots \wedge H_n \Vdash u_n$ *where* $var(C) \cap dom(\theta_{\mathsf{E}}) = \emptyset$*,* $H_1, \ldots, H_n$ *are finite sets of terms,* $u_1, \ldots, u_n$ *are terms, and such that* monotony *and* origination *are satisfied:*
  - Monotony*:* $\emptyset \neq H_1 \subseteq H_2 \subseteq \ldots \subseteq H_n$*;*
  - Origination*:* $var(H_i) \subseteq var(\{u_j \mid H_j \subsetneq H_i\})$ *for* $1 \leq i \leq n$*.*

We let $fvar(D) = var(D) \smallsetminus \tilde{z}$ *and* $\mathsf{LH}(D) = \{H_1, \ldots, H_n\}$*.*

Definition 5 (solution). *Given an inference system,* a solution of *a constraint system* $D = \exists \tilde{z}.[C \mid \mathsf{E}]$ *is a ground substitution* σ *with* $dom(\sigma) = fvar(D)$ *such that there is a ground substitution* τ *with* $dom(\tau) = \tilde{z}$ *such that:*

- $H(\sigma \cup \tau) \vdash u(\sigma \cup \tau)$ *for every* $H \Vdash u \in C$*, and*
- $u(\sigma \cup \tau) = v(\sigma \cup \tau)$ *for every* $u = v \in \mathsf{E}$*.*

We let $Sol(D)$ *be the set of solutions of* D*.*

In the context of security protocols, any solution will correspond to a choice of messages that are constructed by the attacker and that are accepted by the honest parties.

Example 6. We consider the Dolev-Yao inference system given in Example 1.

$$D := \begin{cases} H_1 = a \Vdash x_0 \ \wedge \ a \Vdash x_1 \\ H_2 = \mathtt{enc}(x_0, \langle b, x_1 \rangle, r),\ \mathtt{priv}(a),\ a \Vdash b \end{cases}$$

$H_1 \subseteq H_2$ and the variables x_0, x_1 occur first on the right. Thus, D is a constraint system. $\sigma = \{x_0 \mapsto \mathtt{pub}(a), x_1 \mapsto \langle a, a \rangle\}$ is a solution of D. Here, there are no bounded variables nor equations. This is the case for constraint systems that represent the security protocol executions. Bounded variables and equations may however be introduced by our constraint solving rules.

Putting together Definitions 4 and 5, we get the following problem, whose decision is the subject of this paper:

> Given an inference system and a constraint system D, does there exist a substitution σ such that $\sigma \in Sol(D)$? We also want to find an effective representation of all solutions.

Notation. Let $D = \exists\tilde{z}.[C \mid \mathsf{E}]$ be a constraint system. For every variable $x \in var(D)$, we let H_x be the smallest set $H \in \mathsf{LH}(D)$ such that there is a constraint $H \Vdash u \in D$ with $x \in var(u) \smallsetminus var(H)$. In other words, H_x is the left hand side of the deducibility constraint that introduced the variable x for the first time. By origination and monotony, this is defined for all $x \in var(C)$. By convention, $H_x = \emptyset$ when x does not occur in C.

4 Transformation of deducibility constraints

We show here that we can solve deducibility constraints in such a way that *we do not miss any solution* (as in [7]). The basic idea of the transformation rules is very straightforward, and that is what makes it appealing: we simply guess the last step of the proof, performing a backwards proof search together with narrowing the variables of the constraint. If $\mathsf{R} = I(u_1), \ldots, I(u_n) \to I(u)$ is guessed as the last rule in the proof of $H\sigma \vdash v\sigma$, we simply perform:

$$\exists\tilde{z}.[C \wedge H \Vdash v \mid \mathsf{E}] \rightsquigarrow \exists\tilde{z'}.[C\theta \wedge H\theta \Vdash u_1\theta \wedge \ldots \wedge H\theta \Vdash u_n\theta \mid \mathsf{E}']$$

where $\tilde{z'} = \tilde{z} \cup var(\mathsf{R})$, $\theta = \mathsf{mgu}(u, v)$ and $\mathsf{E}' = \mathsf{E} \cup \theta$.

