# Tree automata with one memory set constraints and cryptographic protocols ${ }^{\text {sh }}$ 

Hubert Comon ${ }^{\mathrm{a}, \mathrm{b}, *, 1}$, Véronique Cortier ${ }^{\mathrm{b}, 2}$<br>${ }^{\text {a Department of Computer Science, Gates 4B, Stanford University, CA 94305-9045, USA }}$<br>${ }^{\mathrm{b}}$ Laboratoire Spécification et Vérification, CNRS and Ecole Normale Supérieure de Cachan, 61 Avenue du Président Wilson, 94235 Cachan cedex, France

Received 30 November 2001; received in revised form 31 October 2003


#### Abstract

We introduce a class of tree automata that perform tests on a memory that is updated using function symbol application and projection. The language emptiness problem for this class of tree automata is shown to be in DEXPTIME.

We also introduce a class of set constraints with equality tests and prove its decidability by completion techniques and a reduction to tree automata with one memory.

Finally, we show how to apply these results to cryptographic protocols. We introduce a class of cryptographic protocols and show the decidability of secrecy for an arbitrary number of agents and an arbitrary number of (concurrent or successive) sessions, provided that only a bounded number of new data is generated. The hypothesis on the protocol (a restricted copying ability) is shown to be necessary: without this hypothesis, we prove that secrecy is undecidable, even for protocols without nonces.


© 2004 Elsevier B.V. All rights reserved.

[^0]
## 1. Introduction

Set constraints were introduced in the 1980s and have been studied thoroughly since, with applications to the analysis of programs of various styles (see [2] for a survey). Typically, the problem of interest is to decide the satisfiability of a conjunction of set expression inclusions $e \subseteq e^{\prime}$ in which the set expressions are built from variables and various constructions, including, e.g., projection. Although some set variables may occur several times in an expression, most classes of set constraints do not make it possible to write a set expression for a set of terms of the form $f(t, t)$, in which one subterm occurs more than once. One exception is the class of constraints studied in [6].

Our motivating interest is to develop classes of cryptographic protocols for which some form of secrecy is decidable. A historical class of decidable protocols are the so-called ping-pong protocols [15]. Although none of the protocols of [8] belongs to this class, pingpong protocols remain a decidable class, while most larger classes of security protocols are undecidable [5]. One of the main restrictions in [16,15] is that messages are built using unary symbols only. In contrast, many protocols of interest are written using a binary encryption symbol and a pairing function. Another restriction in $[16,15]$ is that each protocol participant is stateless: after a message is sent, the participant does not retain any memory of the contents of the message. This is a significant limitation since many protocols rely on challenge-response steps, that require memory. A previous investigation of ping-pong protocols with added state led to undecidability [19].

It is insightful to observe that Dolev and Yao's result [16] can be proved using set constraints. This suggests a generalization of their approach to trees. A technical complication, though, is that the generalization to trees is less expressive than one might expect: in the case of unary functions only, a function and its inverse are set inverses of each other, in the sense that $f\left(f^{-1}(X)\right)$ is precisely $X$. However, this is no longer true with trees: if $f_{1}^{-1}$ and $f_{2}^{-1}$ are the two projections corresponding to a binary function symbol $f$, the set $f\left(f_{1}^{-1}(X), f_{2}^{-1}(X)\right)$ contains pairs $f\left(t_{1}, t_{2}\right)$ which are not necessarily in $X$. In order to increase the expressiveness of set constraints with binary functions, we need a "diagonal construction", enabling us to test for equalities between the members of sets.

In this paper, we introduce a new class of set constraints, allowing limited diagonal constructions. This class is incomparable with the class sketched in [6]. We show that satisfiability is decidable for this class, allowing us to generalize Dolev and Yao's result to trees. More precisely, we define a class of cryptographic protocols whose decidability does not assume any bound on the number of sessions (whether concurrent or not), improving over former decision results, e.g. [3,27,24] (see [12] for a survey on decidability results for cryptographic protocols). We also allow compound keys. Protocols in the class assume a limited copying capability for the agents. More precisely, we assume that an agent can only blindly copy one piece of the received message in the message (s)he sends. By "blindly" we mean here, without any type knowledge; this notion will be made precise in the paper. Let us emphasize that this restriction is satisfied by almost all protocols that we found in the literature. We also prove that this restriction is necessary: secrecy becomes undecidable if we allow two blind copies.

Our class of set constraints does not capture all protocol concepts of interest. In particular, as can be seen from the survey [8], many authentication protocols make use of nonces or
time stamps, which we cannot express (more precisely, we have to assume that there is a bounded number of nonces produced by each principal in any combination of sessions). On the other hand, properties of protocols that are modeled using set constraints are decidable, while nonces and timestamps typically lead to undecidability [5]. Moreover, we can express conservative approximations of general protocols, and it is possible in principle that set constraints with equality tests provide algorithms for determining the security of some such protocols.

We prove the decidability of set constraints with equality tests by a reduction to an emptiness problem for a class of tree automata with constraints. Tree automata with various forms of constraints have been studied by several authors (see [11] for a survey). However, the class we consider in this paper is incomparable with known decidable classes. Roughly, we allow each state to hold one arbitrarily large memory register and restrict the use of this memory to equality tests. Since memory registers are updated using projections and function application, this class is a generalization of pushdown word (alternating) automata. Despite the generality of the class, there is a simple proof that the emptiness decision problem is in DEXPTIME.

We start Section 2.1 by introducing Dolev and Yao result and its formulation in terms of set constraints. In Section 2.2, we recall (one possible) formal semantics of cryptographic protocols. We also prove that, even in the absence of nonces, secrecy is undecidable.

In Section 3.1, we recall classical results on definite set constraints and generalize them to set constraints with nonemptiness guards in Section 3.2. The results of this last section are used in the following sections.

In Section 4, we introduce tree automata with one memory and we prove some decidability results, relying on definite set constraints with non-emptiness guards. This can be seen as a stand-alone decidability result.

Next, we introduce in Section 5 our class of set constraints with one equality, showing how to reduce the satisfiability of these constraints to the nonemptiness decision for tree automata with one memory. The reduction is similar to the saturation process described in [7] for set constraints with intersection, but it is more complicated due to equality tests.

In Section 6.1 we define our class of cryptographic protocols and show how to apply the results of the previous sections to prove that secrecy is decidable for this class.

Several technical proofs, which are not interesting by themselves, are pushed to appendices.

## 2. Protocol motivation

### 2.1. Dolev-Yao's result

Dolev and Yao [16] consider protocols in which each principal holds a single public key (which is known to everybody) and a corresponding private key that is known to them only. The principals are able to build messages using plain text, encryption $e_{X}$ with the public key of $X$ and signatures $d_{X}$ appending the name of principal $X$. Here is a simple example
from [16]:
Example 1 (Dolev and Yao [16]).

$$
A \rightarrow B: e_{B}\left(d_{A}\left(e_{B}(s)\right)\right)
$$

$$
B \rightarrow A: e_{A}(s)
$$

Alice sends to Bob a message encrypted using Bob's public key consisting of a signed encrypted text $s$

Bob acknowledges the reception by sending back to Alice the text $s$, encrypted using the public key of Alice

In this model, communication channels are insecure. This allows an intruder to intercept messages, remember them, and replace them with alternate (possibly forged) messages. The intruder may decrypt a message if the corresponding key has become known to him, may append or remove signatures, and may encrypt using any public key. The secrecy question asks whether there is a way for an intruder to get the plain text message $s$ that is supposed to be kept secret between Alice and Bob. In the above example, the answer is yes (the protocol is insecure). For example, Dolev and Yao give the following attack: After a first session of the protocol, the intruder, $I$, who overhears the messages exchanged during that session, sends to $A$ the message $e_{A}\left(d_{I}\left(e_{A}(s)\right)\right)$, which he can build using the reply from Bob, and receives $e_{I}(s)$ in return.

The possible use of set constraints in cryptographic protocols analysis has been suggested in several papers, e.g. [20]. It is however interesting to see that the Dolev-Yao decidability proof can be summarized using set constraints by letting $I$ be the set of messages that can be built by the intruder (after any number of sessions). Since $I$ can intercept any message of any run of the protocol, we write set constraints putting every protocol message in $I$. For the example protocol above, we have

$$
e_{Y}\left(d_{X}\left(e_{Y}(s)\right)\right) \subseteq I, \quad e_{X}\left(e_{Y}^{-1}\left(d_{X}^{-1}\left(e_{Y}^{-1}(I)\right)\right)\right) \subseteq I
$$

for every pair of principals $X, Y$, since Bob acknowledges a message $m$ from Alice by sending $e_{A}\left(e_{B}^{-1}\left(d_{A}^{-1}\left(e_{B}^{-1}(m)\right)\right)\right)$. Finally, for every principal $X$, we express the ability of the intruder to perform operations using public information about $X$ :

$$
d_{X}(I) \subseteq I, \quad e_{X}(I) \subseteq I, \quad d_{X}^{-1}(I) \subseteq I
$$

This process translates a protocol into a collection of set constraints about the set $I$ of messages available to the intruder. Secrecy now becomes the question whether the set constraints, together with $s \notin I$, is satisfiable? Assuming a fixed number of principals, this is decidable in polynomial time for set constraints arising from Dolev-Yao's ping-pong protocols: we can compute an automaton accepting the minimal solution of the definite set constraint and check the membership of $s$.
There are several restrictions in the Dolev-Yao approach. In particular, only a fixed number of principals and, as mentioned above, only unary symbols may be used. A pairing function or a binary encryption symbol, allowing to write e.g. $e(k, m)$ instead of $e_{k}(m)$, i.e. allowing to consider keys as first-class objects, would considerably increase the expressive power. Such a model is presented below.

### 2.2. A more expressive model

We start from a model inspired by Paulson [26] and developed by Millen and Ruess in [23]. However, we do not use the trace model as in [23] or [26], but a new state-transition model similar to the MSR model proposed by Mitchell et al. [5] or those presented in [13]. Such models are much too expressive to be decidable, thus we consider in this paper a restricted model which does not allow nonce creation but on the other hand we add an arbitrary number of function symbols. In particular, we add compound keys and hashing. If a limited number of nonces is allowed for each pair of principals, nonce creation can be simulated beforehand, using additional binary function symbols $N_{1}, \ldots, N_{k}$ whose arguments are agent names.

In this section, it will be shown that this restricted model is still undecidable but not so far from decidability: Sections 4 and 5 develop a decidable class of set constraints which will be used as a tool to extract a decidable fragment (see Section 6.1) of the model described below.

### 2.2.1. Messages

They are built from a set of function symbols $\mathcal{F}$. Symbols of $\mathcal{F}$ are split into several sets: agent's names: we assume that $\mathcal{F}$ contains constants and function symbols which allow to built agent's names. We assume that the set of agent's names is infinite. Furthermore, we distinguish an infinite subset of honest agents $\mathcal{A}_{h}$.
invertible symbols which, intuitively, correspond to constructions whose components can be computed by an intruder. Typically, the pairing functions belong to this set of symbols since it is assumed that an intruder can retrieve each component $u, v$ from a pair $\langle u, v\rangle$. Such symbols can be applied to any term.
one way symbols which, intuitively, correspond to constructions whose components cannot be computed by an intruder. Typically, hash functions belong to this set. Such function symbols can be applied to any term. In addition, we assume that there are two special function symbols with one argument: $\operatorname{pub}()$ and $\operatorname{priv}()$. Intuitively, $\operatorname{pub}()$ and $\operatorname{priv}()$ return respectively a public and a private key when they are applied to agents names.
partially invertible symbols which intuitively correspond to constructions whose components can be computed by an intruder, subject to some knowledge of the intruder. More specifically, we will consider only one such function: encryption. (This is the only relevant example we can think of, but we could generalize to more symbols in this set). For such a binary function, which takes as argument a term $k$ (a key) and a term $u$ and whose application will be written $\{u\}_{k}$, the intruder can build $\{u\}_{k}$ when he knows $u$ and $k$ and can retrieve $u$ when he knows $\{u\}_{k}$ and the inverse key $k^{-1}$. A priori, the encryption function can be applied to any pair of terms so that we are not restricted to so-called "atomic keys". However, we will assume that the inverse of a key is the key itself, except for expressions $\operatorname{pub}(a)$ and $\operatorname{priv}(a)$ which are inverse of each other.
The set agent's names is denoted by $\mathcal{A \mathcal { G }}$, the set of invertible symbols by $\mathcal{I \mathcal { F }}$ and the set of one way symbols by $\mathcal{O} \mathcal{F}$. We get $\mathcal{F}=\mathcal{A G} \uplus \mathcal{I} \mathcal{F} \uplus \mathcal{O} \mathcal{F} \uplus\left\{\{ \}_{-}\right\}$.

Orthogonally, $\mathcal{F}$ is split into three sets of function symbols: those which are known publicly $\mathcal{P} \mathcal{F}$ (for instance $\left.\operatorname{pub}(),\left\langle_{-},{ }_{-}\right\rangle,\left\{_{-}\right\}_{-}\right)$, those which are cannot be used by the public $\mathcal{U} \mathcal{F}$, but only by specific agents (for instance a key construction function, which is known to a specific server only) and finally those which can be used by an intruder $\mathcal{A} \mathcal{F}$, only with
specific arguments. This last notion is the dual of partially invertible symbols. priv() is an example of such a symbol, which can be used by an intruder $i$, with the argument $i$ only. We will see later more examples. To summarize, the set of function symbols consists of

$$
\mathcal{F}=\mathcal{A G} \uplus \mathcal{I F} \uplus \mathcal{O F} \uplus\left\{\left\{_{-}\right\}_{-}\right\}=\mathcal{P F} \uplus \mathcal{U} \mathcal{F} \uplus \mathcal{A F},
$$

where $\operatorname{pub}() \in \mathcal{P F}$, $\operatorname{priv}() \in \mathcal{A F},\left\{_{-}\right\}_{-} \in \mathcal{P} \mathcal{F}$. For each partially constructible symbol in $\mathcal{A} \mathcal{F}$, it must be specified which of the arguments must be specific and which are unrestricted. The only argument of priv() is restricted.

Moreover, we assume a finite set of sorts containing in particular the sorts Agent, Ah, Ad, Message such that Ah and Ad are subsorts of Agent and Agent is a subsort of Message, the sort of all messages. In addition, the set of messages of sort Ah is exactly $\mathcal{A}_{h}$ and the set of messages of sort Ad is exactly $\mathcal{A G}-\mathcal{A}_{h}$. Elements of $\mathcal{A G}$ are constants or function symbols returning agent's names of sort Agent. The functions symbols pub(), priv(), \{_\}_take messages as argument and return messages: pub(), priv(), $\left\{_{-}\right\}_{-}$: Message $\times$Message $\rightarrow$ Message. The type of other symbols has to be specified with the protocol.

The set of messages is the set of (ground) terms $T(\mathcal{F})$ built over the above described signature and whose sort is Message.

As an example of an additional sort, we could consider nonces. Note however that, in our (un)decidability results, we will always assume that there is a bounded number of nonces; it is then possible to represent them as messages of the form e.g. $n_{i}(a, b)$ where $a, b$ are agents (in which case $n_{i} \in \mathcal{A} \mathcal{F} \cap \mathcal{O \mathcal { F }}$ and it is restricted in its first argument, meaning that only $a$ can generate $n_{i}(a, b)$, for any $b$ ).

Describing protocols and the behavior of honest participants requires variables ranging either over messages or over agents. Variables ranging over agents are usually called roles. Message schemes are terms of sort Message, built over $\mathcal{F}$ and possibly variables.

Example 2. We present here a protocol example (inspired by Kerberos), which will be used as a running example through the paper.

1. $A \rightarrow S: A, B$
2. $S \rightarrow A:\left\{\left\langle B, K(A, B),\{\langle A, K(A, B)\rangle\}_{\operatorname{shr}(B)}\right\rangle\right\}_{\operatorname{shr}(A)}$
3. $A \rightarrow B:\left\langle\{m(A, B)\}_{K(A, B)},\{\langle A, K(A, B)\rangle\}_{\operatorname{shr}(B)}\right\rangle$
4. $B \rightarrow A:\{h(m(A, B))\}_{K(A, B)}$.

In words, $A$ tells the key server $S$ that she wants to securely communicate with $B$. Then $S$ sends back to $A$ a message, encrypted using a key that she shares with the server and containing a session key $K(A, B)$ together with a certificate which can be opened by $B$ only. At the third step, $A$ sends her message $m(A, B)$, encrypted using the key $K(A, B)$, together with the certificate, which is copied blindly from message 2 . Finally, $B$ acknowledges the reception, sending back a digest $h(m(A, B))$ of the previous message, encrypted using the shared key.

We are going to see in more detail how this protocol is formally described in the model. For the moment, let us only make precise the components of the signature.

We assume here six sorts: Nat, Agent, Ah, Ad, Message, Key. The last sort is protocol specific. Introducing such a sort means that the agents are assumed to be able to see whether a message is a key or not (we will discuss this hypothesis later on).

There is a specific constant $s$ (the server) of sort Agent. The way other agent's names are built is irrelevant. We could, for instance, use natural numbers together with a label for (dis)honest participants: $0: \rightarrow$ Nat, succ : Nat $\rightarrow$ Nat, ha: Nat $\rightarrow$ Ah, da $:$ Nat $\rightarrow$ Ad. For simplicity, in what follows, we will use the notation $a_{1}, a_{2}, \ldots$ for honest agents (i.e agent of sort Ah ) and $i_{1}, i_{2}, \ldots$ for dishonest agents (i.e. the other agents). Note that the set of agents is infinite.
Then, we use $\left.\mathcal{I F}=\left\{\left\langle_{-},\right\rangle_{,}\right\rangle\left\langle_{-},{ }_{-},{ }_{-}\right\rangle\right\}$. These tupling functions take arbitrary messages as arguments and return messages.

$$
\mathcal{O F}=\{h, m, K, \text { shr, succ, } 0, h a, d a\}
$$

with $h:$ Message $\rightarrow$ Message, $m:$ Agent $\times$ Agent $\rightarrow$ Message, $K:$ Agent $\times$ Agent $\rightarrow$ Key, shr: Agent $\rightarrow$ Key.

Now, the following are public symbols:

$$
\mathcal{P} \mathcal{F}=\left\{\left\langle_{-},{ }_{-}\right\rangle,\left\langle_{-},{ }_{-},{ }_{-}\right\rangle, \text {pub, }\left\{_{-}\right\}_{-}, h, 0, \text { succ, } d a, h a\right\}
$$

In particular, anybody can know every agent name and every agents public key. Now, $K$ can only be used by the server

$$
\mathcal{U F}=\{K\}
$$

Finally,

$$
\mathcal{A F}=\{m, \mathrm{prv}, \mathrm{shr}\}
$$

where both symbols are restricted in their first argument.

### 2.2.2. Events and global states

There are two kinds of events: message and state events. A state event is of the form $Q=S(\mathrm{~A}, n, X)$ where $S$ is taken in a finite set $\mathcal{F}_{s}$ of function symbols. Typically, $\mathcal{F}_{s}=$ \{Init, Resp, Serv\}. Usually, for state events of the form $\operatorname{Serv}(A, n, X), A$ is always equal to $s$ the constant representing the server. A is a ground term of sort Agent, $n$ is a natural number that represents the step of the protocol, and $X=\operatorname{Mem}(Q)$ is a tuple of messages representing the memory held by the state. A state scheme is built in the same way, except that the agent can be abstracted (using a role) and the messages are replaced with message schemes.

A global state is a set (not a multiset) of events. The content of a global state is its set of messages, written as

$$
\text { Cont }(H) \stackrel{\text { def }}{=} H \cap \text { Messages. }
$$

Example 3 (Example 2 continued). The messages $i_{1}$ or $m\left(i_{1}, a_{2}\right)$ can be built from the formalism described in our running example. Init $\left(i_{1}, 1,\left\langle i_{1}, a_{1}, s\right\rangle\right)$ is a state event. Intuitively, it represents the dishonest agent $i_{1}$ ready to start a session as initiator.

### 2.2.3. Inductive relations

Given a term $t=f\left(t_{1}, \ldots, t_{n}\right)$, $\operatorname{parts}(t)$ is defined inductively as follows:

- if $f \in \mathcal{O} \mathcal{F} \cup \mathcal{A G}$, then parts $(t) \stackrel{\text { def }}{=}\{t\}$,
- if $f \in \mathcal{I F}$, then $\operatorname{parts}(t) \stackrel{\text { def }}{=}\{t\} \cup \bigcup_{i=1}^{n} \operatorname{parts}\left(t_{i}\right)$,
- if $f=\{ \}_{-}$, then parts $\left.\left(\left\{t_{1}\right\}_{t_{2}}\right\}\right) \stackrel{\text { def }}{=}\{t\} \cup \operatorname{parts}\left(t_{1}\right)$.

Given a set of terms $S$, parts $(S)$ is the set of parts of all terms in $S$.
analz $(S)$ is the subset of parts $(S)$ consisting of only those subterms that are accessible to an attacker: analz $(S)$ is the least set $S^{\prime}$ containing $S$ and such that:

- if $f\left(t_{1}, \ldots, t_{n}\right) \in S^{\prime}$ and $f \in \mathcal{I F}$, then $t_{1}, \ldots, t_{n} \in S^{\prime}$,
- if $\left\{t_{1}\right\}_{t_{2}} \in S^{\prime}$ and $t_{2}^{-1} \in S^{\prime}$, then $t_{1} \in S^{\prime}$.

Conversely, an attacker may use any available function to build new messages. synth $(S)$ is the least set of messages $S^{\prime}$ containing $S$ and such that

- If $f \in \mathcal{P} \mathcal{F}$ and $t_{1}, \ldots, t_{n} \in S^{\prime}$, then $f\left(t_{1}, \ldots, t_{n}\right) \in S^{\prime}$
- If $f \in \mathcal{A} \mathcal{F}, f$ is restricted w.r.t. its arguments $j_{1}, \ldots, j_{k}, t_{1}, \ldots, t_{n} \in S^{\prime}$, and $t_{j_{1}}, \ldots, t_{j_{k}}$ $\in \mathcal{A}_{d}$, then $f\left(t_{1}, \ldots, t_{n}\right) \in S^{\prime}$, where $\mathcal{A}_{d}$ is the set of dishonest agents $\left(\mathcal{A}_{d}=\left\{i_{n} \mid n \in\right.\right.$ $\mathbb{N}\}$ ).
The intruder in our model synthesizes faked messages from analyzable parts of a set of available terms and he can iterate the process. This motivates the following definition: fake $(S)$ is the least set $S^{\prime}$ containing $S$ and such that synth $\left(S^{\prime}\right) \subseteq S^{\prime}$ and analz $\left(S^{\prime}\right) \subseteq S^{\prime}$. Note that fake $(S)$ is not necessarily equal to $\operatorname{synth}(\operatorname{analz}(S))$ if we do not assume atomic keys: for instance if an intruder knows $t_{1},\{t\}_{\left\langle t_{1}, t_{2}\right\rangle}, t_{2}$, he can build $t$ by first constructing $\left\langle t_{1}, t_{2}\right\rangle$ and then decrypt the message.