This hardly terminates, even for very simple proof systems and ground goals. Consider the rule (Proj_1) only. We get:

$$\begin{aligned} H \Vdash v &\rightsquigarrow \exists x_1, x_2.[H \Vdash \langle v, x_2\rangle \mid x_1 = v] \\ &\rightsquigarrow \exists x_1, x_2, y_1, y_2.[H \Vdash \langle\langle v, x_2\rangle, y_2\rangle \mid x_1 = v \wedge y_1 = \langle v, x_2\rangle] \;\rightsquigarrow\; \ldots \end{aligned}$$

And similarly for (P) (below, we assume that H is a ground set of terms):

$$H \Vdash x \rightsquigarrow \exists x_1, x_2.[H \Vdash x_1 \wedge H \Vdash x_2 \mid x = \langle x_1, x_2\rangle] \rightsquigarrow \ldots$$

First, we do not aim at explictly enumerating all possible solutions, but only compute *solved forms*, that are a convenient representation of all these solutions. Typically, $H \Vdash v$ will be solved when v is a variable. This rules out the second above non-terminating example.

For decomposition or versatile rules, we may still get the first non-terminating behavior. That is where we use locality: we control the application of such rules, roughly requesting that maximal premises are subterms of H. This is however not complete, as Lemma 2 shows only that, in case of a versatile or decomposition rule, the premises are subterms of the hypotheses *at the ground level.* In other words, if we guessed that the last rule, in the proof of an instance σ of $H \Vdash v$, is a decomposition, we only know that the premises of the last proof step are in $st(H\sigma)$. We use then the property of subterms: $st(H\sigma) = st(H)\sigma \cup st(\sigma)$. If the premises are in $st(H)\sigma$, everything is fine: we can guess subterms of H that are the premises. Otherwise, it is not so straightforward, as σ is unknown. That is where we need some additional strategies.

Another difficulty comes from the introduction of variables. If we keep on introducing variables and equations, the left hand sides of deducibility constraints may grow, hence their subterms too. Then guessing a subterm of H as a premise does not necessarily yield a bounded number of terms.

4.1 Transformation rules

The rules of Figure 1 are applied non-deterministically. When new variables are introduced (in the Dec rule) they are assumed to be fresh, by renaming.

(Axiom) $\exists\tilde{z}.[C \wedge H \Vdash u \mid \sigma] \rightsquigarrow \exists\tilde{z}.[C\theta \mid \sigma \cup \theta]$
where $\theta = \mathsf{mgu}(u, v)$, $v \in H$ and $u \notin \mathcal{X}$

(Triv) $\exists\tilde{z}.[C \wedge H \Vdash x \wedge H' \Vdash x \mid \sigma] \rightsquigarrow \exists\tilde{z}.[C \wedge H \Vdash x \mid \sigma]$
when $H \subseteq H'$

(Comp) $\exists\tilde{z}.[C \wedge H \Vdash \mathsf{f}(u_1, \dots, u_n) \mid \sigma] \rightsquigarrow \exists\tilde{z}.[C \wedge H \Vdash u_1 \wedge \dots \wedge H \Vdash u_n \mid \sigma]$
if f is a public symbol

(Dec) $\exists\tilde{z}.[C \wedge H \Vdash v \mid \sigma] \rightsquigarrow \exists\tilde{z} \cup \tilde{x}.[C\theta \wedge H\theta \Vdash w_1\theta \wedge \dots H\theta \Vdash w_n\theta \quad \mid \sigma \cup \theta]$
$\wedge H'\theta \Vdash v_1\theta \wedge \dots \wedge H'\theta \Vdash v_m\theta$

where:

- $\mathsf{R} = \dfrac{v_1 \;\dots\; v_m \;\; w_1 \;\dots\; w_n}{w}$ is a decomposition or a versatile rule such that $\mathtt{Max}(\mathsf{R}) \subseteq \{w_1, \dots, w_n\}$ and $\tilde{x} = var(\mathsf{R})$;
- $\theta = \mathsf{mgu}(\langle w, w_1, \dots, w_n\rangle, \langle v, u_1, \dots, u_n\rangle)$, $u_1, \dots, u_n \in st(H) \smallsetminus \mathcal{X}$, and $v \notin \mathcal{X}$;
- H' is a left member of a deducibility constraint such that $H' \subsetneq H$.

Fig. 1. Transformation of deducibility constraints

The Dec rule deserves some explanation. We guessed here a versatile or decomposition rule. The premises $w_1, \dots, w_n$ will be those whose instances correspond to a term in $st(H)\sigma$: we can guess the corresponding terms in $st(H)$, namely $u_1, \dots, u_n$. The other premises (that are then subterms in the substitution part) are constrained to be proved with strictly less hypotheses. We will show that this can always be assumed, hence that we get completeness.

Example 7. Consider the constraint system D given in Example 6. First, considering the rule (Proj_1) and applying Dec to the third constraint yields:

$\exists x', y'.[a \Vdash x_0, a \Vdash x_1, \mathtt{enc}(x_0, \langle b, x_1\rangle, r), \mathtt{priv}(a), a \Vdash \langle b, x_1\rangle \mid \{x' \mapsto b, y' \mapsto x_1\}]$.

Now, considering (D) and applying again Dec to the third constraint yields:

$$D' = \begin{cases} \exists x, y, z, x', y'.[\, a \Vdash x_0\theta, \quad a \Vdash x_1\theta \\ \qquad\qquad\qquad H_2\theta \Vdash \mathtt{enc}(x_0\theta, \langle b, x_1\theta\rangle, r) \\ \qquad\qquad\qquad H_2\theta \Vdash \mathtt{priv}(a) \ \mid \ \theta \cup \{x' \mapsto b, y' \mapsto x_1\}] \end{cases}$$

where $\theta = \mathsf{mgu}(\langle x, \mathtt{enc}(\mathtt{pub}(y), x, z), \mathtt{priv}(y)\rangle, \langle\langle b, x_1\rangle, \mathtt{enc}(x_0, \langle b, x_1\rangle, r), \mathtt{priv}(a)\rangle)$
$= \{x \mapsto \langle b, x_1\rangle,\ y \mapsto a,\ z \mapsto r,\ x_0 \mapsto \mathtt{pub}(a)\}$.

Lemma 3 (soundness). *Let D be a constraint system such that $D \rightsquigarrow D'$, then D' is a constraint system and $Sol(D') \subseteq Sol(D)$.*

The transformation rules of Figure 1 also preserve the following invariant.

Definition 6 (uniquely determined). *Let $D = \exists\tilde{z}.[C \mid \mathsf{E}]$ be a constraint system. D is* uniquely determined *if for any ground substitution σ such that $dom(\sigma) = fvar(D)$, there are ground terms $u_1, \ldots, u_\ell$ such that either $\mathsf{mgu}(\mathsf{E}\sigma) = \perp$ or $\mathsf{mgu}(\mathsf{E}\sigma) = \{z_1 = u_1, \ldots, z_\ell = u_\ell\}$ where $\tilde{z} = \{z_1, \ldots, z_\ell\}$. In that case we let $\overline{\sigma}$ be $\sigma \cup \mathsf{mgu}(\mathsf{E}\sigma)$. ($\overline{\sigma}$ is a solution of the constraint system $[C \mid \mathsf{E}]$.)*

Example 8. Let D' be the constraint system given in Example 7. We have that $fvar(D) = \{x_0, x_1\}$. Once values are assigned to x_0, x_1, there is a unique substitution τ that satisfies the equations in $\mathsf{E}(D')$.

Lemma 4. *Let D be a constraint system that is uniquely determined and D' be such that $D \rightsquigarrow D'$. Then D' is uniquely determined.*

Using our transformation rules, solving deducibility constraint systems can be reduced to solving simpler constraint systems that we call solved.

Definition 7. *A constraint system $D = \exists\tilde{z}.[H_1 \Vdash x_1 \wedge \ldots \wedge H_n \Vdash x_n \mid \mathsf{E}]$ is in* solved form *when $x_1, \ldots, x_n$ are distinct variables.*

Solved deducibility constraint systems are particulary simple since they always have a solution.

Lemma 5. *A solved form has always at least one solution.*

4.2 Completeness

Let $H_1 \subseteq H_2 \subseteq \ldots \subset H_n$be a sequence of sets of terms. Let π be a proof of $H_i \vdash u$ for some i $(1 \leq i \leq n)$. We associate to π, the minimal set $\mathtt{Hyp}(\pi) \in \{H_1, \ldots, H_n\}$ containing the leaves of π. Note that $\mathtt{Hyp}(\pi) \subseteq H_i$. Given a constraint system $D = \exists\tilde{z}.[C \mid \mathsf{E}]$ that is uniquely determined and a solution σ, a simple proof w.r.t. D is a simple proof w.r.t. the sequence of sets of terms $\mathsf{LH}(D)\overline{\sigma}$.