Example 4 (Example 3 continued). Assume that (in some state), the intruder holds the following messages:

$$
\begin{aligned}
S_{1}= & \left\{\left\{\left\langle m\left(a_{1}, i_{1}\right), a_{2}\right\rangle\right\}_{h\left(K\left(a_{1}, a_{2}\right)\right)},\left\{h\left(K\left(a_{1}, a_{2}\right)\right)\right\}_{\operatorname{pub}\left(i_{1}\right)},\right. \\
& \left.\left\{m\left(a_{1}, a_{2}\right)\right\}_{K\left(a_{1}, a_{2}\right)}, \operatorname{pub}\left(i_{1}\right)\right\} .
\end{aligned}
$$

Then analz $\left(S_{1}\right)$ contains for instance $h\left(K\left(a_{1}, a_{2}\right)\right), m\left(a_{1}, i_{1}\right)$ but not $m\left(a_{1}, a_{2}\right)$. fake $\left(S_{1}\right)$ contains for instance $\left\{\left\langle a_{2}, h\left(K\left(a_{1}, a_{2}\right)\right)\right\rangle\right\}_{h\left(m\left(a_{1}, i_{1}\right)\right)}$.

### 2.2.4. Protocols

A protocol transition $t$ is of the form $\operatorname{Pre}(t) \longrightarrow \operatorname{Post}(t)$, where $\operatorname{Pre}(t)$ and $\operatorname{Post}(t)$ are (finite) sets of messages and states. Unlike in [13], there is not any new spell: the secrecy policy may be specified independently as presented later. Such transitions specify a possible global state change in a way to be explained below. A transition $t$ shows a state change for one agent. Formally, $\operatorname{Pre}(t)$ and $\operatorname{Post}(t)$ contain at most one state event and $\operatorname{Pre}(t)$ contains one state event if and only if $\operatorname{Post}(t)$ contains one state event.

A protocol is simply a set of protocol transitions, an initial global state $H_{0}$ and a secrecy specification $S_{0}$. When $H_{0}$ is not specified, it is assumed that $H_{0}=\emptyset$. Both the protocol transitions and the secrecy specification is infinite. They are however represented by means of instances of a finite number of terms: typically, the protocol is given by a finite set of rules $u_{i} \rightarrow v_{i}$ where $u_{i}$ and $v_{i}$ are finite sets of message schemes and state schemes. Such

$$
\begin{align*}
\emptyset & \rightarrow\left\{\begin{array}{l}
\operatorname{lnit}(A, 1,\langle A, B, s\rangle), \\
\operatorname{Resp}(B, 1,\langle B, s\rangle) \\
\operatorname{Serv}(s, 1, s)
\end{array}\right\}  \tag{0}\\
\{\operatorname{lnit}(A, 1,\langle A, B, s\rangle)\} & \rightarrow\left\{\begin{array}{l}
\operatorname{Init}(A, 2,\langle A, B, s\rangle),\langle A, B\rangle\}
\end{array}\right.  \tag{1}\\
\left\{\begin{array}{l}
\operatorname{Serv}(s, 1, s) \\
\langle A, B\rangle
\end{array}\right\} & \rightarrow\left\{\begin{array}{l}
\operatorname{Serv}(s, 2, s), \\
\left\{B, K(A, B),\{A, K(A, B)\}_{\operatorname{Shr}(B)}\right\}_{\operatorname{shr}}(A)
\end{array}\right\}  \tag{2}\\
\left\{\begin{array}{l}
\operatorname{Init}(A, 2,\langle A, B, s\rangle), \\
\{B, X, Y\}_{\operatorname{shr}(A)}
\end{array}\right\} & \rightarrow\left\{\begin{array}{l}
\operatorname{Init}(A, 3,\langle A, B, s, X, m(A, B)\rangle), \\
\left\langle\{m(A, B)\}_{X}, Y\right\rangle
\end{array}\right\}  \tag{3}\\
\left\{\begin{array}{l}
\operatorname{Resp}(B, 1,\langle B, s\rangle), \\
\left\langle\{Z\}_{X},\{A, X\}_{\operatorname{shr}(B)}\right\rangle
\end{array}\right\} & \rightarrow\left\{\begin{array}{l}
\{\operatorname{Resp}(B, 1,\langle B, s, A, Z, X\rangle), \\
\left.\{h(Z)\}_{X}\right\}
\end{array}\right\} \tag{4}
\end{align*}
$$

Fig. 1. Rules of the protocol.
rules represent the infinite set $u_{i} \sigma \rightarrow v_{i} \sigma$ where $\sigma$ is any substitution compatible with the types.

The secrecy policy $S_{0}$ is given by an finite union of sets of the form

$$
\left\{t_{1}, \ldots, t_{n} \mid x_{1}, \ldots, x_{k} \in \mathcal{A}_{h}\right\}
$$

where $x_{1}, \ldots, x_{k}$ are the free variables of the message schemes $t_{1}, \ldots, t_{n}$.
$S_{0}$ represents the set of messages that the intruder should not hold.
Example 5 (Example 4 continued). The protocol, as described in Example 2 is a bit sloppy. We used there the standard notations, but, if we want to be more precise, we have to specify for instance in message 3 how Alice retrieves the different components of the message she sends. Typically in such protocols, $A, B$ are roles, not agent's names. The " $B$ " in message 3 can be either the name sent in message 1 or the name passed in message 2 (It does not make a difference in this particular example. But it does make a difference in other situations, as shown by the attack on the Needham-Schroeder protocol [22]).

This protocol should not reveal the messages $m(a, b), K(a, b), \operatorname{shr}(a)$ when $a$ and $b$ are honest agents. This can be expressed by the following secrecy policy $S_{0}$ :

$$
S_{0}=\left\{m(a, b), K(a, b) \mid a, b \in \mathcal{A}_{h}\right\} \cup\left\{\operatorname{shr}(a) \mid a \in \mathcal{A}_{h}\right\} .
$$

Next, the protocol rules are given in Fig. 1. The rule 0 says that, at any time, a new session can be started (the precondition is an empty set). After applying this rule to an instance $a, b$, the agents $a, b, s$ are ready to act as participants of a protocol session.

The rule 1 corresponds to the first step of the protocol: any agent $a$ who is ready to act as $A$ in the protocol can send the message $\langle a, b\rangle$ to $s$ and switch to a state in which she remembers having completed the first step (hence the second argument is 2) and having sent the message $\langle a, b\rangle$ to $s$.

Rule 2 corresponds to the second step of the protocol: if $s$ is ready to serve a key and if the message $\langle a, b\rangle$ has been sent, then the server switches, generates the key $K(a, b)$ and sends the expected message. Note that the variables $A, B$ are local to the rule, hence the instances are not necessarily the same as in the previous step: an intruder can very well perform the first step of the protocol, in which case there are two $\langle a, b\rangle,\left\langle a^{\prime}, b^{\prime}\right\rangle$ in the global state and the second instance may be used instead of the first one.

In the rule 3 , the agent $a$, who completed the first steps of the protocol expects a message of the form $\{b, \alpha, \beta\}_{\operatorname{shr}(a)}$. She can check that the message is an encrypted message containing three components and that the first component is an agent's name, with whom she started a session. However, she cannot check that the second component is indeed $K(a, b)$ and, similarly, she cannot open the third component (the ticket). Hence these two components are left as local variables of the rules which can be instantiated in an arbitrary way, provided that $Y$ gets a term of sort Message and $X$ gets a term of sort Key. (We assume here that $a$ is able to recognize whether a term has type Key or not.)

Similarly, in the last rule, the expected instance of $Z$ is $m(a, b)$, but it could be any faked message: there is no way to check this.

This formal specification of the protocol gives more precision on the abilities of each agent. We make precise here what is expected by each participant and what is his behavior.

### 2.2.5. Global state transitions

Given a protocol $P$ and a set of initial knowledge $I$ (of the intruder), the global succession relation transforms a state $H$ to a new state $H^{\prime}$. A succession is either honest, i.e. it corresponds to an action by an agent following the protocol, or it is faked by the intruder.

- $H^{\prime}$ is an honest successor of $H$, denoted by honest $(P)\left(H, H^{\prime}\right)$, if there exists an applicable transition $t$ in $P$ such that $H^{\prime}=(H \backslash(\operatorname{Pre}(t) \cap$ States $)) \cup \operatorname{Post}(t)$.
- $H^{\prime}$ is a fake successor of $H$, denoted by fake $(I)\left(H, H^{\prime}\right)$, if there exists a field $X \in$ fake $(\operatorname{Cont}(H) \cup I)$ such that $H^{\prime}=H \cup\{X\}$.
In the honest case, a transition $t$ is applicable in $H$ if $\operatorname{Pre}(t) \subseteq H$. In the fake case, the intruder is restricted to adding only messages that can be inferred from the content of the current state and the initial knowledge. In either case, we write $\operatorname{global}(P, I)\left(H, H^{\prime}\right)$. This relation determines a logical transition system with the initial global state $H_{0}$ as its initial state. The set of reachable states of this transition system is denoted by reachable $(P, I)$.


### 2.2.6. Secrecy policy

Given the intruder's initial knowledge $I$ and a secrecy policy $S_{0}$, a global state $H$ is called $I, S_{0}$-secure if fake $(\operatorname{Cont}(H) \cup I) \cap S_{0}=\emptyset$; these states are collected in the set secure $\left(I, S_{0}\right)$. Now, a protocol $P$ is called secure if $\operatorname{secure}\left(I, S_{0}\right)$ is an invariant of the transition relation associated with $P$ and $S_{0}$ is the secrecy policy associated to $P$; i.e. for all $I$, reachable $(P, I)$ is a subset of $\operatorname{secure}\left(I, S_{0}\right)$.
Remark: actually, it is sufficient to prove that $\operatorname{secure}\left(I_{0}, S_{0}\right)$ is an invariant for $I_{0}$ the maximal set compatible with $S_{0}$ :

$$
I_{0}=\left\{m \mid \operatorname{parts}(m) \cap S_{0}=\emptyset\right\} .
$$

This definition is slightly different from the one given in [13] but it matches more precisely the idea of secrecy while the definition given in [13] was an over-approximation of secrecy in order to allow inductive proofs.

### 2.2.7. An undecidability result

We present now an undecidability result. Let us emphasize that we do not have here the nonce construction. Hence, the result is stronger than the undecidability result of [18].

Theorem 6. It is undecidable whether or not a protocol $P$ is secure.
A proof of this result has been proposed by Even and Goldreich in [19] by reducing the Post Correspondence Problem (PCP) to secrecy. A simplified proof was proposed by Rusinowitch. The intuitive idea is the following one: consider a finite alphabet $\Sigma$ and an instance of PCP: $\left(u_{i}, v_{i}\right)_{1 \leqslant i \leqslant n}, u_{i}, v_{i} \in \Sigma^{*}$. We construct the following protocol:

$$
\begin{aligned}
A & \rightarrow B:\{\langle 0,0\rangle\}_{K_{a b}} \\
B & \rightarrow A:\left\{\left\langle N_{1}, N_{2}\right\rangle\right\}_{K_{a b}} \\
A:\{\langle x, y\rangle\}_{K_{a b}} & \rightarrow B:\left\{\left\langle x u_{i}, y v_{i}\right\rangle\right\}_{K_{a b}},\{s\}_{\left\{\left\langle x u_{i}, x u_{i}\right\rangle\right\}_{K_{a b}}}, \quad 1 \leqslant i \leqslant n .
\end{aligned}
$$

The key $K_{a b}$ is a symmetric, private key between $A$ an $B$. The last rule describes $n$ rules for the agent $A$. The left-hand side describes the message expected by $A$. One can show that $s$ remains secret if and only if there is no solution to the considered instance of PCP. A similar protocol can be build without using composed keys.

An inconvenience to both constructions is that for each instance of PCP with no solution, the corresponding protocol does not have one honest instance. Using Petri nets, we construct in [9] a reduction such that the corresponding protocol is a "real" protocol in the sense that each rule of the protocol can be played in the given order: the first rule, then the second and so on, i.e., there is at least an honest instance of the protocol. In addition this reduction only uses standard cryptographic primitives, namely pairing and encryption with symmetric keys and a fixed number of roles (actually only one role) and a finite number of protocol rules. For each reduction (using PCP or Petri nets), the intruder actually may only forwards messages and does not need to forge new ones.

## 3. Definite set constraints

### 3.1. Definite set constraints and intersection constraints

This class of set constraints has been introduced in [21] and studied by various authors (e.g. [7]). Each constraint is a conjunction of inclusions $e_{1} \subseteq e_{2}$ where $e_{1}$ is a set expression and $e_{2}$ is a term set expression. Term set expressions are built out of a fixed ranked alphabet of function symbols $\mathcal{F}$, the symbols $\top, \perp$ and set variables. A set expression is either a term set expression or a union of two set expressions $e_{1} \cup e_{2}$, or an intersection of two set expressions $e_{1} \cap e_{2}$ or the image of set expressions by some function symbol $f\left(e_{1}, \ldots, e_{n}\right)$ or a projection $f_{i}^{-1}\left(e_{1}\right)$ where $f$ is a function symbol and $i \in[1 . . n]$ if $n$ is the rank of $f$. Note that negation is not allowed.

Example 7. Here is a definite set constraint:

$$
\begin{aligned}
f_{2}^{-1}(X) & \subseteq g(Y), & f(f(X, Y) \cap X, X) & \subseteq X \\
g(Y) \cap Y & \subseteq X, & a & \subseteq Y
\end{aligned}
$$

where $a, f, g$ are function symbols and $X, Y$ are set variables.

Set expressions denote sets of subsets of the Herbrand universe $T(\mathcal{F})$; if $\sigma$ assigns each variable to some subset of $T(\mathcal{F})$, then $\llbracket \rrbracket_{\sigma}$ is defined by

$$
\begin{aligned}
& \llbracket X \rrbracket_{\sigma} \stackrel{\text { def }}{=} X \sigma \\
& \llbracket f\left(e_{1}, \ldots, e_{n}\right) \rrbracket_{\sigma} \stackrel{\text { def }}{=}\left\{f\left(t_{1}, \ldots, t_{n}\right) \mid \forall i \in[1 . . n], t_{i} \in \llbracket e_{i} \rrbracket_{\sigma}\right\} \\
& \llbracket e_{1} \cap e_{2} \rrbracket_{\sigma} \stackrel{\text { def }}{=} \llbracket e_{1} \rrbracket_{\sigma} \cap \llbracket e_{2} \rrbracket_{\sigma} \\
& \llbracket f_{i}^{-1}(e) \rrbracket_{\sigma} \stackrel{\text { def }}{=}\left\{t_{i} \mid \exists t_{1}, \ldots, t_{n} . f\left(t_{1}, \ldots, t_{n}\right) \in \llbracket e \rrbracket_{\sigma}\right\} \\
& \llbracket \top \rrbracket_{\sigma} \stackrel{\text { def }}{=} T(\mathcal{F}) \\
& \llbracket \perp \rrbracket_{\sigma} \stackrel{\text { def }}{=} \emptyset \\
& \llbracket e_{1} \cup e_{2} \rrbracket_{\sigma} \stackrel{\text { def }}{=} \llbracket e_{1} \rrbracket_{\sigma} \cup \llbracket e_{2} \rrbracket_{\sigma},
\end{aligned}
$$

$\sigma$ satisfies $e_{1} \subseteq e_{2}$ iff, $\llbracket e_{1} \rrbracket_{\sigma} \subseteq \llbracket e_{2} \rrbracket_{\sigma}$. This extends to conjunctions of inclusions.
Example 8 (Example 7 continued). The substitution $\sigma \stackrel{\text { def }}{=}[X \rightarrow \emptyset, Y \rightarrow\{a\}]$ satisfies the set constraints described in Example 7.

Following a standard translation (see e.g. [7]), the definite set constraints can be rewritten (in polynomial time) into intersection constraints which are conjunction of inclusions of one of the forms:

$$
\begin{array}{ll}
X \subseteq Y, & f\left(X_{1}, \ldots, X_{n}\right) \subseteq X \\
X \subseteq f\left(Y_{1}, \ldots, Y_{m}\right), & f\left(X_{1}, \ldots, X_{n}\right) \subseteq g\left(Y_{1}, \ldots, Y_{m}\right)
\end{array}
$$

where $X, X_{1}, \ldots, X_{n}, Y, Y_{1}, \ldots, Y_{m}$ are intersections of set variables. In other words, the constraints can be flattened and union and projections eliminated thanks (in particular) to the equivalence:

$$
f_{i}^{-1}(X) \subseteq Y \Leftrightarrow X \cap f(\mathrm{~T}, \ldots, \mathrm{~T}) \subseteq f(\mathrm{~T}, \ldots, Y, \mathrm{\top}, \ldots)
$$

where the $Y$ is in $i$ th position and further transformations introducing new variables.
The translation $\tau$ from definite set constraints to intersection constraints may require the introduction of new variables. Formally, $\tau$ preserves the solutions:

Lemma 9. $\sigma$ is a solution of the definite set constraint C if and only if there exists $\sigma^{\prime}$, solution of the intersection constraint $\tau(C)$, such that $\sigma$ is the restriction of $\sigma^{\prime}$ to the variables of $C$.

Theorem 10 (Charatonik and Podelski [7]). The satisfiability of intersection constraints (resp. definite set constraints) is DEXPTIME-complete and each satisfiable constraint has a least solution which is accepted by a finite tree automaton.

Moreover, the decision procedure provides effectively the finite tree automaton accepting the least solution.

### 3.2. Intersection constraints with nonemptiness guards

Now, we consider a slight extension of intersection constraints, yielding a result similar to Theorem 10. If $e$ is a set expression, let nonempty $(e)$ be a statement which is satisfied by $\sigma$ iff $\llbracket e \rrbracket_{\sigma}$ is not empty.

We extend the formalism as follows. A flat expression is either an intersection of set variables or a set expression $f\left(X_{1}, \ldots, X_{n}\right)$ where $X_{1}, \ldots, X_{n}$ are intersections of set variables. An intersection constraint with non-emptiness guards is a conjunction of clauses

$$
\operatorname{nonempty}\left(e_{1}^{\prime}\right), \ldots, \operatorname{nonempty}\left(e_{n}^{\prime}\right) \Rightarrow e_{1} \subseteq e_{2}
$$

where $e_{1}^{\prime}, \ldots, e_{n}^{\prime}, e_{1}, e_{2}$ are flat expressions.
The interpretation of such constraints is the expected one. Note that, of course, they extend intersection constraints. However, the algorithm given in [7] can be applied with slight changes only.

In other words, enriching the intersection constraints with clauses of the above form, we still have the same result as in Theorem 10, as a corollary of [7]:

Theorem 11. The satisfiability of intersection constraints with nonemptiness guards is DEXPTIME-complete and each satisfiable constraint has a least solution which is effectively accepted by a finite tree automaton.

Proof. If we want to be as self-contained as possible, we need to reproduce, at least partly, the proof of [7]. In the next section, we will also rely on this proof.

First, we can assume that all expressions occurring in the guards also occur as members of inclusions (if necessary, add $e \subseteq e$ and flatten again).

Now, according to [7], we saturate the constraints using the inference rules given in Fig. 2.
In this figure, $X, X_{1}, \ldots, X_{n}, X^{\prime}, X_{1}^{\prime}, \ldots, X_{n}^{\prime}$ are intersection of set variables and $e, e^{\prime}$, $e_{1}, e_{2}, e_{1}^{\prime}, e_{2}^{\prime}$ are any flat set expressions. If these rules are applied to intersection constraints, we get as conclusions intersection constraints again, with the convention that expressions $f\left(e_{1}, \ldots, e_{n}\right) \cap f\left(e_{1}^{\prime}, \ldots, e_{n}^{\prime}\right)$ are eagerly normalized into $f\left(e_{1} \cap e_{1}^{\prime}, \ldots, e_{n} \cap e_{n}^{\prime}\right)$.

As shown in [7] the rules of Fig. 2 are correct and applying the inference rules saturates the set constraint in deterministic exponential time (assuming that reflexivity and weakening do not introduce new variables).

For every constraint $\gamma$, we let $\gamma^{C}$ be the saturated set. As in [7] again, let $\gamma^{S}$ be the solved form of $\gamma$ :

$$
\gamma^{S}=\left\{f\left(e_{1}, \ldots, e_{n}\right) \subseteq X \in \gamma^{C} \mid \operatorname{nonempty}\left(f\left(e_{1}, \ldots, e_{n}\right)\right) \in \gamma^{C}\right\}
$$

where $X$ is a variable.
$\gamma^{S}$ is essentially the definition of a tree automaton whose states are set variables. Let $\sigma$ be the substitution, assigning to each variable $X$, the language recognized by this tree automaton in state $X$. We are going to prove that either false $\in \gamma^{C}$ or else $\sigma$ is the least solution of $\gamma$. The minimality of $\sigma$ comes from automata theory. Let us concentrate on the fact that $\sigma$ is a solution of $\gamma$.

The proof that $\sigma$ satisfies all inclusions $e_{1} \subseteq e_{2}$ in $\phi^{C}$ is identical to [7]. Consider a clause nonempty $\left(e_{1}\right), \ldots, \operatorname{nonempty}\left(e_{n}\right) \Rightarrow e \subseteq e^{\prime}$ with $n \geqslant 1$. If there is an $i$ such that $\llbracket e_{i} \rrbracket_{\sigma}$ is


Fig. 2. Inference rules for intersection constraints.
empty, then the clause is trivially satisfied. Otherwise, we may assume that every $e_{i}$ is an intersection variable since $\sigma$ satisfies nonempty $\left(f\left(Y_{1}, \ldots, Y_{n}\right)\right)$ if and only if it satisfies nonempty $\left(Y_{1}\right), \ldots$, nonempty $\left(Y_{n}\right)$. Then let $e_{i}$ be the intersection $X_{i}^{1} \cap \ldots \cap X_{i}^{k_{i}}$. For every $i$, there is a term $t_{i}$ which is accepted by the tree automaton in every state $X_{i}^{j}$. We prove below that, if $t$ is accepted in all states $X_{1}, \ldots, X_{n}$, then nonempty $\left(X_{1} \cap \ldots \cap X_{n}\right) \in \gamma^{C}$. Let us assume this for the moment. Then, by the rules Cut and Propagation 2, $e \subseteq e^{\prime} \in \gamma^{C}$, which proves that $\sigma$ satisfies $e \subseteq e^{\prime}$, thanks to [7].

We prove by induction on the size of $t$ that, if $t$ is accepted in all states $X_{1}, \ldots, X_{n}$, then nonempty $\left(X_{1} \cap \ldots X_{n}\right) \in \gamma^{C}$.