We first show that either the subterms occurring in proofs are subterms of the hypotheses, or else their simple proofs end with a composition or a versatile rule.

Lemma 6. *Let D be a constraint system of the form $[C \mid \mathsf{E}]$. Let $H \in \mathsf{LH}(D)$ be such that for every $y \in var(H)$ there is a constraint $H_y \Vdash y \in D$. Let σ be a solution of D and v be a term such that $H\sigma \vdash v$. Let $u \in st(v)$. Then:*

1. *either $u \in (st(H) \smallsetminus \mathcal{X})\sigma$;*
2. *or $H\sigma \vdash u$ and any simple proof π of $H\sigma \vdash u$ ends with a composition or a versatile rule.*

To prove this lemma, we consider the set Π of simple proofs of $H\sigma \vdash v$ and we prove the lemma by induction on the pair (H, d) where d is the size of a minimal proof in Π.

Now, we define the complexity of the proofs witnessing that σ is a solution of D and show that there is always a rule yielding a strictly smaller complexity, until we reach a solved form. Let $D = \exists \tilde{z}.[C \mid \mathsf{E}]$ be a uniquely determined constraint system and σ be a solution of D.

- If $H \Vdash u \in C$ then $\mathsf{PS}(H \Vdash u, \sigma)$ is the size (i.e. number of nodes) of a simple proof of $H\overline{\sigma} \vdash u\overline{\sigma}$ that has a minimal size.
- If $\mathsf{LH}(D) = \{H_1, \ldots, H_n\}$ with $H_1 \subsetneq \ldots \subsetneq H_n$, then the *level* $\mathsf{lev}(H \Vdash u, D)$ of a deducibility constraint $H \Vdash u \in C$ is the index i such that $H = H_i$.

The measure PS is extended to constraint systems by letting, for any solution σ of D, $\mathsf{PS}(D, \sigma)$ be the multiset of pairs $(\mathsf{lev}(H \Vdash u, D), \mathsf{PS}(H \Vdash u, \sigma))$ for all deducibility constraints $H \Vdash u \in D$. The multisets $\mathsf{PS}(D, \sigma)$ are compared using the multiset extension of the lexicographic composition of the orderings.

Note that the number of different levels in a constraint system might decrease, but it may never increase.

Lemma 7. *If D is a constraint system that is uniquely determined and $\sigma \in Sol(D)$, then either D is in solved form or else there is a D' such that $D \rightsquigarrow D'$, $\sigma \in Sol(D')$, and $\mathsf{PS}(D, \sigma) > \mathsf{PS}(D', \sigma)$.*

Proof. (sketch) If D is not in solved form, there must be a constraint $H \Vdash u \in D$ such that u is not a variable. We consider such a constraint, with a minimal left hand side. Then, depending on the last rule of a minimal size simple proof of $H\sigma \vdash u\sigma$, we may apply some transformation, that yields a smaller PS:

- *If the proof is reduced to a leaf*, then we use the Axiom rule.
- *If the last rule is a composition*, then we apply Comp to D, yielding a smaller PS.
- *If the last rule is versatile or a decomposition*, we have to show that the conditions of Dec are met in order to conclude. To prove this, we rely on Lemma 2 and Lemma 6. ◻

Then, we prove the following lemma by induction on $\mathsf{PS}(D, \sigma)$, applying Lemma 7 for the induction step. Lemma 4 allows us to ensure that the resulting constraint system is uniquely determined and to apply our induction hypothesis.

Lemma 8 (completeness). *If D is a constraint system that is uniquely determined and $\sigma \in Sol(D)$, then there is a solved deducibility constraint D' such that $D \rightsquigarrow^* D'$ and $\sigma \in Sol(D')$.*

(Active) $\exists\tilde{z}.[\mathsf{A} \mid \mathsf{F} \mid \mathsf{E}] \mapsto_{\mathsf{A}} \exists\tilde{z} \cup \tilde{x}.[\mathsf{A}' \mid \mathsf{F} \mid \mathsf{E} \cup \theta]$

if $\mathsf{A} \rightsquigarrow \exists\tilde{x}.[\mathsf{A}' \mid \theta]$ using Axiom, Triv, Comp; or Dec on $H \Vdash v$ and there exists $x \in var(v)$ such that $\mathsf{lev}(x, \mathsf{A}) = \mathsf{lev}(H, \mathsf{A})$.We assume that $\mathsf{mgu}(\mathsf{E} \cup \theta) \neq \bot$.

(Freeze) $\exists\tilde{z}.[\mathsf{A} \wedge H \Vdash v \mid \mathsf{F} \mid \mathsf{E}] \mapsto \exists\tilde{z} \cup \tilde{x}.[\mathsf{A} \wedge H \Vdash u_1 \wedge \ldots H \Vdash u_n \mid \mathsf{F} \wedge H' \Vdash v_1 \wedge \ldots \wedge H' \Vdash v_m \mid \mathsf{E} \cup \theta]$

where:

- $\mathsf{R} = \dfrac{v_1 \;\ldots\; v_m \;\; w_1 \;\ldots\; w_n}{w}$ is a decomposition or a versatile rule such that $\mathtt{Max}(\mathsf{R}) \subseteq \{w_1, \ldots, w_n\}$ and $\tilde{x} = var(\mathsf{R})$;
- $\theta = \mathsf{mgu}(\langle w, w_1, \ldots, w_n\rangle, \langle v, u_1, \ldots, u_n\rangle)$, $u_1, \ldots, u_n \in st(H) \smallsetminus \mathcal{X}$, and $v \notin \mathcal{X}$;
- H' is a left member of a deducibility constraint in A such that $H' \subsetneq H$;
- $\mathsf{mgu}(\mathsf{E} \cup \theta) \neq \bot$ and $\mathsf{lev}(x, \mathsf{A} \wedge H \Vdash v) < \mathsf{lev}(H, A \wedge H \Vdash v)$ for any $x \in var(v)$.

(Open) $\exists\tilde{z}.[\mathsf{A} \mid \mathsf{F} \mid \mathsf{E}] \mapsto_{\mathsf{O}} \exists\tilde{z}.[(\mathsf{A} \cup \mathsf{F})\theta \mid \emptyset \mid \theta]$

when A in solved form and $\theta = \mathsf{mgu}(\mathsf{E})$

Fig. 2. Transformation of extended constraint systems

5 Termination

In order to get termination, we add some control on the transformation rules.

5.1 Our strategy

For every variable x and constraint system D, the *level* $\mathsf{lev}(x, D)$ of x is the level of H_x in D, if $x \in var(D)$, and is 0 otherwise. The deducibility constraints of D are split into an *active part* $\mathsf{Act}(D)$ and a *frozen part* $\mathsf{Fr}(D)$.

Definition 8 (extended constraint system). *An* extended constraint system *D is a formula $\exists\tilde{z}.[\mathsf{A} \mid \mathsf{F} \mid \mathsf{E}]$ where:*

- *$\tilde{z}$ is a sequence of variables;*
- *$\mathsf{E}(D) \stackrel{\mathsf{def}}{=} \mathsf{E}$ is a set of equations (not necessarily in solved form) with $\mathsf{mgu}(\mathsf{E}) \neq \bot$;*
- *$\mathsf{Act}(D) \stackrel{\mathsf{def}}{=} \mathsf{A}$, the* active part *of D, and $\mathsf{Fr}(D) \stackrel{\mathsf{def}}{=} \mathsf{F}$, the* frozen part *of D, are sets of deducibility constraints; A and $(\mathsf{A} \cup \mathsf{F})\mathsf{mgu}(\mathsf{E})$ are constraint systems.*

Let $\theta = \mathsf{mgu}(\mathsf{E})$. A solution of $\exists\tilde{z}.[\mathsf{A} \mid \mathsf{F} \mid \mathsf{E}]$ is a solution of $\exists\tilde{z}.[(\mathsf{A} \cup \mathsf{F})\theta \mid \theta]$. The system D is in solved form when $\mathsf{Fr}(D) = \emptyset$ and $\mathsf{Act}(D)\theta$ is in solved form.

The rules are described in Figure 2. The transformation relation defined by these rules is denoted $\mapsto$. Sometimes, we use $\mapsto_{\mathsf{A/F}}$ instead of $\mapsto_{\mathsf{A}} \cup \mapsto_{\mathsf{F}}$.

In the initial constraint system, nothing is frozen. All rules only apply to the active part and all rules (except Dec) only modify the active part.

- When the rule Dec is applied to a constraint $H \Vdash v$ such that, for some variable $x \in var(v)$, we have that $\mathsf{lev}(x, \mathsf{Act}(D)) = \mathsf{lev}(H, \mathsf{Act}(D))$, it also contributes only to the active part.