- If $t$ is a constant, by definition of the automaton, $t \subseteq X_{i} \in \gamma^{C}$ for every $i$. Then, by Compatibility (applied $n-1$ times), $t \subseteq X_{1} \cap \ldots \cap X_{n} \in \gamma^{C}$
- If $t=f\left(t_{1}, \ldots, t_{m}\right)$. By definition of the automaton, there are inclusions $f\left(e_{1}^{i}, \ldots, e_{m}^{i}\right) \subseteq$ $X_{i} \in \phi^{C}$ such that, for every $j \in[1 . . m]$, for every $i, t_{j} \in \llbracket e_{j}^{i} \rrbracket_{\sigma}$. Now, we apply the
induction hypothesis: for every $j$, nonempty $\left(e_{j}^{1} \cap \ldots \cap e_{j}^{n}\right) \in \gamma^{C}$. By Propagation 2, nonempty $\left(f\left(e_{1}^{1} \cap \ldots \cap e_{1}^{n}, \ldots, e_{m}^{1} \cap \ldots \cap e_{m}^{n}\right)\right) \in \gamma^{C}$. Now, by Compatibility,

$$
\left.f\left(e_{1}^{1} \cap \ldots \cap e_{1}^{n}, \ldots, e_{m}^{1} \cap \ldots \cap e_{m}^{n}\right)\right) \subseteq X_{1} \cap \ldots \cap X_{n} \in \gamma^{C}
$$

and, by Propagation 1, we conclude that nonempty $\left(X_{1} \cap \ldots \cap X_{n}\right) \in \gamma^{C}$.
To summarize: the assignment $\sigma$ defined by the solved form $\gamma^{S}$ also satisfies the conditional inclusions of $\gamma^{C}$, which means that $\gamma^{C}$ is satisfiable whenever false $\notin \gamma^{C}$ and $\sigma$ is then the minimal solution of $\gamma$.

In the proof of the last result, we have seen in passing that nonempty $(X)$ is a logical consequence of the constraint iff it belongs to the saturated set. It follows that:

Corollary 12. Deciding whether the minimal solution of a definite set constraint with nonemptiness guards assigns the empty set to $X$ is DEXPTIME-complete.

Actually, the DEXPTIME-hardness of this corollary is missing so far. But we can reduce the non-emptiness problem of the intersection of $n$ tree automata (which is DEXPTIMEcomplete) by translating the definitions of the automata into intersection constraints and adding a clause

$$
\operatorname{nonempty}\left(X_{1} \cap \ldots \cap X_{n}\right) \Rightarrow a \subseteq X,
$$

where $X_{1}, \ldots, X_{n}$ are the set variables corresponding to the accepting states of the $n$ automata, respectively, $X$ is a new variable and $a$ is a constant.

## 4. Tree automata with one memory

The idea is to enrich the expressiveness of tree automata by allowing them to carry and test some information. For instance, a pushdown automaton will keep a stack in its memory and check the symbols at the top of the stack. What we do here is something similar. Our automata work on trees instead of words and may perform more general constructions and more general tests. We will see later as an example how to express pushdown automata in our formalism.
Informally, a tree automaton with one memory computes bottom-up on a tree $t$ by synthesizing both a state (in a finite set of states $Q$ ) and a memory, which is a tree over some alphabet $\Gamma$. Each transition uses some particular function which computes the new memory from the memories at each direct son. Each transition may also check for equalities the contents of the memories at each son.
Given an alphabet of function symbols $\Gamma$, the set of functions $\Phi$ which we consider here (and which may be used to compute on memories) is the least set of functions over $T(\Gamma)$ which is closed by composition and containing:

- for every $f \in \Gamma$ of arity $n$, the function $\lambda x_{1}, \ldots x_{n} . f\left(x_{1}, \ldots, x_{n}\right)$,
- for every $n$ and every $1 \leqslant i \leqslant n$, the function $\lambda x_{1}, \ldots, x_{n} . x_{i}$,
- for every $f \in \Gamma$ of arity $n$ and for every $1 \leqslant i \leqslant n$, the (partial) function which associates each term $f\left(t_{1}, \ldots, t_{n}\right)$ with $t_{i}$, which we write $\lambda f(\vec{x}) \cdot x_{i}$.

For instance, if $\Gamma$ contains a constant (empty stack) and unary function symbols, $\Phi$ is the set of functions which push or pop after checking the top of the stack.

Definition 13. A tree automaton with one memory is a tuple $\left(\mathcal{F}, \Gamma, Q, Q_{f}, \Delta\right)$ where $\mathcal{F}$ is an alphabet of input function symbols, $\Gamma$ is an alphabet of memory function symbols, $Q$ is a finite set of states, $Q_{f}$ is a subset of final states, $\Delta$ is a finite set of transition relations of the form $f\left(q_{1}, \ldots, q_{n}\right) \xrightarrow[F]{c} q$ where

- $f \in \mathcal{F}$ is called the head symbol of the rule,
- $c$ is a subset of $\{1, \ldots, n\}^{2}$, defining an equivalence relation on $\{1, \ldots, n\}$.
- $F \in \Phi$ such that $F$ takes $k$ arguments where $k$ is the number of equivalence classes w.r.t. $c$
- $q_{1}, \ldots, q_{n}, q \in Q,(q$ is the target of the rule $)$.

A configuration of the automaton consists of a state and a term in $T(\Gamma)$ (the memory). Then computations work as follows: if $t=f\left(t_{1}, \ldots, t_{n}\right)$ and the computation on $t_{1}, \ldots, t_{n}$, respectively yields the configurations $q_{1}, \tau_{1}, \ldots, q_{n}, \tau_{n}$, then the automaton, reading $t$, may move to $q, \tau$ when there is a transition rule $f\left(q_{1}, \ldots, q_{n}\right) \xrightarrow[F]{c} q$ and for every $i=j \in c$, $\tau_{i}=\tau_{j}$ and $\tau=F\left(\tau_{i_{1}}, \ldots, \tau_{i_{k}}\right)$, where $i_{1}, \ldots, i_{k}$ are any representatives of the equivalence classes for $c$ (the way $i_{j}$ is chosen in its equivalence class is not relevant). A tree $t$ is accepted by the automaton whenever there is a computation of the automaton on $t$ yielding a configuration $q, \gamma$ with $q \in Q_{f}$.

Example 14. Assume that the transitions of the automaton $A$ are (other components of the automaton are obvious from the context, $T$ is the identity relation):

$$
\begin{aligned}
& g(q) \xrightarrow[\lambda x_{1} \cdot x_{1}]{\mathrm{T}} \quad q \quad f\left(q_{a}, q_{a}\right) \xrightarrow[\substack{ \\
\lambda x_{1} . h\left(x_{1}\right)}]{1=2} q \\
& g\left(q_{a}\right) \xrightarrow[\lambda x_{1} \cdot h\left(x_{1}\right)]{\top} q \quad f(q, q) \xrightarrow[\lambda h\left(x_{1}\right) \cdot x_{1}]{\substack{\mathrm{T}}} q \\
& a \xrightarrow[b]{\mathrm{T}} \quad q_{a}
\end{aligned}
$$

A computation of the automaton on $f(g(f(a, a)), g(a))$ is displayed in Fig. 3, in which the configurations reached at each node are displayed in a frame.

Pushdown automata (on words) perform transitions $a, q, \alpha \cdot \gamma \rightarrow q^{\prime}, \beta \cdot \gamma$ where $a$ is an input symbol, $q, q^{\prime}$ are states and $\alpha, \beta, \gamma$ are words over the stack alphabet (the rule pops $\alpha$ and pushes $\beta$ ). Such a rule can be translated in the above formalism, viewing letters as unary symbols: $a(q) \xrightarrow[\lambda x . \beta \alpha_{-1}^{-1} x]{ } q^{\prime}$. If $w=a_{1}\left(\ldots a_{k}\left(\_\right) \ldots\right)$, we use here the notation $w^{-1}(x)$ for $a_{k}^{-1}\left(\ldots\left(a_{1}^{-1}(x)\right)\right)$, the additional subscript 1 being implicit for each letter, which has only one argument.

This translation does not make use of equality tests. Orthogonally, it is possible to simulate tree automata with equality tests between brothers [4]. This requires some coding, because the function $F$ can refer to one representative for each class only, hence we cannot keep directly in the memory the subtree recognized so far. However, it is possible to show that


Fig. 3. A tree $t$ and a computation of $A$ on $t$.
any language recognized by an automaton with equality tests between brothers (and, more generally, with nonoverlapping equality tests) is also accepted by an automaton with one memory. We do not need the projections here.

In some respect, our definition is a generalization of both models: we can both use a stack and check for equality, and keep record of deep subtrees. We avoid overlapping tests, which yield undecidability [25,11], because we allow only one representative of each class in the function in the body of $F$.

Theorem 15. The emptiness of the language recognized by a tree automaton with one memory is decidable in DEXPTIME. More generally, the reachability of a given configuration is decidable in DEXPTIME.

Proof. For every $q \in Q$, let $M_{q}$ be the subset of $T(\Gamma)$ of memory contents $m$ such that there is a tree $t$ and a computation of the automaton on $t$ yielding the configuration $\langle q, m\rangle$. We prove that the sets $M_{q}$ are the least solutions of the definite set constraint with non-emptiness guards $C_{A}$, consisting, for each transition rule $f\left(q_{1}, \ldots, q_{n}\right) \xrightarrow[F]{c} q$ of the inclusion

$$
\operatorname{nonempty}\left(e_{q_{i_{1}}}\right), \ldots, \operatorname{nonempty}\left(e_{q_{i_{k}}}\right) \Rightarrow F\left(e_{q_{i_{1}}}, \ldots e_{q_{i_{k}}}\right) \subseteq X_{q}
$$

and $e_{q_{i}}$ is the intersection for all indices $l$ equivalent (w.r.t. $c$ ) to $i_{j}$ of $X_{q l}$.
$C_{A}$ can be assumed to be an intersection constraint with non-emptiness guards (see Section 3).

First, the assignment $\sigma_{0}$ which maps every $X_{q}$ to $M_{q}$ is a solution of the constraint. Indeed, consider any clause of the above form with $F=\lambda x_{1}, \ldots, x_{k} \cdot G$ and assume (for simplicity) that $x_{1}, \ldots, x_{r}$ do not occur in $G$, while $x_{r+1}, \ldots, x_{k}$ occur (once) in $G$.

If $\llbracket e_{q_{i_{j}}} \rrbracket_{\sigma_{0}} \neq \emptyset$ for every $i_{j}$, then it is possible to reach configurations $\left\langle q_{1}, m_{1}\right\rangle, \ldots$, $\left\langle q_{n}, m_{n}\right\rangle$ such that $m_{j} \in \llbracket e_{q_{i j}} \rrbracket \|_{\sigma_{0}}$, i.e. $m_{i}=m_{j}$ whenever $i=j \in c$. Now, consider any terms $m_{1}^{\prime}, \ldots, m_{k-r}^{\prime}$, respectively in $M_{q_{r+1}}, \ldots, M_{k}$. There are trees $t_{1}, \ldots, t_{r}, \ldots, t_{k}$ such that there are computations of the automaton on this trees yielding, respectively, the configurations $\left\langle q_{1}, m_{1}\right\rangle, \ldots,\left\langle q_{r}, m_{r}\right\rangle,\left\langle q_{r+1}, m_{1}^{\prime}\right\rangle, \ldots,\left\langle q_{k}, m_{k-r}^{\prime}\right\rangle$. From these configurations, reading the input $f$, the automaton can move to the configuration $\left\langle q, G\left(m_{1}^{\prime}, \ldots, m_{k-r}^{\prime}\right)\right\rangle$, hence $G\left(m_{1}^{\prime}, \ldots, m_{k-r}^{\prime}\right) \in M_{q}$.

Conversely, we have to prove that any solution $\sigma$ of the constraint is larger than $\sigma_{0}$. Let $m \in M_{q}$. There is a computation of the automaton on some tree $t$, yielding the configuration $\langle q, m\rangle$. We prove, by induction on $t$, that $m \in \llbracket X_{q} \rrbracket_{\sigma}$.

- If $t$ is a constant, then there must be a rule $a \underset{F}{\rightarrow} q$ and $F=m \in T(\Gamma)$. By definition, there is a constraint $F \subseteq X_{q}$. Hence $m \in \llbracket X_{q} \rrbracket_{\sigma}$.
- Now, let $t=f\left(t_{1}, \ldots, t_{n}\right)$ and let $f\left(q_{1}, \ldots, q_{n}\right) \xrightarrow[F]{c} q$ be the last rule applied in the computation yielding $\langle m, q\rangle$. Let moreover $\left\langle q_{1}, m_{1}\right\rangle, \ldots,\left\langle q_{n}, m_{n}\right\rangle$ be the configurations corresponding to computations on $t_{1}, \ldots, t_{n}$. By definition, $m_{i}=m_{j}$ whenever $(i, j) \in$ $c$ and $m=F\left(m_{i_{1}}, \ldots, m_{i_{k}}\right)$. By induction hypothesis, for every $i, m_{i} \in \llbracket X_{i} \rrbracket_{\sigma}$ and, because of the equality constraints, $m_{i} \in \llbracket e_{q_{i_{l}}} \rrbracket_{\sigma}$ if $\left(i_{l}, i\right) \in c$. It follows that $\sigma$ satisfies nonempty $\left(e_{q_{i}}\right)$ for all $j$ and, since $\sigma$ satisfies the clause associated with the rule, it satisfies $F\left(e_{q_{1}}, \ldots, e_{q_{i_{k}}}\right) \subseteq X_{q}$. In particular, $m \in \llbracket X_{q} \rrbracket_{\sigma}$.
This completes the proof that the sets $M_{q}$ are the least solutions of the constraint $C_{A}$.
Then the non-emptiness of the language recognized by $A$ reduces to the problem of deciding whether at least one of some designated variables gets a non-empty set in the least solution of the constraint. This is DEXPTIME-complete, thanks to Corollary 12.

The result can be generalized to alternating tree automata with one memory keeping the same complexity. Alternation here has to be understood as follows: we may replace the states occurring in the left hand sides of the rules with arbitrary positive Boolean combinations of states. The above proof simply works, using additional intersections and unions.

Corollary 16. The emptiness problem of alternating tree automata with one memory is DEXPTIME-complete.

Note however that the class of automata with one memory is neither closed under intersection nor complement (both yield undecidable models).

## 5. Set constraints with equality tests

### 5.1. Definition of the class

### 5.1.1. General set constraints with equality tests

We consider now definite set constraints as in Section 3, with non-emptiness constraints and with an additional construction: function symbols can be labeled with equality tests, which are conjunctions of equalities $p_{1}=p_{2}$ between paths. The intention is to represent sets of terms $t$ such that the subterms at positions $p_{1}$ and $p_{2}$ are identical. We assume, without loss of generality, that there is no union and no projection symbol, which, as we have seen, is not a restriction (provided that the equality tests do not overlap projection symbols).

We use the standard notations on terms [14]. Let us recall some of them. A position will be a string of nonnegative integers. A term $t$ labeled with $\mathcal{F}$ can be seen as a mapping from the set $\operatorname{Pos}(t)$ of its positions to $\mathcal{F}$. The subterm of $t$ at position $p$ is written $\left.t\right|_{p}$.

An equality constraint $c$ is an equivalence relation on a finite set of positions $P(c)$. We assume that no strict prefix of a position in $P(c)$ does belong to $P(c)$ (this restriction will be dropped in Section 5.2.1). We will often write equality constraints as finite sets (or finite conjunctions) of expressions $p_{1}=p_{2}$ where $p_{1}, p_{2}$ are positions. Then, it must be understood that $c$ is the least equivalence relation containing the pairs ( $p_{1}, p_{2}$ ) on the set of positions occurring in some of the equalities. We also say that a position $p$ is checked by $c$ when $p \in P(c)$.

A term $t$ satisfies $c$, which we write $t \vDash c$, if every path in $P(c)$ is a position of $t$ and moreover, $\left.t\right|_{p_{1}}=\left.t\right|_{p_{2}}$ (the subterms of $t$ at positions $p_{1}$ and $p_{2}$ are identical).
We enrich the set expressions of Section 3 with the construction $f^{c}\left(e_{1}, \ldots, e_{n}\right)$ where $c$ is an equality constraint. These expressions are interpreted as follows:

$$
\llbracket f^{c}\left(e_{1}, \ldots, e_{n}\right) \rrbracket_{\sigma} \stackrel{\text { def }}{=}\left\{t \in \llbracket f\left(e_{1}, \ldots, e_{n}\right) \rrbracket_{\sigma} \mid t \vDash c\right\}
$$

The set of paths in an expression $e$ is defined as follows:

$$
\begin{gathered}
\Pi\left(f^{c}\left(e_{1}, \ldots, e_{n}\right)\right) \stackrel{\text { def }}{ }\{\varepsilon\} \cup 1 \cdot \Pi\left(e_{1}\right) \cup \ldots \cup n \cdot \Pi\left(e_{n}\right) \\
\Pi\left(e_{1} \cap e_{2}\right) \stackrel{\text { def }}{=} \Pi\left(e_{1}\right) \cup \Pi\left(e_{2}\right) \\
\Pi(X) \stackrel{\text { def }}{=}\{\varepsilon\} .
\end{gathered}
$$

Let $p \in \Pi(e)$. We let $\left.e\right|_{p}$ be the set of subexpressions at position $p$ :

$$
\begin{gathered}
\left.e\right|_{\varepsilon} \stackrel{\text { def }}{=}\{e\} \\
\left.\left.\left.\left(e_{1} \cap e_{2}\right)\right|_{i \cdot p} \stackrel{\text { def }}{\text { def }} e_{1}\right|_{i \cdot p} \cup e_{2}\right|_{i \cdot p} \\
\left.\left(f^{c}\left(e_{1}, \ldots, e_{n}\right)\right)\right|_{i \cdot p} ^{\text {def }}=\left.e_{i}\right|_{p} \\
\left.X\right|_{i \cdot p} \stackrel{\text { def }}{=} \emptyset .
\end{gathered}
$$

When $\left.e\right|_{p}$ contains only one element, we confuse $\left.e\right|_{p}$ with this element and say that $\left.e\right|_{p}$ is the subexpression of e at $p$.

We will assume that, in every expression $f^{c}\left(e_{1}, \ldots, e_{n}\right)$,

$$
P(c) \subseteq \Pi\left(f^{c}\left(e_{1}, \ldots, e_{n}\right)\right)
$$

All other constructions are the same as in Section 3. In particular, right hand sides of inclusions should not contain constructions $f^{c}$. When $c$ is empty, we may omit it or write $T$.

Example 17. $f^{21=12}(f(Z, Y) \cap X, g(X) \cap Y) \subseteq f(Y, g(X))$ is an inclusion constraint. $\sigma=\{X \mapsto\{a, b, f(a, b)\} ; Y \mapsto\{b, g(a), g(b), f(a, b)\} ; Z \mapsto\{a, b\}\}$ is a solution of the constraint since $\llbracket f^{12=21}(f(Z, Y) \cap X, g(X) \cap Y) \rrbracket_{\sigma}=\{f(f(a, b), g(b))\}$.

### 5.1.2. A complete deduction system

We first design a complete deduction system and show that every satisfiable set constraint has a least solution. These results are not meant to be practical.

Let $S$ be a set constraint as in the previous section, whose variables are $\operatorname{Var}(S)=$ $\left\{X_{1}, \ldots, X_{n}\right\}$. Let $\Sigma$ be the subset of $\left(2^{T(\mathcal{F})}\right)^{n}$ of assignments $\sigma$ mapping every variable $X_{i}$ to a finite set.

We may assume in this section that, in every clause

$$
\phi \Rightarrow e \subseteq e^{\prime}
$$

the expression $e^{\prime}$ does not contain intersection symbols. This is not a restriction as a clause $\phi \Rightarrow e \subseteq e^{\prime}\left[e_{1} \cap e_{2}\right]$ is equivalent to the two clauses $\phi \Rightarrow e \subseteq e^{\prime}\left[e_{i}\right]$ for $i=1,2$.

With this assumption, we can associate with each right hand side of an inclusion $e^{\prime}$ a term $t_{e^{\prime}} \in T(\mathcal{F}, \mathcal{X})$ such that every variable occurs only once in $t_{e^{\prime}}$ and $e^{\prime}$ is obtained from $t_{e^{\prime}}$ by substituting set variables to the variables of $t_{e^{\prime}}$.
We define the one step deduction relation $T_{S}$ on $\Sigma \cup\{\square\}$ as follows:

## $T_{S}(\square) \stackrel{\text { def }}{=} \square$.

If there is a clause $\phi \Rightarrow e \subseteq e^{\prime}$ in $S$ such that $\sigma \vDash \phi$ and there is a $t \in \llbracket e \rrbracket_{\sigma}$ such that $t$ is not an instance of $t_{e^{\prime}}$, then $T_{S}(\sigma)=\square$.
Otherwise, for each clause $C=\phi \Rightarrow e \subseteq e^{\prime}$ in $S$

- If $\sigma \not \vDash \phi$, then we let $\theta_{C}$ be the assignments mapping every set variable to the empty set
- If $\sigma \vDash \phi$, then for every term $t \in \llbracket e \rrbracket_{\sigma}$, and every set variable $X$, we let $\theta_{t, C}(X)$ be the set of terms $\left.t\right|_{p}$ such that $\left.e^{\prime}\right|_{p}=X$. Finally, we define $T_{S}(\sigma)$ by:

$$
\llbracket X_{i} \rrbracket_{T_{S}(\sigma)} \stackrel{\text { def }}{=} \llbracket X_{i} \rrbracket_{\sigma} \bigcup_{C \in S, t \in \llbracket \ell \rrbracket_{\sigma}} \theta_{t, C}\left(X_{i}\right) .
$$

We let $\sigma_{\emptyset}$ be the assignment mapping every set variable to the empty set and we define $\sigma^{\omega}$ as the least fixed point of $\vdash$ :

$$
\sigma^{\omega}\left(X_{j}\right)=\bigcup_{i=1}^{\infty} T_{S}^{i}\left(\sigma_{\emptyset}\right)\left(X_{j}\right)
$$

if $T_{S}^{i}\left(\sigma_{\emptyset}\right) \neq \square$ for every $i$ and $\sigma^{\omega}=\square$ otherwise.
Example 18. Consider the set constraint $S$ consisting in the single clause $C=f(a) \subseteq$ $f(X)$. Then $t_{f(X)}=f(x)$ and $\llbracket f(a) \rrbracket_{\sigma_{\emptyset}}=\{f(a)\}$. The term $f(a)$ is an instance of $f(x)$ and $\theta_{f(a), C}(X)=\{a\}$.

Then, applying the deduction step, we get $\llbracket X \rrbracket_{T_{S}\left(\sigma_{\emptyset}\right)}=\emptyset \cup\{a\}=\{a\}$ and $T_{S}^{2}\left(\sigma_{\emptyset}\right)=$ $T_{S}\left(\sigma_{\emptyset}\right): \sigma^{\omega}(X)=\{X \mapsto\{a\}\}$.

Proposition 19. $\sigma^{\omega}=\square$ iff $S$ is not satisfiable.
If $\sigma^{\omega} \neq \square$, it is the least solution of $S$.
Proof. It is similar to the standard result that the least fixed point of the direct consequence operator of a Horn clause set is the least model of the program.
If $\sigma^{\omega} \neq \square$, then $\sigma^{\omega}$ is contained in any solution of $S$ (by induction on $i, T_{S}^{i}\left(\sigma_{\emptyset}\right)$ is contained in any solution of $S$ ).

Now, if $\sigma^{\omega} \neq \square$, then $\sigma^{\omega}$ is a solution of $S$ : this is a routine verification.