– Otherwise, when for all $x \in var(v)$, $\mathsf{lev}(x, \mathsf{Act}(D)) < \mathsf{lev}(H, \mathsf{Act}(D))$, then only the constraints $H \Vdash u_1 \wedge \ldots \wedge H \Vdash u_n$ are kept in the active part, and the remainder falls in the frozen part.

When $\mathsf{Act}(D)$ is in solved form (and only then), we open the fridge and pour the frozen part into the active one, performing all necessary replacements.

First, we have to establish the soundness of the transformation rules. The main point is to show that the active part remains a constraint system.

Lemma 9 (soundness). *Let D be an extended constraint system such that $D \mapsto D'$ then D' is an extended constraint system and $Sol(D') \subseteq Sol(D)$.*

If there is a loop on the active part by using only Active and Freeze, i.e. $D \mapsto^* D_1 \mapsto^*_{\mathsf{A/F}} D_2$ and $\mathsf{Act}(D_1) = \mathsf{Act}(D_2)$, we remove all the branches that begin with this prefix. We will show that this strategy is both:

– *complete*: for any D, for any $\sigma \in Sol(D)$, there is a sequence $D \mapsto^* D'$ authorized by the strategy such that $\sigma \in Sol(D')$ and D' is in solved form.
– *terminating*: there are no infinite transformation sequences.

5.2 Termination of our strategy

Now, we clarify the role of the fridge, by showing that the level of a variable in the active part is never increasing. Moreover, if new variables are introduced in the active part, their level is strictly smaller than the level of an older variable, whose respective level strictly decreased.

Lemma 10. *If $D \mapsto_{\mathsf{A/F}} D'$, then:*

1. $\mathsf{lev}(x, \mathsf{Act}(D')) \leq \mathsf{lev}(x, \mathsf{Act}(D))$ *for every* $x \in var(\mathsf{Act}(D))$.
2. *Let* $M = var(\mathsf{Act}(D')) \smallsetminus var(\mathsf{Act}(D))$. *If* $M \neq \emptyset$ *then there exists* $x \in var(\mathsf{Act}(D))$ *such that* $\mathsf{lev}(z, \mathsf{Act}(D')) < \mathsf{lev}(x, \mathsf{Act}(D))$ *for any* $z \in M \cup \{x\}$.

Now, we get a first termination lemma:

Lemma 11. *There is no infinite transformation sequence without opening the fridge.*

The reason for this is that either we do not introduce variables in which case there must be a loop (this is forbidden by the strategy) or else, thanks to Lemma 10, there is a level ℓ such that the level of some variable at level ℓ strictly decreases, while all new variables have a strictly smaller level than ℓ.

Finally, we cannot open very often the fridge:

Lemma 12. *Let D be an extended constraint system such that $\mathsf{E}(D) = \mathsf{Fr}(D) = \emptyset$. In any transformation sequence starting with D, we can open at most $2|\mathsf{LH}(D)| + |\mathsf{MLH}(D)|$ times the fridge – $\mathsf{MLH}(D)$ is the maximal set of hypotheses in $\mathsf{LH}(D)$.*

The idea is that the maximal level variables x occur in a constraint $H \Vdash x$ when we are about to open the fridge. Furthermore, in the fridge, all variables have a strictly smaller level. It follows that $H \Vdash x$ will not participate any more in any transformation: further transformations only take place at lower levels. More precisely, we show that between two openings, the number of levels or the number of distinct terms in the maximal left hand side decreases; otherwise in the resulting system further transformations only take place at lower levels.

From the previous lemmas, we derive the following corollary.

Corollary 1 (termination). *Our strategy is strongly terminating: there is no infinite transformation sequence of (extended) deducibility constraints.*

5.3 Completeness of our strategy

For an extended constraint system $D = \exists\tilde{z}.[\mathsf{A} \mid \mathsf{F} \mid \mathsf{E}]$, we let:

$$\mathsf{Open}(D) = \exists\tilde{z}.[(\mathsf{A} \cup \mathsf{F})\mathsf{mgu}(\mathsf{E}) \mid \mathsf{mgu}(\mathsf{E})]$$

and we say that D is uniquely determined if $\mathsf{Open}(D)$ is uniquely determined. Note that by definition of Open, we have that $Sol(D) = Sol(\mathsf{Open}(D))$ and "uniquely determined" is preserved by $\mapsto$.