### 5.1.3. An undecidability result

As a consequence of undecidability results on tree automata with equality tests (see e.g. [11]), the satisfiability of such general constraints is undecidable, because of possible overlapping tests.

Proposition 20. The satisfiability of such general constraints (even without non-emptiness preconditions) is undecidable.

Note that such a result is consistent with Theorem 6 and the translation of security problems into set constraints as given in Section 6.1. We sketch the proof of this proposition, because, even if the reader should already be convinced, the proof sheds some lights on the restrictions we take later on.

Proof (sketch). We encode Turing machine computations. A configuration is represented as a triple containing the state, the part of the tape on the left of the head (including the head position) and the part of the tape on the right of the head. Tape contents are encoded using unary symbols (one for each element in the tape alphabet), in such a way that symbols which are close to the head appear on the top of the terms. For instance a tape content $\begin{gathered}a b a a b a a b \\ \uparrow\end{gathered}$ is represented by the words $b(a(a(b(a(0))))), a(a(b(0)))$. We use a binary tupling symbol $\left\langle{ }_{-},{ }^{\prime},\right\rangle$ to put together the two components of the tape and the state. Now, for instance, with each transition rule $\langle q, a\rangle \rightarrow\left\langle q^{\prime}, b\right.$, left $\rangle$ we associate the constraint:

$$
f^{12=2121,131=213}\left(\left\langle q^{\prime}, X, b(Y)\right\rangle, Z \cap f(\langle q, a(X), Y\rangle, \mathrm{T})\right) \subseteq Z
$$

$X, Y$ being tape contents, the equality tests ensure that we keep the same remaining tape contents when we move from one state to another.

The idea is that the least solution $\sigma_{0}$ of the constraint will assign to $Z$ the (encoding of the) set of computations of the Turing machine. Adding

$$
Z \cap f\left(\left\langle q_{f}, \top, T\right\rangle, T\right) \subseteq X_{0}
$$

for the final states and

$$
\left\langle q_{0}, 0,0\right\rangle \subseteq Z
$$

for the initial state, the emptiness of $\sigma_{0}\left(X_{0}\right)$ is equivalent to the halting problem (i.e. the reachability of the state $q_{f}$ ).

### 5.1.4. Basic variables and expressions

That is why we are going to put more restrictions on the constraints. The idea is to divide the set variables into two sets: the basic and the nonbasic variables. The basic ones correspond to sets of terms whose only a fixed part can be seen. This corresponds to noninvertible symbols in Section 2.2. We do not impose any restrictions on the equality tests for such basic variables since, intuitively, the noninvertible symbols impose a boarder in the terms under which no test takes place, hence limiting the overlaps of equalities which yield the undecidability result.

For nonbasic variables, we impose a restriction, which, roughly, allows to check the equalities using one memory only. The goal is of course to use the results of Section 4.
If $X$ is a variable of a constraint $S$, then let $\mathcal{R}(X)$ be the set of atomic constraints whose right hand side contains $X$.

We introduce now one-way function symbols of a constraint $S$. This notion is of course related to the one-way function symbols of Section 2 (it is a generalization). Intuitively, a symbol is one-way in a constraint $S$ if, in each of its applications, there is no way to look at the subterms. "Looking at the subterms" occur in two cases: when we apply a "projection" (i.e. when there is an inclusion whose right member is headed with that symbol) and when we check for equality of some subterms.

Definition 21. A function symbol $g$ is one-way in a set constraint $S$ if

- it does not occur on the right of an inclusion constraint of $S$
- in any expression $e=f^{c}\left(e_{1}, \ldots, e_{n}\right)$ occurring in $S$, for every $\pi \in P(c)$ and for every strict prefix $\pi^{\prime}$ of $\pi,\left.e\right|_{\pi^{\prime}}$ does not contain any expression headed with $g$.

Let $\mathcal{O} \mathcal{F}(S)$ be the set of one-way function symbols in $S$.
Definition 22. The basic variables of a set constraint $\gamma$ is the largest set of variables occurring in $\gamma$ such that

- If $X$ is basic then $\mathcal{R}(X)$ only contains one-way symbols and basic variables.
- If $X$ is basic then
- either $\mathcal{R}(X)$ contains only one clause and this clause has the form: $\phi \Rightarrow e \subseteq X$ where $X$ does not occur in $e$.
- or for every function symbol $f$ occurring in $\mathcal{R}(X)$, if $f$ occurs in some $\mathcal{R}(Y)$ then $Y$ is basic.

Intuitively, the function symbols used recursively to construct basic variables cannot be used for nonbasic variables.

Example 23. The following example is inspired by examples from Section 2. Let Nat, A, DA, HA, M, Key, In be set variables and $\gamma$ consist of:

| 0 | $\subseteq$ Nat | $\operatorname{succ}($ Nat $)$ | $\subseteq \mathrm{Nat}$ |
| ---: | :--- | ---: | :--- |
| $d a($ Nat $)$ | $\subseteq \mathrm{DA}$ | $h a($ Nat $)$ | $\subseteq \mathrm{HA}$ |
| DA | $\subseteq \mathrm{A}$ | HA | $\subseteq \mathrm{A}$ |
| $K(\mathrm{~A}, \mathrm{~A})$ | $\subseteq$ Key | $\operatorname{shr}(\mathrm{A})$ | $\subseteq \mathrm{M}$ |
| A | $\subseteq \mathrm{M}$ | Key | $\subseteq \mathrm{M}$ |
| $\langle\mathrm{M}, \mathrm{M}\rangle$ | $\subseteq \mathrm{M}$ | $\{\mathrm{M}\}_{\mathrm{M}}$ | $\subseteq \mathrm{M}$ |

Intruder capabilities such as

$$
\begin{aligned}
\langle\ln , \operatorname{In}\rangle & \subseteq \ln & \{\ln \}_{\ln } & \subseteq \ln \\
\ln \cap\langle T, T\rangle & \subseteq\langle\ln , \ln \rangle & \ln \cap\{T\}_{\ln } & \subseteq\{\ln \}_{\ln } \\
A & \subseteq \ln & \operatorname{shr}(D A) & \subseteq \ln \\
\ln \cap\langle T, T, T\rangle & \subseteq\langle\ln , \ln , \ln \rangle & &
\end{aligned}
$$

And protocol-specific constraints such as

$$
\begin{aligned}
\langle\mathrm{A}, \mathrm{~A}\rangle & \subseteq \ln \\
\left\rangle^{c}\left(\{\langle\mathrm{~A}, \mathrm{Key}, \mathrm{M}\rangle\}_{\operatorname{shr}(\mathrm{A})} \cap \operatorname{In},\{m(\mathrm{~A}, \mathrm{~A})\}_{\text {Key }}, \mathrm{M}\right)\right. & \subseteq \mathrm{In}
\end{aligned}
$$

where $c$ stands here for $121=211 \wedge 111=212 \wedge 112=22 \wedge 113=3$. (We will see in section 6.1 how to translate cryptographic protocol into set constraints and, in particular, we will develop a full example). In this example, all function symbols are one-way, except the tupling $\left\langle_{-},\right\rangle$and $\left\langle_{-},{ }_{-},\right\rangle$and encryption $\left\}_{-}\right.$, because of the intruder's constraints.
Then In and M are not basic while all other variables are basic.

Remark. Conditional inverses of the form

$$
f\left(u_{1}, \ldots, u_{n}\right) \in I, u_{1} \in I, \ldots u_{k} \in I\left(k\langle n) \Rightarrow u_{k+i} \in I\right.
$$

can always be expressed using set constraints:

$$
f(\ln , \ldots, \ln , T, \ldots, T) \cap \ln \subseteq f(\ln , \ldots, \ln )
$$

The public key example is also given in Fig. 10.
The notion of basicity is extended to expressions: an expression $e$ is basic if

- $e$ is a basic variable or
- $e$ is an intersection $e_{1} \cap e_{2}$ and either $e_{1}$ or $e_{2}$ is basic
- $e$ is an expression $f\left(e_{1}, \ldots, e_{n}\right)$ (or $f^{c}\left(e_{1}, \ldots, e_{n}\right)$ ) and $e_{1}, \ldots, e_{n}$ are basic.


### 5.1.5. Our assumptions

Definition 24 (Basicity condition). An an equality test $c$ in an expression $f^{c}\left(e_{1}, \ldots, e_{n}\right)$ satisfies the basicity condition (w.r.t. a set of basic variables) if

$$
\left.\begin{array}{r}
p \cdot i \cdot q \sim_{c} p^{\prime} \\
i \neq j \\
p^{\prime} \not ¥_{\text {pref }} p \\
\forall w, p \cdot j \cdot w \not \chi_{c} p \cdot i \cdot q
\end{array}\right\} \Rightarrow \begin{aligned}
& \text { There are positions } p_{1}, p_{2} \text { such that } \\
& p_{1} \sim_{c} p^{\prime}, p_{2} \sim_{c} p \cdot j \text { and } \\
& \text { either }\left.e\right|_{p_{1}} \text { or }\left.e\right|_{p_{2}} \\
& \text { contains basic expressions only }
\end{aligned}
$$

where $\geqslant_{p r e f}$ is the prefix ordering on positions.


Fig. 4. The basicity condition.

An expression $e$ satisfies the basicity condition (w.r.t. a set of basic variables) if for each expression $f^{c}\left(e_{1}, \ldots, e_{n}\right)$, the equality test $c$ satisfies the basicity condition.

The situation is depicted in Fig. 4: one of the three terminal positions on the picture should hold basic expressions only.

Example 25. Let us examine the last constraint which is displayed in Example 23.

$$
\left\rangle^{c}\left(\{\langle\mathrm{~A}, \mathrm{Key}, \mathrm{M}\rangle\}_{\mathrm{shr}(\mathrm{~A})} \cap \operatorname{In},\{m(\mathrm{~A}, \mathrm{~A})\}_{\text {Key }}, \mathrm{M}\right) \subseteq \operatorname{In}\right.
$$

where $c$ stands for $121=211 \wedge 111=212 \wedge 112=22 \wedge 113=3$. At positions 121,111 , there is only one subexpression A , which is basic. At position 112, there is only one subexpression: Key, which is also basic. Remain the positions 3, 113, which do not hold basic expressions.

In the definition, $p^{\prime}$ can only be 3 and $p \in\{1,11\}$. The basicity condition reduces to check that subexpressions at positions $12,111,113$ are basic, which is the case here.

Note that, if there are only two occurrences of non-basic variables in the expression, then the basicity condition is always satisfied.

The basicity condition looks a bit complicated, but let us give more intuition.
From tree automaton point of view, while computing on the trees bottom-up, we want to be able to check the equalities without carrying more than one memory at each node. The difficulty is that we need the stability under intersection of this property.

Consider for instance the following constraints:

$$
\begin{aligned}
& f^{12=2}(X \cap f(X, Y), Y) \subseteq Y \\
& f^{11=2}(Y \cap f(X, Y), X) \subseteq X .
\end{aligned}
$$

Only one memory is sufficient to recognize the instances of any of the two constraints. Putting them together, however, we can derive

$$
f^{12=2}\left(f^{11=2}(Y \cap f(\mathbf{X}, Y), X) \cap f(X, Y), Y\right) \subseteq Y
$$

Now, we need 2 memories to accept the instances of the left hand side since, when reaching $\mathbf{X}$ we must keep this term in the memory (it is checked for equality higher up) and we must also keep in the memory $f(\mathbf{X}, Y)$, which is also checked for equality higher up. Note that here $X, Y$ are not basic since $f$ is not a one-way function symbol.
Actually, a more natural, weaker, condition would be to assume that, in any $c$, if $p, q$ are in two different equivalence classes, then they do not share any prefix. Imposing such a condition only yields an undecidable class of constraints.

On the other hand, if we intersect a basic expression with any expression, the result is a basic expression. Hence, the basicity condition expresses roughly that on the sides of a path checked for equality, we only find basic expressions, freeing us from keeping additional information when we intersect with another expression.

The basicity condition is also relevant for our application, as we will see.
The constraints satisfying the basicity condition are called set constraints with equality tests (ET-constraints for short). Let us summarize:

Definition 26. An ET-constraint is a finite conjunction of clauses

$$
\phi \Rightarrow e \subseteq e^{\prime}
$$

in which $e, e^{\prime}$ are set expressions built using

- Set variables.
- The constant symbol $\perp$.
- Intersection.
- Function symbol application $f^{c}(\ldots)$ or $f(\ldots)$.

We assume:

- That right hand sides (the expression $e^{\prime}$ above) do not make use of the constructions $f^{c}$ with a nonempty $c$.
- For every construction $f^{c}\left(e_{1}, \ldots, e_{n}\right), P(c) \subseteq \Pi\left(f^{c}\left(e_{1}, \ldots, e_{n}\right)\right)$.
- The basicity condition.

ET-constraints contain properly intersection constraints since we can construct an ETconstraint whose least solution is the set of trees $\Delta=\{f(t, t) \mid t \in T(\mathcal{F})\}$. The only other decidable set constraint formalism which allows to express $\Delta$ is the class defined in [6], in which, however, equality tests are restricted to brother positions (which is not the case here). On the other hand, we have restrictions which are not present in [6].

### 5.2. Saturation

We use here a fixed point computation method which is similar to the one in [7]: the goal is to deduce enough consequences so that the inclusions whose right hand side is not a variable become redundant, hence can be discarded. Unfortunately, the first step (representation) in [7] cannot be used in the same way here, since it does not preserve the class of constraints we consider.

First, we prepare the saturation in Sections 5.2.1-5.2.5, reducing ET-constraints to SETconstraints. Then we design deduction rules (generalizing the rules of [7]), which preserve SET-constraints (Lemma 47), are terminating (Lemma 49) and which yield to a saturated system: the solved part of a saturated constraint $S$ is satisfiable if and only if $S$ is satisfiable (Theorem 50).
The preparation steps consist in:

- normalizing the set expressions (Section 5.2.1): in particular, the subexpression at position $p$ of a normal expression is uniquely defined (Lemma 29),
- flattening the set expressions: we only keep paths which checked by an equality constraint (Section 5.2.2),
- getting ride of basic variables (Section 5.2.3): we show that the constraint $S$ is equivalent to finitely many instances of $S$, in which basic variables are replaced by ground expressions (Lemma 39),
- simplifying again the expressions, taking advantage of the previous steps (section 5.2.5). We start with some simplifications of the constraints.


Property $2 \quad$ Property 6
Fig. 5. The properties 2 and 6.

### 5.2.1. Normalization

For every expression $e$, let us define two notions of size:

- $|e|_{\mathcal{F}}$ is the cardinal of $\left.\bigcup_{p \in \Pi(e)} e\right|_{p}$. This is proportional to the memory size, which is required to store the expression, regardless to the equality tests.
- $|e|_{t}$ is the sum, for every expression $\left.f^{c}\left(e_{1}, \ldots, e_{n}\right) \in \bigcup_{p \in \Pi(e)}{ }^{e}\right|_{p}$ of the size of $c$. The size of an individual test $c$ is the sum of sizes of positions checked by $c$.
The goal of the first transformation step (Normalization) is to reduce the expression to a normal form.

Definition 27. An expression $e$ is normal if the following conditions are satisfied for $e$ :

1. All subexpressions of $e$ satisfy the basicity condition.
2. If $\left.f^{c}\left(e_{1}, \ldots, e_{n}\right) \cap e^{\prime} \in e\right|_{p_{0}}, p \sim_{c} q$ and $p \cdot p_{1} \sim_{c} q_{1}$ for a nontrivial $p_{1}$, then $\left.e\right|_{p_{0} \cdot p \cdot p_{1}}$ is a basic expression("For ancestor positions, the lowest one is basic").
3. For every $p \in \Pi(e)$, if $\left.g^{c}\left(e_{1}, \ldots, e_{n}\right) \cap e^{\prime} \in e\right|_{p}$, then, for every $p_{1} \sim_{c} p_{2}$, the subexpressions at positions $p_{1}$ and $p_{2}$ in $g^{c}\left(e_{1}, \ldots, e_{n}\right)$ are identical.
4. for every equality test $c$ occurring in $e$, every equivalence class of $c$ contains at least two positions which do not share any nontrivial prefix.
5. For every $p \in \Pi(e),\left.e\right|_{p}$ is either an intersection of variables or an intersection $g^{c}\left(e_{1}, \ldots\right.$, $\left.e_{n}\right) \cap X_{1} \cap \ldots \cap X_{m}$, in which case, for every $p \in \Pi(e),\left.e\right|_{p}$ is a singleton.
6. If $\left.f^{c}\left(e_{1}, \ldots, e_{n}\right) \cap e^{\prime} \in e\right|_{p_{0}}, p \cdot p_{1} \sim_{c} q$ for non-empty $p, p_{1}, g^{c^{\prime}}\left(e_{1}^{\prime}, \ldots, e_{m}^{\prime}\right) \cap e^{\prime \prime} \in$ $\left.e\right|_{p_{0} \cdot p}$, then either $\left.e\right|_{p_{0} \cdot p \cdot p_{1}}$ is a basic expression, or else for every $p^{\prime} \sim_{c^{\prime}} q^{\prime},\left.e\right|_{p_{0} \cdot p \cdot p^{\prime}}$ is a basic expression ("For overlapping tests, the lowest one is basic").

Conditions 1 and 2 are satisfied initially. Actually, even a property stronger than condition 2 is initially satisfied since, so far, any two distinct positions in $P(c)$ are incomparable w.r.t. the prefix ordering. We need however this weaker property to keep it invariant.

Properties 2, 6 are illustrated in Fig. 5.
The main result of this section, whose proof is quite long and technical and is given in Appendix A is the following:

Lemma 28. Every expression e which satisfies the basicity condition can be transformed into a normal expression $e^{\prime}$ such that, for every $\sigma, \llbracket e \rrbracket_{\sigma}=\llbracket e^{\prime} \rrbracket_{\sigma}$.

We also conjecture that the transformation, as described in the appendix, yields an expression $e^{\prime}$ such that $\left|e^{\prime}\right|_{\mathcal{F}}$ and $\left|e^{\prime}\right|_{t}$ are polynomially bounded by $|e|_{t}$ and $|e|_{F}$.

As a side consequence, the subexpression at a given position is now defined in a unique way:

Lemma 29. If e is an expression satisfying condition 5, then for every $p \in \Pi(e),\left.e\right|_{p}$ is a singleton.

Proof. We prove that $\left.e\right|_{p}$ is a singleton for every $p \in \Pi(e)$ by induction on $e$. If $e$ is a variable or a constant, then $\left.e\right|_{p}$ is a singleton consisting in $e$ itself.

Now, assuming $e$ satisfies condition 5, $e$ is either an intersection of variables or an expression $f^{c}\left(e_{1}, \ldots, e_{n}\right) \cap X_{1}, \ldots \cap X_{n}$. In the first case $\Pi(e)=\{\varepsilon\}$ and $\left.e\right|_{p}=\{e\}$ by definition. In the latter case, if $p=i \cdot p^{\prime}$, then $\left.e\right|_{p}=\left.\left.e_{i}\right|_{p^{\prime}} \cup X_{1}\right|_{i \cdot p^{\prime}} \ldots=\left.e_{i}\right|_{p^{\prime}}$ by definition. And, by induction hypothesis, $\left.e_{i}\right|_{p^{\prime}}$ is a singleton.

So, now, we can use the terminology "the subexpression at position $p$ ", as well as replacement at position $p: C[e]_{p}$ means either (this will be unambiguous from the context) that $e$ occurs at position $p$ in the expression $C[e]_{p}$ or that we have replaced the subexpression at position $p$ with $e$.

The normalization is extended to constraints: every expression occurring in the constraint can be assumed normal thanks to Lemma 28.

### 5.2.2. Abstractions

We abstract out subexpressions introducing new variables, as long as this preserves the form of the constraints. For instance, for contexts $C[]_{p}$, an inclusion $e \subseteq C\left[f\left(\overrightarrow{e^{\prime}}\right)\right]_{p}$ becomes $C[X]_{p} \subseteq e, f\left(\overrightarrow{e^{\prime}}\right) \subseteq X$ where $X$ is a new variable. This results in an equivalent constraint (on the original variables) in which the inclusions are $e \subseteq e^{\prime}$ where $e^{\prime}$ is either an intersection of variables $X_{1} \cap \ldots \cap X_{n}$ or an expression $f\left(X_{1}, \ldots, X_{n}\right)$.

More formally, we use the following rules, assuming that $n \geqslant 2$ and $p$ is not the root position:

$$
\begin{aligned}
\left(\mathrm{A}_{1}\right) & \phi \Rightarrow f^{c}(\vec{e}) \cap e_{1} \subseteq e^{\prime} \\
\left(\mathrm{A}_{2}\right) & (\exists X) f^{c}(\vec{e}) \subseteq X, \phi \Rightarrow X \cap e_{1} \subseteq e^{\prime}, \\
\phi \Rightarrow e \subseteq C\left[f\left(\overrightarrow{e^{\prime}}\right)\right]_{p} & \rightarrow(\exists X) \phi \Rightarrow e \subseteq C[X]_{p}, f\left(\overrightarrow{e^{\prime}}\right) \subseteq X, \\
& X \subseteq f\left(\overrightarrow{e^{\prime}}\right), \\
\left(\mathrm{A}_{3}\right) \phi \Rightarrow e \subseteq C\left[e_{1} \cap \ldots \cap e_{n}\right]_{p} & \rightarrow(\exists X) \phi \Rightarrow e \subseteq C[X]_{p}, e_{1} \cap \ldots \cap e_{n} \subseteq X .
\end{aligned}
$$

In these rules, $X$ is a new variable: we assume that there is no capture. The following lemma is a consequence of the definitions:

Lemma 30. Applying abstraction does terminate on any constraint $S$, resulting in a constraint $S^{\prime}$ such that the solutions of $S$ are the restrictions of solutions of $S^{\prime}$ to the free variables of $S$. Moreover, if $S$ is an ET-constraint, then so is $S^{\prime}$ and if every expression is normal in $S$, then every expression is normal in $S^{\prime}$.

We can also abstract out in the conditions of the inclusions. However, using such a rule in an unrestricted way would lead to non termination of the saturation. That is why we are going to use it only once, to simplify the original constraint and forget it afterwards:

$$
\begin{aligned}
\left(\mathrm{A}_{4}\right) & \text { nonempty }(e), \phi \\
& \rightarrow(\exists Y) e \subseteq Y \text { e } \subseteq \subseteq e_{2} \\
& \wedge \text { nonempty }(Y), \phi \Rightarrow e_{1} \subseteq e_{2} .
\end{aligned}
$$

In this rule, $e$ is assumed not to be a variable. It is also assumed that there is no capture ( $Y$ is a new variable).

Lemma 31. ( $\mathrm{A}_{4}$ ) preserves the solutions of the constraint.
Proof. Assume $S \rightarrow S^{\prime}$ using the rule (A4). If $\sigma$ is a solution of $S$, extending $\sigma$ with $Y \mapsto \llbracket e \rrbracket_{\sigma}$ yields a solution of $S^{\prime}$.

Conversely, if $\sigma^{\prime}$ is a solution of $S^{\prime}$, then its restriction $\sigma$ to variables other than $Y$ is a solution of $S$ : either $\sigma \neq$ nonempty $(e), \phi$, in which case $\sigma$ satisfies nonempty $(e), \phi \Rightarrow$ $e_{1} \subseteq e_{2}$, or else $\llbracket e \rrbracket_{\sigma}$ is non-empty. In the latter case, $\llbracket Y \rrbracket_{\sigma^{\prime}}$ is nonempty (because of the constraint $e \subseteq Y$ ) and $\llbracket e_{1} \rrbracket_{\sigma}=\llbracket e_{1} \rrbracket_{\sigma^{\prime}} \subseteq \llbracket e_{2} \rrbracket_{\sigma^{\prime}}=\llbracket e_{2} \rrbracket_{\sigma}$.

Inspecting the normal forms w.r.t. $\left(A_{1}\right),\left(A_{2}\right),\left(A_{3}\right),\left(A_{4}\right)$, together with our hypotheses, the atomic constraints are now of the form $\phi \Rightarrow e \subseteq e^{\prime}$ where $\phi$ is a conjunction of nonempty $(X), e^{\prime}$ is of the form $f\left(X_{1}, \ldots, X_{n}\right)$ or $X_{1} \cap \ldots \cap X_{n}$ and $e$ is either $X_{1} \cap \ldots \cap X_{n}$ or $f^{c}(\vec{e})$.

### 5.2.3. Getting rid of basic variables

Next, we can get rid of basic type variables. The main idea is that we can replace each basic variable with a suitably chosen finite set, while keeping the desirable properties. This is described in the next lemmas.
We let $\mathcal{B}(S)$ be the set of basic variables of $S$. We split each ET-constraint into two parts $S=S_{B} \uplus S_{N B}$ : $S_{B}$ is the union of $\mathcal{R}(X)$ for $X \in \mathcal{B}(S)$ and $S_{N B}$ is the remaining constraint.

Remember that for any basic variable, either every function symbols of $\mathcal{R}(X)$ occurs only in $S_{B}$ (first type) or $\mathcal{R}(X)$ contains only one clause on the form $\phi \Rightarrow C_{X}\left(X_{1}, \ldots, X_{k}\right) \subseteq X$ where $X$ is distinct from the $X_{i}$ (second type). We first get ride of the basic variables $X$ of second type by replacing them by the clause $C_{X}\left(X_{1}, \ldots, X_{k}\right)$.

Lemma 32. Given an ET-constraint $S$, let

$$
S^{\prime} \xlongequal{\text { def }} S\left[X \mapsto C_{X}\left(X_{1}, \ldots, X_{k}\right)\right]_{X \text { of second type }}
$$

Then $S^{\prime}$ is an ET-constraint and $S$ is satisfiable if and only if $S^{\prime}$ is satisfiable.
Proof. Since $C_{X}\left(X_{1}, \ldots, X_{k}\right)$ contains only one-way function symbols and basic variables, $C_{X}\left(X_{1}, \ldots, X_{k}\right)$ is a basic expression thus $S^{\prime}$ is an ET-constraint.

If $\sigma$ is a solution of $S^{\prime}$, then $\sigma$ extended to the basic variables of second type by $\sigma(X)=$ $\llbracket C_{X}\left(X_{1}, \ldots, X_{k}\right) \rrbracket_{\sigma}$ is clearly a solution of $S$.

Conversely, if $S$ is satisfiable, then $S$ has a minimal solution $\sigma$. By minimality of $\sigma$, we have necessarily $\llbracket X \rrbracket_{\sigma}=\llbracket C_{X}\left(X_{1}, \ldots, X_{k}\right) \rrbracket_{\sigma}$. Thus $\sigma$ is solution of $S^{\prime}$.

From now on, we consider only ET-constraints with only basic variables of second type. In particular, the function symbols occurring in $S_{B}$ do not occur in $S_{N B}$.

Example 33. We consider some of the constraints presented in Example 23.

$$
\begin{aligned}
& S_{B}\left[\begin{array}{rlrl}
0 & \subseteq \mathrm{Nat} & \operatorname{succ}(\mathrm{Nat}) & \subseteq \mathrm{Nat} \\
d a(\mathrm{Nat}) & \subseteq \mathrm{DA} & h a(\mathrm{Nat}) & \subseteq \mathrm{HA} \\
\mathrm{DA} & \subseteq \mathrm{~A} & \mathrm{HA} & \subseteq \mathrm{~A} \\
K(\mathrm{~A}, \mathrm{~A}) & \subseteq \text { Key }
\end{array}\right. \\
& S_{N B}\left[\begin{array}{rl}
\mathrm{A} & \subseteq \mathrm{M} \\
& \text { Key } \subseteq \mathrm{M} \\
\langle\mathrm{M}, \mathrm{M}\rangle & \subseteq \mathrm{M}\{\mathrm{M}\}_{\mathrm{M}} \subseteq \mathrm{M} \\
\operatorname{shr}(\mathrm{~A}) & \subseteq \mathrm{M}
\end{array}\right.
\end{aligned}
$$

Then there is one basic variable of second type: Key. Thus we transform our ET-constraint following Lemma 32:

Lemma 34. $S_{B}$ has a least solution $\sigma_{m}$. It is possible to compute a finite tree automaton $\mathcal{A}_{m}$, whose states are finite sets of variables in $S_{B}$ and such that $\sigma_{m}(X)$ is the set of trees accepted in the state $\{X\}$.

Proof. Since one-way function symbols do not occur on the right of inclusions, in any constraint $\phi \Rightarrow e \subseteq e^{\prime}, e^{\prime}$ is an intersection of variables. Then $S_{B}$ is satisfiable: assigning every variable to $T(\mathcal{O F})$ is a solution. Then, by Proposition 19 there is a minimal solution $\sigma_{m}$.

We can also easily construct the minimal solution in an effective way, applying e.g. the saturation rules of Fig. 2 to this particular case: because there are only one-way functions in $S_{B}$, there is no construction $f^{c}(\ldots)$ here. The saturated constraint coincides here with the solved form (since there are no function symbols on the right).
As in Section 3.2, the solved form corresponds to a tree automaton $\mathcal{A}_{m}$ whose states are set variables.

Let $\approx$ be any equivalence relation on $T(\mathcal{O F}) . \approx$ is extended to the least congruence relation on $T(\mathcal{F})$, which we write again $\approx$. Then, every assignment $\sigma$ from the set of variables to $2^{T(\mathcal{F})}$ is extended into the assignment $\sigma_{\approx}$ defined by:

$$
\sigma_{\approx(X)} \stackrel{\text { def }}{=}\{t \in T(\mathcal{F}) \mid \exists u \in T(\mathcal{F}), t \approx u, u \in \sigma(X)\} .
$$

in other words, $\sigma$ is saturated by $\approx$.

Lemma 35. For every equivalence relation $\approx$ on $T(\mathcal{O F})$, if $\sigma$ is a solution of an ETconstraint $S$, in normal form w.r.t. Norm, $\left(\mathrm{A}_{1}\right),\left(\mathrm{A}_{2}\right),\left(\mathrm{A}_{3}\right),\left(\mathrm{A}_{4}\right)$, then $\sigma \approx$ is a solution of $S$.

Proof. Assume $\phi \Rightarrow e \subseteq e^{\prime} \in S$. Since $\phi$ only consists of formulas nonempty $(X)$ where $X$ is a variable, $\sigma \vDash \phi$ if and only if $\sigma \approx \vDash \phi$.

Let $t \in \llbracket e \rrbracket_{\sigma}$ and $u \approx t$.
We prove, by induction on the size of $e^{\prime}$ that

$$
t \in \llbracket e^{\prime} \rrbracket_{\sigma} \Rightarrow u \in \llbracket e^{\prime} \rrbracket_{\sigma_{\approx}} .
$$

- If $e^{\prime}$ is a variable, the implication follows from the congruence property of $\approx$.
- If $e^{\prime}=e_{1} \cap e_{2}$, this is straightforward.
- If $e^{\prime}=f\left(e_{1}, \ldots, e_{n}\right)$ then $f \notin \mathcal{O} \mathcal{F}$ (by definition of one-way symbols). From $t \in \llbracket e^{\prime} \rrbracket$, it follows that $t=f\left(t_{1}, \ldots, t_{n}\right)$. Then $u=f\left(u_{1}, \ldots, u_{n}\right)$ with $t_{i} \approx u_{i}$ for every $i$, since $f \notin \mathcal{O} \mathcal{F}(S)$. By induction hypothesis, for every $i, t_{i} \in \llbracket e_{i} \rrbracket_{\sigma}$, hence $u_{i} \in \llbracket e_{i} \rrbracket_{\sigma_{\approx}}$, therefore $u \in \llbracket e^{\prime} \rrbracket_{\sigma_{\approx}}$.

Now, the idea is to construct a finite index relation $\approx$ such that we may interpret the basic variables in a set of representatives modulo $\approx$.

Lemma 36. There is a congruence $\approx$ and an assignment $\sigma^{0}$ to the variables of $S$ such that $\sigma_{\approx}^{0}=\sigma_{m}$ and $\sigma^{0}(X)$ is finite for every variable $X$.

Proof. We use the automaton describing $\sigma_{m}$ and we consider $\approx$ defined by $t \approx u$ iff $t$ and $u$ are accepted in exactly the same states of the automaton. Let $R$ be a set of representatives for $\approx$ such that if $f\left(t_{1}, \ldots, t_{n}\right)$ is in $R$ then, $t_{1}, \ldots, t_{n}$ are also in $R . \sigma^{0}$ assigns $R \cap \llbracket X \rrbracket_{\sigma_{m}}$ to $X$.

Remark. Note that every element of $R$ is recognized by at least one state $X$ of the automaton.

Example 37. We consider again the constraints presented in Example 33. The automaton $\mathcal{A}_{m}$ associated with $S_{B}$ is the following one:

$$
\begin{array}{rlrl}
0 & \rightarrow q_{\mathrm{Nat}} & \operatorname{succ}\left(q_{\mathrm{Nat}}\right) & \rightarrow q_{\mathrm{Nat}} \\
d a\left(q_{\mathrm{Nat}}\right) & \rightarrow q_{\mathrm{DA}} & h a\left(q_{\mathrm{Nat}}\right) & \rightarrow q_{\mathrm{HA}} \\
q_{\mathrm{DA}} & \rightarrow q_{\mathrm{A}} & q_{\mathrm{HA}} & \rightarrow q_{\mathrm{A}}
\end{array}
$$

Thus the equivalence classes are:

$$
\left\{\operatorname{succ}^{n}(0) \mid n \in \mathbb{N}\right\},\left\{d a\left(\operatorname{succ}^{n}(0)\right) \mid n \in \mathbb{N}\right\},\left\{h a\left(\operatorname{succ}^{n}(0)\right) \mid n \in \mathbb{N}\right\}
$$

We choose $R=\{0, h a(0), d a(0)\}$ and $\sigma^{0}($ Nat $)=\{0\}, \sigma^{0}(\mathrm{HA})=\{h a(0)\}, \sigma^{0}(\mathrm{DA})=$ $\{d a(0)\}, \sigma^{0}($ Agent $)=\{h a(0), d a(0)\}$.

If $\sigma$ is an assignment of basic set variables to finite set of terms in $T(\mathcal{O F})$, then $\sigma\left(S_{N B}\right)$ is the set constraint obtained, replacing each basic variable $X$ with $\bigcup_{t \in \sigma(X)} t$. Since basic
variables do not occur on the right hand sides of inclusions in $S_{N B}, \sigma\left(S_{N B}\right)$ can be normalized in an ET-constraint, removing unions occurring on the left or in the conditions by duplicating the constraints.

Example 38. In our example 37, $\sigma^{0}\left(S_{N B}\right)$ is equal to:

$$
\begin{array}{rlrl}
h a(0) & \subseteq \mathrm{M} & d a(0) & \subseteq \mathrm{M} \\
K(h a(0), h a(0)) & \subseteq \mathrm{M} & K(h a(0), d a(0)) & \subseteq \mathrm{M} \\
K(d a(0), h a(0)) & \subseteq \mathrm{M} & K(d a(0), d a(0)) & \subseteq \mathrm{M} \\
\langle\mathrm{M}, \mathrm{M}\rangle & \subseteq \mathrm{M} & \{\mathrm{M}\}_{\mathrm{M}} & \subseteq \mathrm{M} \\
\operatorname{shr}(h a(0)) & \subseteq \mathrm{M} & \operatorname{shr}(d a(0)) & \subseteq \mathrm{M}
\end{array}
$$

Lemma 39. Let $\sigma^{0}$ be the restriction to basic variables of the assignment defined as in Lemma 36. Let $\approx$ be the congruence as defined in Lemma 36. Then $S$ is satisfiable iff $\sigma^{0}\left(S_{N B} \cup S_{O F}\right)$ is satisfiable.

Proof. First assume that $S$ is satisfiable and $\sigma$ is a solution of $S$. Thanks to Lemma 35, we can assume w.l.o.g. that $\sigma=\sigma_{\approx}$. Let us construct $\sigma^{\prime}$ such that $\sigma_{\approx}^{\prime}=\sigma$ and $\sigma^{\prime}$ is a solution of $\sigma^{0}\left(S_{N B}\right)$.

For every term $t \in T(F)$, let $t \downarrow$ be its representative for $\approx$. We define

$$
\sigma^{\prime}(X)=\{t \downarrow \mid t \in \sigma(X)\} .
$$

Let us prove that $\sigma^{\prime}$ is a solution of $\sigma^{0}\left(S_{N B}\right)$. Let $\phi \Rightarrow e \subseteq e^{\prime}$ in $S_{N B}$. If $\sigma^{\prime} k \phi$ then $\sigma^{\prime}$ satisfies the clause. Otherwise, $\sigma \vDash \phi$ and therefore $\llbracket e \rrbracket_{\sigma} \subseteq \llbracket e^{\prime} \rrbracket_{\sigma}$. Let $t \in \llbracket e \rrbracket_{\sigma}$ (and hence $\left.t \in \llbracket e^{\prime} \rrbracket_{\sigma}\right)$. We want to prove that $t \downarrow \in \llbracket e^{\prime} \rrbracket_{\sigma^{\prime}}$. By abstraction, $e^{\prime}$ is either an intersection of variables or an expression $f\left(X_{1}, \ldots, X_{n}\right)$. In the first case, $t \in \llbracket e^{\prime} \rrbracket_{\sigma}$ implies, by definition of $\sigma^{\prime}, t \downarrow \in \llbracket e^{\prime} \rrbracket \rrbracket_{\sigma^{\prime}}$. In the second case, $f \notin \mathcal{O \mathcal { F }}$, by definition of one-way function symbols. Then $t \in \llbracket e^{\prime} \rrbracket_{\sigma}$ implies that $t=f\left(t_{1}, \ldots, t_{n}\right)$ and $t \downarrow=f\left(t_{1} \downarrow, \ldots, t_{n} \downarrow\right)$. Then $t_{i} \in \llbracket X_{i} \rrbracket_{\sigma}$ implies, by definition of $\sigma^{\prime}$, that $t_{i} \downarrow \in \llbracket X_{i} \rrbracket_{\sigma^{\prime}}$, hence $t \in \llbracket e^{\prime} \rrbracket_{\sigma^{\prime}}$.

Conversely, assume that $\sigma^{\prime}$ is the minimal solution of $\sigma^{0}\left(S_{N B}\right)$. We extend $\sigma^{\prime}$ with $\sigma^{0}$ to basic variables. Let us prove that $\sigma_{\approx}^{\prime}$ is a solution of $S$. We first need to establish some properties on $\sigma^{\prime}$ :

Lemma 40. If $u \in \llbracket X \rrbracket_{\sigma^{\prime}}$, then $u=u \downarrow$.
Proof. For every term $t \in T(F)$, let $t \downarrow$ be its representative for $\approx . t \downarrow=C\left[t_{1} \downarrow, \ldots, t_{n} \downarrow\right]$ for some context $C$, such that $t=C\left[t_{1}, \ldots, t_{n}\right]$ and for every term $u \approx t, u=C\left[u_{1}, \ldots, u_{n}\right]$ with $t_{i} \approx u_{i}$. The maximal context $C$ verifying the property above is called the canonical context of $t$. Note that since the $t_{i} \downarrow$ are representatives of the minimal solution of $S_{B}$, then the function symbols of the $t_{i} \downarrow$ do not occur in $S_{N B}$. In addition, the $t_{i}$ are equivalent (modulo $\approx$ ) to the $t_{i} \downarrow$, thus we have also that the function symbols of the $t_{i}$ do not occur in $S_{N B}$.

Let us first prove by induction on $e$ that:
for every $\sigma$, if for every term $t$ and every set variable $X, t \in \llbracket X \rrbracket_{\sigma}$ implies $t=t \downarrow$, then for every expression $e$ occurring in $\sigma^{0}\left(S_{N B}\right), t \in \llbracket e \rrbracket_{\sigma}$ implies $t=t \downarrow$.

Indeed, assume that for every term $t$ and every set variable $X, t \in \llbracket X \rrbracket_{\sigma}$ implies $t=t \downarrow$ and consider $e$ occurring in $\sigma^{0}\left(S_{N B}\right)$ and $u \in \llbracket e \rrbracket_{\sigma}, u=C\left[u_{1}, \ldots, u_{n}\right]$ where $C$ is the canonical context of $u$. For every $i$, let us split up both $e$ and $C$ :

- either there exists $p_{i}$ and $C_{i} \leqslant C$ such that $\left.C_{i}\left[u_{i}\right] \in \llbracket e\right|_{p_{i}} \|_{\sigma}$ and $\left.e\right|_{p_{i}}=e^{\prime \prime} \cap Y$. In particular, $C_{i}\left[u_{i}\right] \in \llbracket Y \rrbracket_{\sigma}$, thus by hypothesis, $C_{i}\left[u_{i}\right]=C_{i}\left[u_{i}\right] \downarrow$. By construction of the context, $C_{i}\left[u_{i}\right] \downarrow=C_{i}\left[u_{i} \downarrow\right]$ thus $u_{i}=u_{i} \downarrow$.
- or there exists $p_{i}$ such that $\left.u_{i} \in \llbracket e\right|_{p_{i}} \|_{\sigma}$. Since $e=\sigma^{0}\left(e_{1}\right)$ for some $e_{1}$ occurring in $S_{N B}$, we have to consider again two cases:
- either $p_{i}$ is not a path in $e_{1}$, i.e., there exists $q_{i}\left\langle p_{i}\right.$ such that $\left.e_{1}\right|_{q_{i}}=X$ where $X$ is a basic variable and $\left.e\right|_{q_{i}} \in \sigma^{0}(X)$. Thus $\left.u\right|_{q_{i}} \in \sigma^{0}(X) \subseteq R$ and $\left.u\right|_{q_{i}}=C_{i}\left[u_{i}\right]$. Since $\left.u\right|_{q_{i}} \in R, C_{i}\left[u_{i}\right]=C_{i}\left[u_{i}\right] \downarrow=C_{i}\left[u_{i} \downarrow\right]$, thus $u_{i}=u_{i} \downarrow$.
- either $p_{i}$ is a path in $e_{1}$ : Since the function symbols of $u_{i}$ do not occur in $S_{N B}$ and $\left.u_{i} \in \llbracket e\right|_{p_{i}} \rrbracket_{\sigma},\left.e_{1}\right|_{p_{i}}$ is necessarily an intersection of variables: $\left.e_{1}\right|_{p_{i}}=X_{1} \cap \ldots \cap X_{n}$. If one of the variable, say $X_{1}$, is basic then $\sigma^{0}\left(e_{1}\right) \in \sigma^{0}(X)$ and we conclude like above. Else $u_{i} \in \llbracket X_{1} \cap \ldots \cap X_{n} \rrbracket_{\sigma}$ and we conclude by hypothesis.
We are now ready to end the proof of Lemma 40 by induction on the fixed point of our deduction system: assume that for every $n^{\prime}\left\langle n\right.$, then $u \in \llbracket X \rrbracket_{T^{n^{\prime}}\left(\sigma_{\emptyset}\right)}$ implies $u=u \downarrow$. and let us show that $u \in \llbracket X \rrbracket_{T^{n}\left(\sigma_{⿹ 勹}\right)}$ implies $u=u \downarrow$. Let $\phi \Rightarrow e \subseteq e^{\prime} \in \sigma^{0}\left(S_{N B}\right)$ be the clause which generated $u$ (we assume w.l.o.g. that $e^{\prime}$ does not contain intersection symbols like in Section 5.1.2):
if $e^{\prime}=X$ then $u \in \llbracket e \rrbracket_{T^{n-1}\left(\sigma_{\emptyset}\right)}$ and we conclude by induction.
if $e^{\prime}=f\left(X_{1}, \ldots, X_{n}\right)$ and $X=X_{i}$ then there exists $v \in \llbracket e \rrbracket_{T^{n-1}\left(\sigma_{\emptyset}\right)}$ such that $v=f(\ldots$,
$u, \ldots)$. Since $f$ does not occurs in $\mathcal{R}(X)$ for $X$ basic variable, $v$ can not be accepted in any state $X$ where is a basic variable, thus $v \downarrow=f(\ldots, u \downarrow, \ldots)$. By induction and the property we have just demonstrated, we know that $v=v \downarrow$ which implies $u=u \downarrow$.

We are now ready to prove Lemma 39. $\sigma_{\approx}^{\prime}$ is a solution of $S_{B}$, by definition of $\approx$ and $\sigma^{\prime}$. Let us consider a clause $\phi \Rightarrow e \subseteq e^{\prime}$, which does not belong to $S_{B}$. Since $\phi$ only contains atomic formulas of the form nonempty $(X), \sigma^{\prime} \vDash \phi$ iff $\sigma_{\approx}^{\prime} \vDash \phi$. Assume that $\sigma^{\prime} \vDash \phi$ (if it is not the case, then $\sigma_{\approx}^{\prime}$ trivially satisfies the clause).

By induction on $e$, any term $t \in \llbracket e \rrbracket_{\sigma^{\prime}}$ is equivalent, modulo $\approx$ to a term $u \in \llbracket e \rrbracket_{\sigma^{\prime}}$ such that $u \in \llbracket e^{\prime} \rrbracket_{\sigma^{\prime}}$. Indeed, if $e$ is a variable, this is true by definition of $\approx$. If $e=e_{1} \cap e_{2}$, then, by induction, there exists $u_{1} \in \llbracket e_{1} \rrbracket_{\sigma^{\prime}}$ and $u_{2} \in \llbracket e_{2} \rrbracket_{\sigma^{\prime}}$ such that $t \approx u_{1}$ and $t \approx u_{2}$. By Lemma 40, $u_{1}=u_{1} \downarrow, u_{2}=u_{2} \downarrow$, thus $u_{1}=u_{2}$ and $u_{1} \in \llbracket e \rrbracket_{\sigma^{\prime}}$. If $e=f\left(e_{1}, \ldots, e_{n}\right)$ (last case), then $t=f\left(t_{1}, \ldots, t_{n}\right), t_{i} \in \llbracket e_{i} \rrbracket_{\sigma_{\approx}^{\prime}}$. By induction, there exists $u_{i} \approx t_{i}$ such that $u_{i} \in \llbracket e_{i} \rrbracket_{\sigma^{\prime}}$, thus $u \xlongequal{\text { def }} f\left(u_{1}, \ldots, u_{n}\right) \in \llbracket e \rrbracket_{\sigma^{\prime}}$ and $u \approx t$.