The following corollary ensures that PS is decreasing on the active part.

Corollary 2 (of Lemma 7). *Let D be a uniquely determined extended constraint system such that $\mathsf{Act}(D)$ is not in solved form and $\sigma \in Sol(D)$. Then there is a D' such that $D \mapsto_{\mathsf{A/F}} D'$ and $\sigma \in Sol(D')$.*

Moreover, $\mathsf{PS}(\mathsf{A}, \overline{\sigma}|_{var(\mathsf{A})}) > \mathsf{PS}(\mathsf{A}', \overline{\sigma}|_{var(\mathsf{A}')})$ where $\mathsf{A} = \mathsf{Act}(D)$, $\mathsf{A}' = \mathsf{Act}(D')$ and $\overline{\sigma}$ is the extension of σ w.r.t. D'.

Thanks to this, if there is a loop on the active part, we can close the branch, still having a complete strategy:

Lemma 13. *Our strategy is complete.*

6 Conclusion

We gave a simple set of transformation rules that allows to derive a complete and effective representation of all solutions of a deducibility constraint. This works for any good inference system that satisfies some additional syntactic conditions. We believe that this is the starting point of several further works:

1. It would be nice to remove the additional syntactic restrictions (or to prove that they are necessary)
2. Getting a full generalization of [1], requires to introduce predicate symbols.
3. We need to enrich the syntax of constraints, in order to get effective algorithms for infinite (recursive) good inference systems.

4. Our transformation rules are not only preserving all solutions, but also all simple proofs, i.e. some witnesses that they are indeed solutions. This suggests that the same transformation rules can be used for the decision of the symbolic equivalence of constraint systems.
5. Covering all current decision results requires an extension that includes AC-symbols.

References

1. D. Basin and H. Ganzinger. Automated complexity analysis based on ordered resolution. *Journal of the ACM*, 48(1):70–109, 2001.
2. M. Baudet. Deciding security of protocols against off-line guessing attacks. In *Proc. 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 16–25, Alexandria, Virginia, USA, 2005. ACM Press.
3. S. Bursuc, H. Comon-Lundh, and S. Delaune. Deducibility constraints. Research Report LSV-09-17, LSV, ENS Cachan, France, 2009. 36 pages.
4. Y. Chevalier and M. Kourjieh. Key substitution in the symbolic analysis of cryptographic protocols. In *Proc. 27th International Conference on Foundations of Software Technology and Theoretical Computer Science (FTS&TCS'07)*, volume 4855 of *LNCS*. Springer, 2007.
5. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with Diffie-Hellman exponentiation and products in exponents. In *Proc. 23th International Conference on Foundations of Software Technology and Theoretical Computer Science (FTS&TCS'03)*, volume 2914 of *LNCS*, 2003.
6. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with xor. In Kolaitis [12].
7. H. Comon-Lundh, V. Cortier, and E. Zalinescu. Deciding security properties of cryptographic protocols. application to key cycles. *Transaction on Computational Logic*, 2009. To appear.
8. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in preence of exclusive or. In Kolaitis [12].
9. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.
10. S. Delaune, P. Lafourcade, D. Lugiez, and R. Treinen. Symbolic protocol analysis for monoidal equational theories. *Information and Computation*, 206(2-4), 2008.
11. D. Kähler and R. Küsters. Constraint Solving for Contract-Signing Protocols. In *Proc. 16th International Conference on Concurrency Theory (CONCUR 2005)*, volume 3653 of *LNCS*, pages 233–247. Springer, 2005.
12. P. Kolaitis, editor. *18th Annual IEEE Symposium on Logic in Computer Science.* IEEE Comp. Soc., 2003.
13. D. McAllester. Automatic recognition of tractability in inference relations. *Journal of the ACM*, 40(2), 1993.
14. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*, 2001.
15. J. Millen and V. Shmatikov. Symbolic protocol analysis with products and Diffie-Hellman exponentiation. In *Proc. 16th Computer Security Foundation Workshop (CSFW'03)*, pages 47–62. IEEE Comp. Soc. Press, 2003.
16. M. Rusinowitch and M. Turuani. Protocol insecurity with a finite number of sessions, composed keys is NP-complete. *Theoretical Computer Science*, 1-3, 2003.