Now, either $e^{\prime}$ is the intersection of the variables $X_{i}$, in which case there exists $u_{i} \in \llbracket X_{i} \rrbracket_{\sigma^{\prime}}$ for every $i$ such that $u_{i} \approx t$, hence $t \in \llbracket X_{i} \rrbracket_{\sigma_{\approx}^{\prime}}$ for every $i$, or else $e^{\prime}=f\left(X_{1}, \ldots, X_{n}\right)$. In the latter case, $f \notin \mathcal{O F}$ and therefore $t=f\left(t_{1}, \ldots, t_{n}\right), u=f\left(u_{1}, \ldots, u_{n}\right)$ with $t_{i} \approx u_{i}$ and $u_{i} \in \llbracket X_{i} \rrbracket_{\sigma^{\prime}}$. Again, this implies that $t_{i} \in \llbracket X_{i} \rrbracket_{\sigma_{\approx}^{\prime}}$ for every $i$, hence $t \in \llbracket e^{\prime} \rrbracket_{\sigma_{\approx}^{\prime}}$.

Thanks to Lemma 39, and as far as satisfiability is concerned, we can now restrict our attention to the constraint $\sigma_{0}\left(S_{N B}\right)$ in which there is no longer any basic variable.

From the cryptographic protocols point of view, if we assume that the set of principal names correspond in the set constraint formalism to a basic variable $N$ (which is the case in all formalism we know), Lemma 39 shows that, if there is an attack, then there is an attack with a bounded number of principals. The bound is given by the cardinal of $\sigma_{0}(N)$. Again, in any description of principals that we can think of, $\sigma_{0}(N)$ will contain at most two elements. Then, the result shows that, if there is an attack, then there is an attack involving two distinct principals only (a honest one and a dishonest one).

### 5.2.4. Complexity issues in eliminating the basic variables

Before going any further, let us comment on the complexity of $\sigma_{0}\left(S_{N B}\right)$ with respect to $S$.
First consider the computation of $\sigma_{m}$. Following Theorem 11, the computation of $\mathcal{A}_{m}$ requires deterministic exponential time in general, since it is quite easy to encode the emptiness problem for the intersection of tree automata [28].

On the other hand, we want to point out a particular case which can be relevant to the application to cryptographic protocols. $S_{B}$ often satisfies additional properties, which we describe below.

For every basic variable $X$, let $\operatorname{Head}(X)$ be the least set of (one-way) function symbols such that

- if $f(\ldots) \subseteq X$ is an inclusion of $\mathcal{R}(X)$, then $f \in \operatorname{Head}(X)$
- if $X_{1} \cap \ldots \cap X_{n} \subseteq X \in \mathcal{R}(X)$, then $\operatorname{Head}\left(X_{1}\right) \cap \ldots \cap \operatorname{Head}\left(X_{n}\right) \subseteq \operatorname{Head}(X)$

Lemma 41. If for every two basic variables $X, Y$, either $X \subseteq Y \in S_{B}$ or $Y \subseteq X \in$ $S_{B}$ or $\operatorname{Head}(X) \cap \operatorname{Head}(Y)=\emptyset$, then it is possible to compute in polynomial time a finite tree automaton whose states contain the basic variables and which accepts $\sigma_{m}(X)$ in state $X$.

Proof. We can compute $\operatorname{Head}(X)$ in polynomial time. Then, while saturating $S_{B}$, we replace every intersection with either $\emptyset$ or the largest variable, preventing the combinatorial explosion.

A second source of complexity comes from the computation of an ET-constraint out of $\sigma_{0}\left(S_{N B}\right)$ : eliminating the disjunctions may lead to an exponential blow-up in general. However, with the same hypothesis as above, the cardinal of $\sigma_{0}(X)$ is smaller or equal to the number of inclusions of the form $Y \subseteq X$. In particular, in our running example, only $\sigma_{0}(A)$ contains more than one element: $\sigma_{0}(A)=\{h a(0), d a(0)\}$. In addition, if we assume that there is no inclusion between basic variables as it was the case in our previous version [10], then $\sigma_{0}(X)$ assigns each basic variable either the empty set of a singleton set and therefore $\sigma_{0}\left(S_{N B}\right)$ is smaller in size than $S$ itself.

### 5.2.5. Simplifying again the expressions

The goal of this section is to achieve further simplifications. In particular we show that, after eliminating the basic variables, we can get rid of nested constructions $f^{c}(\ldots)$.

Thanks to Lemma 39, we can now restrict our attention to the constraint $\sigma_{0}\left(S_{N B}\right)$. In such a constraint, there is no longer any basic variable, which allows for several simplifications. First, we can abstract one the left side of inclusions such that the inclusions are $e \subseteq e^{\prime}$ where


Fig. 6. Simplification rules.
$e$ is either an intersection of variables or an expression $f^{c}(\vec{e})$ in which, at any position which is not a strict prefix of a position checked by $c$, there is a variable.
Formally, we use the following rules, assuming that $n \geqslant 2$ and $p$ is not the root position: and that $p$ is not a strict prefix of any path checked (higher) in $C$ :

$$
\begin{aligned}
& \left(\mathrm{A}_{4}\right) \quad \phi \Rightarrow C\left[f^{c}(\vec{e})\right]_{p} \subseteq e^{\prime} \rightarrow(\exists X) \phi \Rightarrow C[X]_{p} \subseteq e^{\prime}, f^{c}(\vec{e}) \subseteq X \\
& \text { ( } \left.\mathrm{A}_{5}\right) \phi \Rightarrow C\left[e_{1} \cap \ldots \cap e_{n}\right]_{p} \subseteq e^{\prime} \rightarrow(\exists X) \phi \Rightarrow C[X]_{p} \subseteq e^{\prime}, e_{1} \cap \ldots \cap e_{n} \subseteq X
\end{aligned}
$$

In these rules, $X$ is a new variable: we assume that there is no capture. The following lemma is a consequence of the definitions:

Lemma 42. Applying abstraction does terminate on any constraint $S$, resulting in a constraint $S^{\prime}$ such that the solutions of $S$ are the restrictions of solutions of $S^{\prime}$ to the free variables of $S$. Moreover, if $S$ is an ET-constraint, then so is $S^{\prime}$ and if every expression is normal in $S$, then every expression is normal in $S^{\prime}$.

In addition, in the equality tests, if $f^{c \wedge p=q}(\vec{e})$ is an expression such that the subexpression at position $p$ (or $q$ ) is basic then the expressions at positions $p, q$ must contain the same ground term. This is also sufficient: the equality test $p=q$ can then be removed if the appropriate inclusions $t \subseteq X$ ( $t$ is ground) are added. Formally, we use the rules displayed in Fig. 6.

Lemma 43. The simplification rules displayed in Fig. 6 are terminating. If S is an ETconstraint in which all expressions are normal, then the result $S^{\prime}$ of simplifying and abstracting $\sigma_{0}\left(S_{N B}\right)$ is an ET-constraint in which all expressions are normal and which is satisfiable iff $S$ is satisfiable. Moreover, in any expression $f^{c}\left(e_{1}, \ldots, e_{n}\right)$ occurring in $S^{\prime}$, $e_{1}, \ldots, e_{n}$ do not contain a construction $g^{c^{\prime}}(\ldots)$.

Proof. The correctness of the rules is a routine verification. let us only consider the rule $\left(N_{9}\right)$. If $\sigma \quad \not \vDash \phi$, it suffices to assign $T(\mathcal{F})$ to $Y$ and both sides are satisfied by $\sigma$. If $\sigma \vDash \phi$ and $\llbracket e\left[t \cap e_{0}\right] \rrbracket_{\sigma} \subseteq \llbracket e^{\prime} \rrbracket_{\sigma}$, then extending $\sigma$ with $Y \mapsto \llbracket t \cap e_{0} \rrbracket_{\sigma}$ we get a solution of the right hand side:

- either $\llbracket t \cap e_{0} \rrbracket_{\sigma}=\emptyset$ and this is straightforward,
- or else $\llbracket t \cap e_{0} \rrbracket_{\sigma}=\{t\}$, since $t$ is a ground term, in which case $\llbracket e\left[t \cap e_{0}\right] \rrbracket_{\sigma}=\llbracket e[t] \rrbracket_{\sigma}$.

Conversely, if $\sigma$ is a solution of the right hand side, either $\llbracket Y \rrbracket_{\sigma}$ is empty, which means that $t \notin \llbracket e_{0} \rrbracket_{\sigma}$ and the left hand side is satisfied by $\sigma$ or $\llbracket Y \rrbracket_{\sigma}=\{t\}$, in which case $\llbracket e\left[t \cap e_{0}\right] \rrbracket_{\sigma}=\llbracket e[t] \rrbracket_{\sigma} \subseteq \llbracket e^{\prime} \rrbracket_{\sigma}$.

Thanks to Lemma 39, it only remains to show that all expressions are normal in $S^{\prime}$ whenever all expressions are normal in $S$ and that, moreover, there is no longer any nested equality test.
For every expression $f^{c}(\vec{e}), P(c) \subseteq \Pi\left(f^{c}(\vec{e})\right)$. Only the case of $\left(N_{9}\right)$ is not trivial. The property is ensured by the side condition.
Condition 5 is satisfied. $\sigma^{0}$ may replace variables with ground terms, hence replace expressions $g^{c}(\vec{e}) \cap X_{1} \cap \ldots X_{n}$ with $g^{c}(\vec{e}) \cap t_{1} \cap \ldots \cap t_{n}$. However, each $t_{i}$ is either a variable or is ground. If at least one of them is ground, we can apply either ( $N_{8}$ ) or ( $N_{9}$ ).
Condition 3 is satisfied. There are two situations in which property 3 is not trivially preserved: first when, while removing disjunctions in $\sigma^{0}\left(S_{N B}\right)$, we do not keep the consistency with equality tests: in an expression $f^{p=q \wedge c}(\vec{e}), X$ has been replaced with $t$ at a position $p \cdot p_{1}$, while $X$ has been replaced with $u$ at the position $q \cdot p_{1}$. This case is handled by rule $\left(N_{11}\right)$.
The second situation in which condition 3 may not be preserved is when we apply the rule $\left(N_{9}\right)$. However, in this case, applying the rule to all identical expressions restores condition 3.

Condition 1 is satisfied. Thanks to $\left(N_{10}\right)$ and $\left(N_{11}\right)$, we cannot have $p \sim_{c} q,\left.e\right|_{p}$ ground and $\left.e\right|_{q}$ not ground. So, a repeated application of $\left(N_{10}\right)$ consists in removing an equivalence class, which preserves condition 1 , thanks to lemma 72 . The rules other than $\left(N_{10}\right)$ trivially preserve condition 1.

Conditions 4, 2 and 6 are satisfied. Again, the only rule to be considered is $\left(N_{10}\right)$ since this is the only rule in the set which modifies the tests without removing them entirely. As above, since its repeated application removes a class, properties 4, 2 and 6 are preserved.
There is no nested test. Assume that there are nested tests: $\left.f^{c}(\vec{e})\right|_{p}=g^{c^{\prime}}\left(\overrightarrow{e^{\prime}}\right) \cap e^{\prime \prime}$.
First, if there are $p_{1} \sim_{c} p_{2}$ such that $p$ is a prefix of $p_{1}$, by properties 6 and 1 , for every $p^{\prime} \sim_{c^{\prime}} q^{\prime},\left.g^{c^{\prime}}\left(\overrightarrow{e^{\prime}}\right)\right|_{p^{\prime}}$ must be a basic expression, hence a ground term. Then, the rules $\left(N_{8}\right),\left(N_{9}\right),\left(N_{10}\right),\left(N_{11}\right)$ ensure that $c^{\prime}$ is empty.
On the other hand, if this is not the case and if there are $p_{1} \sim_{c} p_{2}$ such that $p_{1}$ shares a non-trivial prefix with $p$, then, by property $1,\left.f^{c}(\vec{e})\right|_{p}$ must be a basic expression, hence a ground term. In this last case $c^{\prime}$ must be empty again.
Remains only the case in which, for every $p_{1} \sim_{c} p_{2}, p_{1}$ is either incomparable with $p$ or a prefix of $p$. Then, Abstract can be applied, which contradicts the hypothesis on $S_{N B}$.

We use a final simplification rule, abstracting away some more expressions:

$$
\begin{aligned}
\left(N_{12}\right) \phi \Rightarrow & e\left[X \cap g\left(e_{1}^{\prime}, \ldots, e_{m}^{\prime}\right)\right]_{p} \subseteq e^{\prime} \rightarrow\left(\exists Y_{1}, \ldots, Y_{m}\right) \\
& X \cap g(T, \ldots, T) \subseteq g\left(Y_{1}, \ldots, Y_{m}\right) \\
& \phi \Rightarrow e\left[g\left(Y_{1} \cap e_{1}^{\prime}, \ldots, Y_{m} \cap e_{m}^{\prime}\right)\right]_{p} \subseteq e^{\prime}
\end{aligned}
$$

If $p$ is nonempty.
The rule assumes that there is a variable $T$ which captures all terms (this is easy to define).
The correctness of the rule as well as the preservation of all properties is quite straightforward. Let us now inspect the constraints we have still to consider.

Definition 44. The SET-constraints (simplified equality tests constraints) are a subclass of ET-constraints in which, for every clause

$$
\operatorname{nonempty}\left(e_{1}^{\prime}\right), \ldots, \operatorname{nonempty}\left(e_{m}^{\prime}\right) \Rightarrow e \subseteq e^{\prime}
$$

(resp. nonempty $\left(e_{1}^{\prime}\right), \ldots$, nonempty $\left(e_{m}^{\prime}\right) \Rightarrow$ false).

1. each of $e, e_{1}^{\prime}, \ldots, e_{m}^{\prime}$ is either an intersection of variables or an expression $f^{c}\left(e_{1}, \ldots, e_{n}\right)$ such that $e_{1}, \ldots, e_{n}$ do not contain any equality tests nor expressions $X \cap g(\ldots)$.
2. For every $p \in \Pi(e)$, except the root, either $p$ is a strict prefix of some $q \in P(c)$, or else $\left.e\right|_{p}$ is a (basic) ground term or $p \in P(c)$ and $\left.e\right|_{p}$ is an intersection of variables.
3. If $p \sim_{c} q,\left.e\right|_{p}=\left.e\right|_{q}$.
4. $e^{\prime}$ is either a variable or an expression $f^{\prime}\left(X_{1}, \ldots, X_{n}\right)$ where $X_{1}, \ldots, X_{n}$ are variables.

Lemma 45. The simplification rules displayed in Fig. 6 are terminating. If S is a SETconstraint in which all expressions are normal, then the result $S^{\prime}$ of simplifying and abstracting $\sigma_{0}\left(S_{N B}\right)$ is an ET-constraint

Proof. Let us show that $S^{\prime}$ verifies the four conditions of SET-constraints.

1. Assume $e$ or one of the $e_{i}^{\prime}$ is of the form $f^{c}\left(e_{1}, \ldots, e_{n}\right)$ and that one of the $e_{j}$ contains an equality test $c_{j}$. Then, by abstracting, it must be that $c$ overlaps $c_{j}$ which is not possible since $S^{\prime}$ is normalized.
2. After abstraction, if $p$ is not a strict prefix of some $q \in P(c)$ and $p$ is in $\Pi(e)$, then $p \in P(c)$ and $\left.e\right|_{p}$ is a variable. After simplifying, if $p$ is not a strict prefix of some $q \in P(c)$, then $\left.e\right|_{p}$ is either a variable or a ground term.
Conditions 3 and 4 are consequences of the definitions.

### 5.2.6. Deduction rules

Now, we are ready to apply the deduction rules given in Fig. 7. $c \downarrow_{i}$ is defined by $\left(c \wedge c^{\prime}\right) \downarrow_{i} \stackrel{\text { def }}{=} c \downarrow_{i} \wedge c^{\prime} \downarrow_{i},(i \cdot p=i \cdot q) \downarrow_{i}=$ def $p=q$ and $(j \cdot p=q) \downarrow_{i} \stackrel{\text { def }}{=} T$ when $i \neq j$. $e^{c}$ is the expression in which the top symbol of $e$ is constrained by $c$. (It is used only in a context where $e$ must be headed with a function symbol or $c=\mathrm{T}$ ). Finally, $X$ denotes a variable in these rules.

Lemma 46. The inference rules in Fig. 7 are correct: the new constraint is a consequence of the previous ones.

$$
\begin{aligned}
& \text { Transitivity } \frac{\phi_{1} \Rightarrow e_{1} \subseteq X \quad \phi_{2} \Rightarrow X \subseteq e_{2}}{\phi_{1}, \phi_{2} \Rightarrow e_{1} \subseteq e_{2}} \\
& \phi \Rightarrow X \cap e_{1} \subseteq e_{1}^{\prime} \quad \phi^{\prime} \Rightarrow e_{2} \subseteq X \\
& \text { Compatibility } \\
& \phi, \phi^{\prime} \Rightarrow e_{1} \cap e_{2} \subseteq e_{1}^{\prime} \\
& \text { If both } e_{1} \text { and } e_{2} \text { are intersections of variables } \\
& \text { Clash } \quad \phi \Rightarrow f(\vec{e}) \subseteq g\left(\overrightarrow{e^{\prime}}\right) \\
& \text { if } f \neq g \\
& \phi \Rightarrow \text { false } \\
& \phi \Rightarrow f^{c}\left(e_{1}, \ldots, e_{n}\right) \subseteq f\left(e_{1}^{\prime}, \ldots, e_{n}^{\prime}\right) \\
& \text { Projection } \\
& \overline{\phi, \operatorname{nonempty}\left(f^{c}\left(e_{1}, \ldots, e_{n}\right)\right) \Rightarrow e_{i}^{c \downarrow_{i}} \subseteq e_{i}^{\prime}} \\
& \phi_{1} \Rightarrow f^{c_{1}}\left(e_{1}^{1}, \ldots, e_{n}^{1}\right) \subseteq X_{1} \\
& \phi_{k} \Rightarrow f^{c_{k}}\left(e_{1}^{k}, \ldots, e_{n}^{k}\right) \subseteq X_{k} \\
& \phi \Rightarrow X_{1} \cap \ldots \cap X_{k} \subseteq e \\
& \exists X_{1}^{1}, \ldots X_{1}^{n}, \ldots, X_{k}^{1}, \ldots, X_{k}^{n} \text {. } \\
& \phi \Rightarrow f\left(X_{1}^{1} \cap \ldots \cap X_{k}^{1}, \ldots, X_{1}^{n} \cap \ldots \cap X_{k}^{n}\right) \subseteq e \\
& \phi_{1} \Rightarrow f^{c_{1}}\left(e_{1}^{1}, \ldots, e_{n}^{1}\right) \subseteq f\left(X_{1}^{1}, \ldots, X_{n}^{1}\right) \\
& \phi_{k} \Rightarrow f^{c_{k}}\left(e_{1}^{k}, \ldots, e_{n}^{k}\right) \subseteq f\left(X_{1}^{k}, \ldots, X_{n}^{k}\right)
\end{aligned}
$$

The clause $\phi \Rightarrow f\left(X_{1}^{1} \cap \ldots \cap X_{k}^{1}, \ldots, X_{1}^{n} \cap \ldots \cap X_{k}^{n}\right) \subseteq e$ in the conclusion of Deduction is marked so that it cannot be used as a premisse of Deduction. In addition, if $\phi_{1} \Rightarrow e_{1} \subseteq X$ is a marked clause, then for every clause $\phi_{2} \Rightarrow X \subseteq e_{2}$, then clause $\phi_{1}, \phi_{2} \Rightarrow e_{1} \subseteq e_{2}$ is also a marked clause.

Fig. 7. The saturation rules.

Proof. Only Projection and Deduction are not trivially correct. Let us start with Projection.
We want to prove that, if $\sigma$ is a solution of $\phi \Rightarrow f^{c}\left(e_{1}, \ldots, e_{n}\right) \subseteq f\left(e_{1}^{\prime}, \ldots, e_{n}^{\prime}\right)$, then $\sigma$ is a solution of $\phi, \operatorname{nonempty}\left(f^{c}\left(e_{1}, \ldots, e_{n}\right)\right) \Rightarrow e_{i}^{c \downarrow i} \subseteq e_{i}^{\prime}$.

Assume $\sigma \vDash \phi$, $\operatorname{nonempty}\left(f^{c}\left(e_{1}, \ldots, e_{n}\right)\right)$. Then there is an $u \in \llbracket f^{c}\left(e_{1}, \ldots, e_{n}\right) \rrbracket_{\sigma}$ and $\llbracket f^{c}\left(e_{1}, \ldots, e_{n}\right) \rrbracket_{\sigma} \subseteq \llbracket f\left(e_{1}^{\prime}, \ldots, e_{n}^{\prime}\right) \rrbracket_{\sigma}$.

Let $t \in \llbracket e_{i}^{c \downarrow i} \rrbracket_{\sigma}$. We build $v$ as follows: $v$ is the term $u$ in which

- $\left.u\right|_{i}$ is replaced with $t$,
- for every $i \cdot p \sim_{c} j \cdot q$ with $i \neq j,\left.u\right|_{j \cdot q}$ is replaced with $\left.t\right|_{p}$.

Let us show that $v \in \llbracket f^{c}\left(e_{1}, \ldots, e_{n}\right) \rrbracket_{\sigma}$.

- First, $v \in \llbracket f\left(e_{1}, \ldots, e_{n}\right) \rrbracket_{\sigma}$ : let $v=f\left(v_{1}, \ldots, v_{n}\right) . v_{i}=t \in \llbracket e_{i} \rrbracket_{\sigma}$ and we prove by induction on $|c|$ that, for every $i \neq j, v_{j} \in \llbracket e_{j} \rrbracket$.
- If $c$ is empty, or if $c$ does not contain any equation $i \cdot p=j \cdot q$, then $v_{j}=u_{j}$ and therefore $v_{j} \in \llbracket e_{j} \rrbracket_{\sigma}$.
- If $i \cdot p \sim_{c} j \cdot q$, by induction hypothesis, $w_{j}=v_{j}\left[\left.u\right|_{j \cdot q}\right]_{q} \in \llbracket e_{j} \rrbracket_{\sigma}$. Moreover, by property $3,\left.e_{j}\right|_{q}=\left.e_{i}\right|_{p}$, hence $\left.\left.t\right|_{p} \in \llbracket e_{j}\right|_{q} \rrbracket_{\sigma}$. Now, since we assumed in the condition of the projection rule, that there is no intersection symbol along the path $j \cdot q, v_{j} \in \llbracket e_{j} \rrbracket_{\sigma}$.
- We have to prove now that $v \vDash c$ :
- For the tests $i \cdot p=i \cdot q \in c, v \vDash i \cdot p=i \cdot q$ follows from $t \vDash p=q$ (since $t \vDash c \downarrow i$ ).
- The tests $i \cdot p=j \cdot q$ with $i \neq j$ are satisfied by construction.
- For the tests $j_{1} \cdot p=j_{2} \cdot q$, either there is a $r$ such that $j_{1} \cdot p \sim_{c} i \cdot r$ and we are back to the previous case, or else $\left.u\right|_{j_{1} \cdot p}=\left.v\right|_{j_{1} \cdot p}$ and $\left.u\right|_{j_{2} \cdot q}=\left.v\right|_{j_{2} \cdot q}$, which implies $v \vDash j_{1} \cdot p=j_{2} \cdot q$ since $u \vDash j_{1} \cdot p=j_{2} \cdot q$.
Now, $v \in \llbracket f^{c}\left(e_{1}, \ldots, e_{n}\right) \rrbracket_{\sigma}$ implies that $v \in \llbracket f\left(e_{1}^{\prime}, \ldots, e_{n}^{\prime}\right) \rrbracket_{\sigma}$, hence $t \in \llbracket e_{i}^{\prime} \rrbracket_{\sigma}$.
Now consider Deduction. The rule is actually a combination of several rules which are all correct: for every $i$, we introduce

$$
\text { (1) } X_{i} \cap f(T, \ldots, T)=f\left(X_{1}^{i}, \ldots, X_{n}^{i}\right) \text {. }
$$

Then, we may intersect both sides of $X_{1} \cap \ldots \cap X_{k} \subseteq e$ with $f(T, \ldots, T)$ and use a compatibility. We get

$$
\phi \Rightarrow f\left(X_{1}^{1}, \ldots, X_{n}^{1}\right) \cap \ldots \cap f\left(X_{1}^{k}, \ldots, X_{n}^{k}\right) \subseteq e
$$

Normalizing the left hand side, we get the clause

$$
\phi \Rightarrow f\left(X_{1}^{1} \cap \ldots \cap X_{k}^{1}, \ldots, X_{1}^{n} \cap \ldots \cap X_{k}^{n}\right) \subseteq e
$$

Now, for every $i$, from (1) and the inclusion $\phi_{i} \Rightarrow f^{c_{i}}\left(e_{1}^{i}, \ldots, e_{n}^{i}\right) \subseteq X_{i}$, intersecting again both sides with $f(\mathrm{~T}, \ldots, \mathrm{~T})$, we deduce by transitivity and since $f^{c}(\ldots) \subseteq f(\mathrm{~T}, \ldots, \mathrm{~T})$ :

$$
\phi_{i} \Rightarrow f^{c_{i}}\left(e_{1}^{i}, \ldots, e_{n}^{i}\right) \subseteq f\left(X_{1}^{i}, \ldots, X_{n}^{i}\right)
$$

Lemma 47. Every transformation rule transforms a SET-constraint into a SET-constraint.
Proof. Only the projection rule has to be considered: we need to check that $e_{i}^{c \downarrow i}$ satisfies the conditions of SET-constraints, which follows from the fact that $p \sim_{c \downarrow_{i}} q$ iff $p \cdot i \sim_{c} q \cdot i$.

Now, we consider the termination problem. The main problem is to control the creation of new variables.

Lemma 48. The number of variables created during the saturation procedure can be bounded by $|S|_{t} \times a$.

Proof. Only Deduction introduces new variables. It is simpler to see the Deduction rule as a variable introduction (rule (1) in the proof of Lemma 46) combined with other deduction rules, which preserve the semantics and do not introduce variables. As far as variables creation is concerned, we can, w.l.o.g, assume that the conclusion of Deduction are the clauses $X_{i} \cap f(\mathrm{~T}, \ldots, \mathrm{~T}) \subseteq f\left(X_{1}^{i}, \ldots, X_{n}^{i}\right)$. In particular, if the rule is applied twice with the same variable $X_{i}$ and the same function symbol $f$, we can use the same variables $X_{j}^{i}$.
The main problem is that these new variables $X_{j}^{i}$ may trigger again the introduction of new variables. However, if we trace the origin of such variables, we observe that introducing the $X_{j}^{i}$ is subject to the presence of a constraint $\phi_{i} \Rightarrow f^{c_{i}}(\ldots) \subseteq X_{i}$. Now, if later a constraint $\phi_{i}^{\prime} \Rightarrow f^{c_{i}^{\prime}}(\ldots) \subseteq X_{j}^{i}$ triggers the introduction of new variables again, the expression $f^{c_{i}^{\prime}}(\ldots)$ must be the projection of some expression occurring at the previous level. And since we can only perform a bounded number of projections on a given expression, we can bound the number of successive variables generation. Let us formalize this.
Let us associate first with each variable a level: the variables occurring in the original constraint have level 0 , and, each time we introduce new variables with:
(1) $X \cap f(T, \ldots, T) \subseteq f\left(X_{1}, \ldots, X_{n}\right)$
the level of every variable $X_{i}$ is one plus the level of $X$.
We prove, by induction on the level $m$ of a variable $X$, that, if (1) is applied to $X$, then there is an expression $e$ in the original set constraint and clauses $\phi \Rightarrow e^{\prime} \subseteq X^{\prime}, \phi_{1} \Rightarrow X^{\prime} \subseteq X_{1}$, $\ldots, \phi_{n+1} \Rightarrow X_{n} \subseteq X$ such that $e^{\prime}$ is obtained by at least $m$ successive projections of $e$ (we say that $e^{\prime}$ is a projection of $e$ if $e=f^{c}\left(e_{1}, \ldots, e_{n}\right)$ and $e^{\prime}=e_{i}^{c \downarrow i}$ for some $\left.i\right)$.

When $m=0$, observe that clauses $\phi \Rightarrow f^{c}(\ldots) \subseteq X^{\prime}$ are either in the original set constraint, or obtained by projection, or obtained by Transitivity or obtained by Deduction itself. In case Transitivity has been applied, there exist another clause $\phi^{\prime} \Rightarrow$ $f^{c}(\ldots) \subseteq X^{\prime \prime}$ and a clause $\phi^{\prime \prime} \Rightarrow X^{\prime \prime} \subseteq X^{\prime}$. Thus, by a simple induction, there exist a clause $\phi^{\prime} \Rightarrow f^{c}(\ldots) \subseteq X^{\prime \prime}$ and clauses $\phi_{1} \Rightarrow X^{\prime \prime} \subseteq X_{1}, \ldots, \phi_{n+1} \Rightarrow X_{n} \subseteq X^{\prime}$ such that $\phi^{\prime} \Rightarrow f^{c}(\ldots) \subseteq X^{\prime \prime}$ has been obtained by projection or by Deduction itself. However, in the latter case, we explicitly prevented using the resulting clause as a premisse of

## Deduction.

When $m>0$, observe that the variables $X_{j}^{i}$ created by Deduction appear in only one clause on the right of an inclusion: the clause $\phi_{i} \Rightarrow f^{c_{i}}\left(e_{1}^{i}, \ldots, e_{n}^{i}\right) \subseteq f\left(X_{1}^{i}, \ldots, X_{n}^{i}\right)$. Only Projection, applied to this clause, may produce a clause in which $X_{j}^{i}$ occurs on the right of an inclusion: there exists a clause $\phi_{i} \Rightarrow e_{j}^{i_{i} \downarrow j} \subseteq X_{j}^{i}$ such that $e_{j}^{i}{ }_{j} \downarrow j$ is the projection of $f^{c_{i}}\left(e_{1}^{i}, \ldots, e_{n}^{i}\right)$. Then we apply the induction hypothesis to $\phi_{i} \Rightarrow f^{c_{i}}\left(e_{1}^{i}, \ldots, e_{n}^{i}\right) \subseteq X_{i}$ since $X_{i}$ is of level $m-1$.

Now, how many expressions $f^{c}(\ldots)$ can be derived by projection from a given expression ? Note first that no projection can be applied to a ground (basic) expression since one-way symbols do not occur on the right of inclusions. Then, by definition of expressions occurring in a SET-constraint, the number of expressions $f^{c}(\ldots)$ which can be derived by projection from an expression $g^{c^{\prime}}(\ldots)$ is the number of strict prefixes of positions in $P\left(c^{\prime}\right)$. It follows that the number of new variables is bounded by $|S|_{t} \times a$.

Lemma 49. The rules of Fig. 7 are terminating: a fixed point is reached after finitely many steps (at most $\mathrm{O}\left(|S|^{a} \times 2^{(a+1)|S|}\right)$ where $a$ is the maximal arity of a function symbol and $|S|=|S|_{t}+|S|_{\mathcal{F}}$ is the size of the original constraint.

Proof. We are going to show that only a finite number of distinct clauses can be generated by the rules.

As we have seen in the proof of lemma 48, the number of distinct expressions $f^{c}(\ldots)$ occurring on the left of an inclusion is bounded by $|S|_{t}$, plus the number of times Deduction is applied, which is itself bounded by $|S|_{t} \times a$ thanks to Lemma 48. The other left sides of inclusions are intersection of variables, hence there are at most $2^{a|S|}{ }^{a}+\left|V_{S}\right|$ such possible left hand sides, thanks to Lemma 48.

The right sides of inclusions are variables or a function symbol applied to variables, which gives a bound of $a|S|_{t}+\left|V_{S}\right|+|\mathcal{F}| \times\left(a|S|_{t}+\left|V_{S}\right|\right)^{a}$.

Finally, we have to analyze the possible preconditions. They are conjunctions of - nonempty $(X)$ where $X \in V_{S}$,

- nonempty $\left(f^{c}\left(e_{1}, \ldots, e_{n}\right)\right)$ where $f^{c}\left(e_{1}, \ldots, e_{n}\right)$ occurs as a left member of an inclusion constraint.
Thanks to Lemma 48, this gives the following bound for the number of possible distinct preconditions:

$$
2^{\left|V_{S}\right|+|S|_{t}} .
$$

Now, putting everything together, at most

$$
\begin{aligned}
& \left((a+1) \times|S|_{t}+2^{a|S|_{t}+\left|V_{S}\right|}\right) \times\left(a|S|_{t}+\left|V_{S}\right|+|\mathcal{F}| \times\left(a|S|_{t}+\left|V_{S}\right|\right)^{a}\right) \\
& \quad \times 2^{\left|V_{S}\right|++\mid S_{t}}
\end{aligned}
$$

distinct clauses can be generated. Since $|S|_{t}+\left|V_{S}\right| \leqslant|S|$ and $a|S|_{t}+\left|V_{S}\right| \leqslant a \times|S|$, we get the bound $\mathrm{O}\left(|S|^{a} \times 2^{(a+1)|S|}\right)$.

If $S$ is an ET-constraint, let solved $(S)$ be the clauses $\phi \rightarrow a$ in $S$ such that either $a$ is false or else $a$ is an inclusion $f^{c}(\vec{e}) \subseteq X$ where $X$ is a variable.

As in [7], the following completeness result is obtained by inspecting each clause $C \in$ $S$ which is not in solved $(S)$, showing that, thanks to saturatedness, the least solution of solved $(S)$ is a solution of $C$. There are only some additional cases for nonflat constraints e.g. $f^{c}\left(X \cap g(\vec{e}), \overrightarrow{e^{\prime}}\right) \subseteq f\left(\overrightarrow{e^{\prime \prime}}\right)$.

Theorem 50. If $S$ is saturated, then either both $S$ and solved $(S)$ are unsatisfiable or else $S$ has a least solution, which is the least solution of solved $(S)$.

Proof. If solved $(S)$ is unsatisfiable, then $S$, which contains solved $(S)$, is unsatisfiable. Now, assume $\operatorname{solved}(S)$ is satisfiable and let $\alpha$ be its least solution. We show that $\alpha$ is a solution of $S$.

We prove, by induction on $n+\operatorname{size}(t)$ that, for every clause $\phi \Rightarrow e \subseteq e^{\prime}$ in $S$ such that $\alpha \vDash \phi$, and for every $t \in \llbracket e \rrbracket_{T_{\text {solved }(S)}^{n}}{ }^{(\varnothing)}$ (which we abbreviate $t \in \llbracket e \rrbracket_{n}$ ), $t \in \llbracket e^{\prime} \rrbracket_{\alpha}$.

The result will follow, by minimality of $\alpha$.

There are only three kinds of clauses $\phi \Rightarrow e \subseteq e^{\prime}$, which are possibly in $S$ and not in solved $(S)$. We study each of them ( $e^{\prime}$ is either a variable or an expression $f\left(X_{1}, \ldots, X_{n}\right)$ ). $\phi \Rightarrow X \subseteq g\left(Y_{1}, \ldots, Y_{m}\right), t \in \llbracket X \rrbracket_{n}$, hence there is a clause $\phi^{\prime} \Rightarrow e_{0} \subseteq X$ in solved $(S)$ such that $\alpha \vDash \phi^{\prime}$ and $t \in \llbracket e_{0} \rrbracket_{n-1}$. By Transitivity, there is a clause $\phi, \phi^{\prime} \Rightarrow e_{0} \subseteq$ $g\left(Y_{1}, \ldots, Y_{m}\right)$ in $S$. Since $\alpha \vDash \phi, \phi^{\prime}$ and $t \in \llbracket e_{0} \rrbracket_{n-1}$, we apply the induction hypothesis and get $t \in \llbracket g\left(Y_{1}, \ldots, Y_{m}\right) \rrbracket_{\alpha}$.
$\phi \Rightarrow X_{1} \cap \ldots \cap X_{p} \subseteq e$ We use an induction on the multiset $M\left(X_{1} \cap \ldots \cap X_{p}\right) \xlongequal{\text { def }}\left\{k_{1}, \ldots\right.$,
$\left.k_{p}\right\}$ of integers $k_{i}$ such that $t \in \llbracket X_{i} \rrbracket_{k_{i}}$ and $t \notin \llbracket X_{i} \rrbracket_{k_{i}-1}$. The maximum of $k_{1}, \ldots, k_{p}$ is, by hypothesis, smaller or equal to $n$. If it is strictly smaller than $n$, we use directly the induction hypothesis. If this multiset is equal to $\{n\}$, then we are back to the first case. Hence, let us assume now that the multiset is strictly larger than $\{n\}$, which means in particular that $p \geqslant 2$.
Since $t \in \llbracket X_{i} \rrbracket_{k_{i}}$ for every $i$, there are clauses $\phi_{i} \Rightarrow e_{0}^{i} \subseteq X_{i}$ in solved $(S)$ such that $\alpha \vDash \phi_{i}$ and $t \in \llbracket e_{0}^{i} \rrbracket_{k_{i}-1}$. If one of the expressions $e_{0}^{i}$ is an intersection of variables, then by Compatibility, there is a clause $\phi_{i}, \phi^{\prime} \Rightarrow X_{1} \cap \ldots \cap X_{i-1} \cap e_{0}^{i} \cap X_{i+1} \cap \ldots \cap X_{p} \subseteq e$ in $S$. Moreover, $\alpha \vDash \phi^{\prime}$ and $t \in \llbracket X_{1} \cap \ldots \cap X_{i-1} \cap e_{0}^{i} \cap X_{i+1} \cap \ldots \cap X_{p} \rrbracket_{n}$. Finally, $M\left(X_{1} \cap \ldots \cap X_{i-1} \cap e_{0}^{i} \cap X_{i+1} \cap \ldots \cap X_{p}\right)$ is obtained replacing $k_{i}$ in $M\left(X_{1} \cap \ldots, \cap X_{p}\right)$ with a multiset of strictly smaller numbers. Hence we get a strictly smaller multiset and we may apply the induction hypothesis: $t \in \llbracket e \rrbracket_{\alpha}$.
Now, if none of the expressions $e_{0}^{i}$ is an intersection of variables: $t=f\left(t_{1}, \ldots, t_{n}\right)$ and $e_{0}^{i}=f^{c_{i}}\left(e_{1}^{i}, \ldots, e_{n}^{i}\right)$. If one of the clauses

$$
\phi_{i} \Rightarrow f\left(e_{1}^{i}, \ldots, e_{n}^{i}\right) \subseteq X_{i}
$$

is marked, then there exist a clause $\phi_{i}^{\prime} \Rightarrow f\left(e_{1}^{i}, \ldots, e_{n}^{i}\right) \subseteq X_{i}^{\prime}$ marked by Deduction and clauses $\psi_{1} \Rightarrow X_{i}^{\prime} \subseteq X^{1}, \ldots, \psi_{n+1} \Rightarrow X^{n} \subseteq X_{i}$ such that $\phi_{i}=\phi_{i}^{\prime}, \psi_{1}, \ldots, \psi_{n+1}$. Thus, there is a clause $\phi_{i}^{\prime} \Rightarrow Z_{1} \cap \ldots \cap Z_{m} \subseteq X_{i}^{\prime}$ which triggered the application of Deduction. Then, applying repeatedly transitivity, there is a clause $\phi_{i} \Rightarrow Z_{1} \cap \ldots \cap Z_{m} \subseteq X_{i}$. In such a case, for every $k, e_{k}^{i}=Z_{k}^{1} \cap \ldots \cap Z_{k}^{m}$. Now, $t \in \llbracket e_{0}^{i} \rrbracket_{k_{i}-1}$, hence, for every $k, j$, $t_{k} \in \llbracket Z_{k}^{j} \rrbracket_{k_{i}-1}$. It follows that, for every $j, t \in \llbracket Z_{j} \rrbracket_{k_{i}-1}$. Now, by Compatibility, there is a clause $\phi, \phi_{i} \Rightarrow X_{1} \cap \ldots Z_{1} \cap \ldots \cap Z_{m} \ldots \cap X_{n} \subseteq e$ in $S$. And, as before, we get an intersection of variables with a strictly smaller multiset. We conclude thanks to the induction hypothesis.
Remains only the case where none of the clauses $\phi_{i} \Rightarrow f\left(e_{1}^{i}, \ldots, e_{n}^{i}\right) \subseteq X_{i}$ is marked. In this last case, we can apply Deduction. For every clause $\phi_{i} \Rightarrow f^{c_{i}}\left(e_{1}^{i}, \ldots, e_{n}^{i}\right) \subseteq$ $f\left(X_{1}^{i}, \ldots, X_{n}^{i}\right), \alpha \vDash \phi_{i}$ and $t \in \llbracket f^{c_{i}}\left(e_{1}^{i}, \ldots, e_{n}^{i}\right) \rrbracket_{k_{i}-1}$, hence $t_{j} \in \llbracket X_{j}^{i} \rrbracket_{k_{i}}$ for every $i, j$, by induction hypothesis.
If $t$ is a constant, then $\phi \Rightarrow t \subseteq e$ is a clause of $S$ and we can conclude.
Otherwise, if $e$ is a variable, then

$$
\phi \Rightarrow f\left(X_{1}^{1} \cap \ldots \cap X_{p}^{1}, \ldots, X_{1}^{n} \cap \ldots \cap X_{p}^{n}\right) \subseteq X
$$

is a clause of $\operatorname{solved}(S)$. Hence $t=f\left(t_{1}, \ldots, t_{n}\right) \in \llbracket X \rrbracket_{n+1}$.

If $e$ is not a variable, then $e$ must be $f\left(Y_{1}, \ldots, Y_{n}\right)$ and, by Projection, for every $j$,

$$
\phi, \operatorname{nonempty}\left(f\left(X_{1}^{1} \cap \ldots \cap X_{p}^{1}, \ldots, X_{1}^{n} \cap \ldots \cap X_{p}^{n}\right)\right) \Rightarrow X_{1}^{j} \cap \ldots \cap X_{p}^{j} \subseteq Y_{j}
$$

is a clause of $S . \alpha$ satisfies the premisse since $t$ is a witness for the second part of the precondition. Moreover, for every $j, t_{j} \in \llbracket X_{1}^{j} \cap \ldots \cap X_{p}^{j} \rrbracket_{n}$ and $M\left(X_{1}^{j} \cap \ldots \cap X_{p}^{j}\right)$ is smaller or equal to $M\left(X_{1} \cap \ldots, X_{p}\right)$. Then, by induction hypothesis, $t_{j} \in \llbracket Y_{j} \rrbracket_{\alpha}$ for every $j$. Hence $t \in \llbracket e \rrbracket_{\alpha}$.
$\phi \Rightarrow f^{c}\left(e_{1}, \ldots, e_{p}\right) \subseteq g\left(X_{1}, \ldots, X_{m}\right)$. If $f \neq g$, then, by Clash, the clause $\phi \Rightarrow$ false is in $S$, hence in $\operatorname{solved}(S)$, which contradicts $\alpha \vDash \phi$.
Assume now $f=g$. Then $t=f\left(t_{1}, \ldots, t_{n}\right)$. By Projection, there is a clause

$$
C_{i} \stackrel{\text { def }}{=} \phi, \operatorname{nonempty}\left(f^{c}\left(e_{1}, \ldots, e_{p}\right)\right) \Rightarrow e^{c \downarrow i} \subseteq X_{i}
$$

in $S$, for every $i$. Since $t \in \llbracket f^{c}\left(e_{1}, \ldots, e_{n}\right) \rrbracket_{\alpha}$,

$$
\alpha \vDash \phi, \operatorname{nonempty}\left(f^{c}\left(e_{1}, \ldots, e_{n}\right)\right) .
$$

For every $i, t_{i} \in \llbracket e_{i} \rrbracket_{n}$ since $t \in \llbracket f\left(e_{1}, \ldots, e_{p}\right) \rrbracket_{n}$ and $t_{i} \vDash c \downarrow_{i}$ since $t \vDash c$ (and by definition of $c \downarrow_{i}$ ). Thus $t_{i} \in \llbracket e_{i}^{c \downarrow i} \rrbracket_{n}$ which implies $t_{i} \in \llbracket X_{i} \rrbracket_{\alpha}$ by induction, thus $f\left(t_{1}, \ldots, t_{p}\right) \in \llbracket f\left(Y_{1}, \ldots, Y_{p}\right) \rrbracket_{\alpha}$.

We are now reduced to prove that the satisfiability of $\operatorname{solved}(S)$ is decidable.

### 5.3. Connection with automata with one memory

Theorem 51. For every satisfiable SET-constraint $S$, there is an (effectively computable) alternating automaton with one memory $A_{S}$ and an homomorphism $H$ such that $A_{S}$ accepts $t$ in the state $X$ iff $H(t) \in \llbracket X \rrbracket_{\alpha}$ where $\alpha$ is the least solution of solved $(S)$.

Proof. The memory alphabet of the automaton is the set of function symbols used in the constraint and the alphabet $\mathcal{F}_{A_{S}}$ is the memory alphabet with some additional symbols allowing to check on auxiliary branches non-emptiness conditions. More precisely,

- the states of $A_{S}$ consist of
- the variables of $S$. We write them $q_{X}$ such states, for $X$ a variable of $S$
- the states $q_{\phi}$ for every $\phi=$ nonempty $\left(e_{1}\right), \ldots$, nonempty $\left(e_{m}\right)$ such that nonempty $\left(e_{1}\right), \ldots, \operatorname{nonempty}\left(e_{m}\right)$, nonempty $\left(e_{m+1}\right), \ldots, \operatorname{nonempty}\left(e_{n}\right)$ is a precondition of a clause in $S$.
- for every expression $e=f^{c}\left(e_{1}, \ldots, e_{n}\right)$ and for every nonleaf position $p$ of such an expression, a state $q_{e, p}$ a state $q_{a}$ for every constant $a \in \mathcal{F}$
- The memory alphabet is $\mathcal{F}$
- The set of function symbols $\mathcal{F}_{A_{S}}$ consists in an auxiliary binary symbol $E$ and, for every symbol $f \in \mathcal{F}$ a symbol $f^{+1}$ whose arity is one plus the arity of $f$.

Let $H$ be the homomorphism:

$$
\begin{aligned}
H\left(f\left(t_{1}, \ldots, t_{n}\right)\right) & \stackrel{\text { def }}{\text { def }} f\left(H\left(t_{1}\right), \ldots, H\left(t_{n}\right)\right) \\
H\left(f^{+1}\left(t_{0}, t_{1}, \ldots, t_{n}\right)\right) & \xlongequal{\text { def }} f\left(H\left(t_{1}\right), \ldots, H\left(t_{n}\right)\right) \\
H\left(E\left(t_{1}, t_{2}\right)\right) & \stackrel{\text { def }}{=} a
\end{aligned}
$$

where $a$ is any constant in $\mathcal{F} . \alpha(X)$ will be the image by $H$ of the trees recognized in state $q_{X}$ by $A_{S}$.

By convention, if $p$ is a leaf position of $f^{c}\left(e_{1}, \ldots, e_{n}\right)$, we let $q_{e, p}$ be the conjunction $q_{X_{1}} \wedge \ldots \wedge q_{X_{m}}$ if $\left.e\right|_{p}=X_{1} \cap \ldots \cap X_{m}$ and $q_{e, p}=q_{a}$ if $\left.e\right|_{p}$ is the constant $a$.

The transition rules of $A_{S}$ consist of

1. $a \underset{a}{T} q_{a}$.
2. $f\left(q_{e, 1}, \ldots, q_{e, n}\right) \xrightarrow[\lambda x_{1}, \ldots, x_{n}, a]{\gamma} q_{f^{c}\left(e_{1}, \ldots, e_{n}\right)}$ for every literal nonempty $\left(f^{c}\left(e_{1}, \ldots, e_{n}\right)\right)$ occurring in $S . \gamma$ is the equality test $\tilde{c}$ defined as follows: $i \cdot \widetilde{p=j} \cdot q \stackrel{\text { def }}{=} i=j$ and $\widetilde{c_{1} \wedge c_{2}} \xlongequal{=} \widetilde{c_{1}} \wedge \widetilde{c_{2}}$.
3. $g\left(q_{e, p \cdot 1}, \ldots, q_{e, p \cdot k}\right) \xrightarrow[F]{\widetilde{c \downarrow p}} q_{e, p}$ in which $e=f^{c}\left(e_{1}, \ldots, e_{n}\right)$ occurs in $S, p$ is a nonroot position of $e,\left.e\right|_{p}$ is headed with $g . F=\lambda x_{1}, \ldots, x_{k} \cdot x_{i}$ if $p \cdot i$ is a prefix of a position in $P(c)$. Note that if there are several such indices, Lemmas 28 and 43 imply that the choice is irrelevant. $c \downarrow p$ is defined as in the projection rule: $c \downarrow \varepsilon \stackrel{\text { def }}{=} c, c_{1} \wedge c_{2} \downarrow$ $i \cdot p \stackrel{\text { def }}{=}\left(c_{1} \downarrow i \cdot p\right) \wedge\left(c_{2} \downarrow i \cdot p\right),\left(j \cdot p_{1}=k \cdot p_{2}\right) \downarrow i \cdot p \stackrel{\text { def }}{=}$ if either $i \neq j$ or $k \neq i$ and $\left(i \cdot p_{1}=i \cdot p_{2}\right) \downarrow i \cdot p \xlongequal{\text { def }}\left(p_{1}=p_{2}\right) \downarrow p$.
 $\left(e^{\prime}\right)$ which is an initial sequence of a precondition of a clause of $S$.
4. $E\left(q_{\phi}, q_{e}\right) \xrightarrow[\lambda x_{1}, x_{2}, x_{1}]{T} q_{\phi, \text { nonempty }(e)}$ for every $\phi$, nonempty $(e)$ which is an initial sequence of a precondition of a clause of $S$.
5. $f^{+1}\left(q_{\phi}, q_{e, 1}, \ldots, q_{e, n}\right) \xrightarrow[F]{\underset{\sim}{c}} q_{X}$ for every clause $\phi \Rightarrow e \subseteq X$ in $S$ such that $e=$ $f^{c}\left(e_{1}, \ldots, e_{n}\right) . F=\lambda x_{i_{1}}, \ldots, x_{i_{k}} \cdot t$ where $t$ is the expression $e$ in which, at each $\left.e\right|_{j \cdot p}$ such that $j \cdot p \in P(c)$ is replaced with $x_{i_{j}}$ where $i_{j}=j \in \tilde{c}$.
The following intermediate results are proved in Appendix B:

- If a term is accepted in state $q_{\phi}$ where $\phi$ is a precondition, then $\alpha \vDash \phi$.
- If $\alpha \vDash \phi$, then there is a term accepted in state $q_{\phi}$.
- If a term $t$ is accepted in state $q_{X}$, then $H(t) \in \alpha(X)$.
- If $t \in \alpha(X)$, then there exists $t^{\prime}$ is accepted in state $q_{X}$ such that $H\left(t^{\prime}\right)=t$.

From these lemmas, it follows that $A_{S}$ accepts the least solution $\alpha$ of $\operatorname{solved}(S)$ in the sense that $t$ is accepted in the state $X$ iff $H(t) \in \llbracket X \rrbracket_{\alpha}$.

Remark ( $W$ ). e conjecture that the minimal solution of a SET-constraint is recognized by an alternating tree automata with one memory. However, to prove this would require more saturation rules to get rid of nonemptiness conditions (as in Section 3).

As a consequence of Theorem 51 we get:
Theorem 52. The satisfiability of ET-constraints is decidable.
Proof. As a consequence of Lemmas 46, 49, and Theorem 50, we can assume that $S$ is a solved form.

Then, consider all clauses $\phi \Rightarrow$ false in $S$ and let $S^{\prime}$ be the rest of $S$.
$S^{\prime}$ is satisfiable, then, thanks to Theorem 51, there is an automaton $A_{S}$ such that $t$ is accepted in $q_{X}$ iff $H(t) \in \llbracket X \rrbracket_{\alpha}$ where $\alpha$ is the minimal solution of $S^{\prime}$.

Let $Q_{f}$ be the set of states $q_{\phi}$ such that $\phi \Rightarrow$ false is in $S$. Then $S$ is satisfiable iff, the automaton $A_{S}$ with final states $Q_{f}$ does not accept any tree, which is decidable thanks to Theorem 15.

## 6. Analysis of cryptographic protocols

We present here a decidable fragment of the class of protocols described in Section 2.2 and we illustrate the relevance of this fragment by an example (inspired by Kerberos).

### 6.1. A decidable class of protocols

As we have seen in Section 2, the specification of a protocol and its secrecy policy rely not only on the rules of the protocol, but also on the signature. In particular, we must say what are the expected types of each argument of a function symbol. This is far from being innocent, since this corresponds to the ability of each agent to recognize different data types. If all function symbols are assumed to take messages or agents as arguments and return messages or agents, then the principals are assumed to distinguish only agents from other messages. For instance, a pair of agents can be taken as a key in this case. Since many attacks are due to type confusion [8], such a signature specification would allow to detect many more attacks. On the other hand, if typing information is available, then deciding secrecy is easier.
One specificity of our model is that both the signature, hence the available type information, and the protocol itself are parameters. The defining conditions for our class will therefore depend on both the signature and the protocol and be more restrictive when the typing policy is more sloppy.

A rule of a protocol is of the form

$$
\left\{S(A, i, M), M_{1}, \ldots, M_{p}\right\} \longrightarrow\left\{S\left(A, i+1, M^{\prime}\right), M_{1}^{\prime}, \ldots, M_{q}^{\prime}\right\}
$$

where $M_{i}$ are messages. As we have seen in Section 2, the secrecy for general protocols of this form is undecidable. To obtain a decidable class, we consider protocols such that, for each rule, the variables which are shared by $M_{1}, \ldots, M_{p}, M_{1}^{\prime}, \ldots, M_{q}^{\prime}$ satisfy a "basicity condition". Roughly, such a condition will state that only one variable may occur several times in different contexts without being of basic type. For instance, if we do not assume
any special ability of agents to recognize data types, then repeated variables occurring in different contexts must be agents names, except for possibly one such variable.

In order to express our condition, let $X_{s}$ be a variable for each sort $s$ in the signature and let $\mathcal{C}_{\mathrm{msg}}$ be the union of the definite set constraints

$$
f\left(X_{\text {sort }_{1}}, \ldots, X_{\text {sort }_{n}}\right) \subseteq X_{\text {sort }}
$$

for every function symbol $f$ of type sort $_{1} \times \cdots \times$ sort $_{n} \rightarrow$ sort, and the definite set constraints

$$
X_{\text {sort }} \subseteq X_{\text {sort }}
$$

if sort is a subsort of sort'. The basic sorts are defined as in Definition 22: this is the largest set of sorts such that

- If $s$ is basic then $\mathcal{R}\left(X_{s}\right)$ only contains one-way symbols and basic sorts.
- If $s$ is basic then
- either $\mathcal{R}\left(X_{s}\right)$ contains only one clause $e \subseteq X_{s}$ such that $X_{s}$ does not occur in $e$.
- or every function symbol occurring in $\mathcal{R}\left(X_{s}\right)$ occurs (possibly) only in $\mathcal{R}\left(X_{s^{\prime}}\right)$ where $s^{\prime}$ is basic.
For instance, we have seen in Example 23 that, in our running example, all sorts are basic except Message. Let $\mathcal{B}_{\mathrm{msg}}$ be the set of basic sorts.

Now, for each rule

$$
r l=\left\{S(A, i, M), M_{1}, \ldots, M_{p}\right\} \longrightarrow\left\{S\left(A, i+1, M^{\prime}\right), M_{1}^{\prime}, \ldots, M_{q}^{\prime}\right\},
$$

let $t_{r l} \stackrel{\text { def }}{=}\left\langle\langle A, M\rangle, M_{1}, \ldots, M_{p},\left\langle A, M^{\prime}\right\rangle, M_{1}^{\prime}, \ldots, M_{q}^{\prime}\right\rangle$ and for each variable $Y$, occurring in $r l$, let $S_{r l, Y} \xlongequal{\text { def }}\left\{p\right.$ such that $\left.\left.t_{r l}\right|_{p}=Y\right\}$. Let $c_{r l, Y}$ be the equality constraint $p_{1}=\ldots=\left.p\right|_{k}$ for $p_{j} \in S_{r l, Y}$. In the particular case where $S_{r l, Y}$ is a singleton, $c_{r l, Y}$ is the empty constraint. Finally, let $c_{r l}$ be the conjunction of the $c_{r l, Y}$ for all variables $Y$ occurring in $r l$.

Example 53. We describe here the $t_{r l}$ and $c_{r l}$ corresponding to our running example (see Fig. 1, Example 5).

For the rule

$$
\emptyset \longrightarrow\left\{\begin{array}{l}
S(A, 1,\langle A, B, s\rangle), \\
S(B, 1,\langle B, s\rangle), \\
S(s, 1, s)
\end{array}\right\}
$$

we get the term and constraints:

$$
\begin{aligned}
& t_{r l_{0}} \stackrel{\text { def }}{=}\langle\langle A,\langle A, B, s\rangle\rangle,\langle B,\langle B, s\rangle\rangle,\langle s, s\rangle\rangle \\
& c_{r l_{0}} \stackrel{\text { def }}{=} 11=121 \wedge 122=21=221
\end{aligned}
$$

For rule 1:

$$
\begin{aligned}
& \{S(A, 1,\langle A, B, s\rangle)\} \longrightarrow\{S(A, 2,\langle A, B, s\rangle),\langle A, B\rangle\} \\
& t_{r l_{1}} \stackrel{\text { def }}{=}\langle\langle A,\langle A, B, s\rangle\rangle,\langle A,\langle A, B, s\rangle\rangle,\langle A, B\rangle\rangle, \\
& c_{r l_{1}} \stackrel{\text { def }}{=} 11=121=212=221 \wedge 122=222
\end{aligned}
$$

For rule 2:

$$
\begin{aligned}
& \left\{\begin{array}{l}
S(s, 1, s) \\
\langle A, B\rangle
\end{array}\right\} \rightarrow\left\{\begin{array}{l}
S(s, 2, s), \\
\left\{B, K(A, B),\{A, K(A, B)\}_{\operatorname{shr}(B)}\right\}_{\operatorname{shr}(A)}
\end{array}\right\} \\
& t_{r l_{2}} \stackrel{\text { def }}{=}\langle\langle s, s\rangle,\langle A, B\rangle,\langle s, s\rangle, \\
& \left.\quad\left\{\left[B, k_{1}(A, B),\left\{A, k_{1}(A, B)\right\}_{\operatorname{shr}(B)}\right\}_{\operatorname{shr}(A)}\right\rangle\right\rangle \\
& c_{r l_{2}} \stackrel{\text { def }}{=} 21=4121=41311=413121=42 \wedge 22=411=4122 \\
& =413122=4132
\end{aligned}
$$

For rule 3:

$$
\begin{gather*}
\qquad\left\{\begin{array}{l}
S(A, 2,\langle A, B, s\rangle), \\
\{B, X, Y\}_{\operatorname{shr}(A)}
\end{array}\right\} \rightarrow\left\{\begin{array}{l}
S(A, 3,\langle A, B, s, X, m(A, B)\rangle), \\
\left\langle\{m(A, B)\}_{X}, Y\right\rangle
\end{array}\right\} \\
t_{r l_{3}} \stackrel{\text { def }}{\stackrel{\text { di }}{ }}\left\langle\langle A,\langle A, B, s\rangle\rangle,\{B, X, Y\}_{\operatorname{shr}(A)},\right.  \tag{5}\\
c_{r l_{3}} \stackrel{\text { def }}{=} 11=121=22=31=321=3251=4111 \wedge  \tag{6}\\
\quad 122=211=322=3252=4112 \wedge 212=324=412 \wedge 213=42 . \tag{7}
\end{gather*}
$$

For rule 4:

$$
\begin{aligned}
& \left\{\begin{array}{l}
\left.\begin{array}{l}
S(B, 1,\langle B, s\rangle), \\
\left\langle\{Z\}_{X},\{A, X\}_{\operatorname{shr}(B)}\right\rangle
\end{array}\right\} \rightarrow\left\{\begin{array}{l}
\{S(B, 1,\langle B, s, A, Z, X\rangle), \\
\left.\{h(Z)\}_{X}\right\}
\end{array}\right\} \\
t_{r l_{4}} \stackrel{\text { def }}{=}\left\langle\langle B,\langle B, s\rangle\rangle,\left\langle\langle Z\}_{X},\{A, X\}_{\operatorname{shr}(B)\rangle},\right.\right. \\
\left.\langle B,\langle B, s, A, Z, X\rangle\rangle,\{H(Z)\}_{X}\right\}
\end{array}\right. \\
& c_{r l_{4}} \stackrel{\text { def }}{=} 11=121=222=31=321 \wedge 211=324=411 \wedge 212=2212 \\
& \quad=325=42
\end{aligned}
$$

For each term $t$, let $\tilde{t}$ be the expression obtained by replacing in $t$ each variable of sort $s$ with $X_{s}$. Finally, let $u_{r l}$ be the expression (with equality constraints) obtained from $\tilde{t_{r l}}$ by adding (at the top) the constraint $c_{r l}$.

We are now ready to define the basicity condition.
Definition 54. A protocol $P$ satisfies the basicity condition if, for each rule $r l$ of $P, u_{r l}$ as defined above satisfies the basicity condition w.r.t. the set of basic sorts.

We tried to give here a definition which is as general as possible, hence might be a bit difficult to grasp. Let us give a simple sufficient condition:

Proposition 55. A protocol satisfies the basicity condition if,for each rule rl of the protocol, one of the following holds:

- There is at most one variable occurring at least twice in rl and whose sort is not basic.
- There is a decomposition of tl into $C\left[t_{1}, \ldots, t_{n}\right]$ such that every variable which is not of sort Agentoccurs in at most one $t_{i}$.

For instance, in Example 53, only $Y, Z$ have a nonbasic sort, hence the first condition above is met by every rule.

For simplicity, we often write $A$ and $M s g$ instead of, respectively, $X_{\text {Agent }}$ and $X_{\text {Message }}$.

Theorem 56. If $P$ satisfies the basicity condition, then the secrecy of $P$ is decidable.
In particular, our running example satisfies the basicity condition. Indeed, there are at most two occurrences of nonbasic variables in each expressions. However, note that giving the ability for agents to recognize messages of the form $K(a, b)$ was not innocent: if we replace our key variable $K$ by a message variable $X$ then our protocol does not remain in our decidable class.
We prove the theorem in next section.

### 6.2. Proof of Theorem 56

The proof of Theorem 56 proceeds in two steps. First, we show that every protocol can be "translated" into Horn clauses such that a message can be sent if and only if a corresponding formula can be derived from the Horn clauses. Then, we show that if a protocol satisfies the basicity condition then the corresponding Horn clauses can be expressed as ET-constraints, thus secrecy is decidable.

Step 1:
Lemma 57. Let $P$ be a protocol with its secrecy policy $S_{0}$. Let $I_{0}$ be the maximal set of initial knowledge of the intruder (compatible with the secrecy policy $S_{0}$ of $P$ ) mentioned in Section 2. Then, there exists a set $\mathcal{H}$ of Horn clauses with a special predicate I such that $I(m)$ is derivable from $\mathcal{C}$ (where $m$ is a message), iff there exists a reachable $H$ such that $m \in H$.

Proof. $\mathcal{H}$ is built as the union of four sets: $\mathcal{H}_{\mathrm{msg}}, \mathcal{H}_{I_{0}}, \mathcal{H}_{I}$ and $\mathcal{H}_{P}$.
$\mathcal{H}_{\mathrm{msg}}$ corresponds to the construction of the messages with their sort: $\mathcal{H}_{\mathrm{msg}}$ is the union of the clauses

$$
\frac{P_{\text {sort }_{1}}\left(x_{1}\right) \ldots P_{\text {sort }_{n}}\left(x_{n}\right)}{P_{\text {sort }}\left(f\left(x_{1}, \ldots, x_{n}\right)\right)}
$$

for every function symbol $f$ of type sort $_{1} \times \cdots \times$ sort $_{n} \rightarrow$ sort, where the $P_{\text {sort }}$ are new predicates. We also add to $\mathcal{H}_{\mathrm{msg}}$ the union of the clauses

$$
\frac{P_{\text {Sort }}(x)}{P_{\text {Sort }^{\prime}}(x)}
$$

for every sort, sort' such that sort is a subsort of sort'. Moreover, we need to distinguish between symmetric and public keys: every term is symmetric except the terms of the


Fig. 8. Horn clauses for the intruder capabilities.
form $\operatorname{pub}(t)$ or $\operatorname{prv}(t)$. Thus we add to $\mathcal{H}_{\mathrm{msg}}$ the clauses

$$
\frac{P_{\text {Message }}\left(x_{1}\right) \ldots P_{\text {Message }}\left(x_{n}\right)}{\operatorname{Sym}\left(f\left(x_{1}, \ldots, x_{n}\right)\right)}
$$

for every function symbol $f, f \neq \operatorname{pub}$ and $f \neq \mathrm{prv}$. Then it is easy to prove that $P_{\text {Sort }}(m)$ is derivable from $\mathcal{H}_{\mathrm{msg}}$ if and only if $m$ is a message of sort sort and $\operatorname{Sym}(m)$ is derivable from $\mathcal{H}_{\mathrm{msg}}$ if and only if $m$ is a symmetric term.
Then, by the following lemma (proved in Appendix C), there exists a set of Horn clauses $\mathcal{H}_{I_{0}}$ such that $I_{0}(m)$ is derivable from $\mathcal{H}_{I_{0}} \cup \mathcal{H}_{\mathrm{msg}}$ if and only if $m \in I_{0}$.

Lemma 58. Let $t_{1}, \ldots, t_{n}$ be message schemes with the free variables $x_{1}, \ldots, x_{k}$. Then, there exists a set of Horn clauses $\mathcal{H}_{I_{0}}$ with two predicates $I_{0}$ and $P \geqslant p$ such that $I_{0}(m)$ is derivable from $\mathcal{H}_{I_{0}} \cup \mathcal{H}_{\mathrm{msg}}$ if and only if parts $(m) \cap\left\{t_{i} \sigma \mid 1 \leqslant i \leqslant n, \sigma\left(x_{i}\right) \in \mathcal{A}_{h}\right\}=\emptyset$.

The clauses of $\mathcal{H}_{I}$ are described Fig. 8. They simulate the capabilities of the intruder.
To simulate the protocol rules, we first define the set of expressions types $(t)$ generated by a message scheme $t$ by induction on $t$. If $t$ is a constant, then types $(t)=\emptyset$. If $t$ is a variable of sort sort, then types $(t)=\left\{P_{\text {Sort }}(t)\right\}$. If $t$ is a term of the form $f\left(t_{1}, \ldots, t_{n}\right)$, then

$$
\begin{array}{llll}
I\left(s_{i}(A, M)\right) & I\left(M_{1}\right) & \cdots I\left(M_{p}\right) & \operatorname{types}\left(A, M, M_{1}, \ldots, M_{p}, M_{1}^{\prime}, \ldots, M_{q}^{\prime}\right) \\
\hline & I\left(s_{i+1}\left(A, M^{\prime}\right)\right) \\
\hline \begin{array}{llll}
I\left(s_{i}(A, M)\right) & I\left(M_{1}\right) & \cdots I\left(M_{p}\right) & \operatorname{types}\left(A, M, M_{1}, \ldots, M_{p}, M_{1}^{\prime}, \ldots, M_{q}^{\prime}\right)
\end{array} \\
\hline & I\left(M_{i}^{\prime}\right)
\end{array}
$$

Fig. 9. Horn clauses corresponding to the rule $\left\{S(A, i, M), M_{1}, \ldots, M_{p}\right\} \longrightarrow\left\{S\left(A, i+1, M^{\prime}\right), M_{1}^{\prime}, \ldots, M_{q}^{\prime}\right\}$.
types $(t)=\bigcup_{1 \leqslant i \leqslant n}$ types $\left(t_{i}\right)$. Intuitively types $(t)$ is the set of constraints corresponding to the sorts of the variables of $t$. We sometimes write types $\left(t_{1}, t_{2}\right)$ instead of types $\left(t_{1}\right) \cup$ types $\left(t_{2}\right)$.

Example 59. Let us consider the message scheme $\{B, K, X\}_{K}$ of our running example. Then

$$
\operatorname{types}\left(\{B, X, Y\}_{\operatorname{shr}(A)}\right)=\left\{P_{\text {Agent }}(B), P_{\text {Key }}(X), P_{\text {Message }}(Y), P_{\text {Agent }}(A)\right\}
$$

ensures that $A$ and $B$ stand for agent variables, $X$ stands for a key variable and $Y$ for a message variable.

Then $\mathcal{H}_{P}$ is the union, for each rule

$$
\left\{S(A, i, M), M_{1}, \ldots, M_{p}\right\} \longrightarrow\left\{S\left(A, i+1, M^{\prime}\right), M_{1}^{\prime}, \ldots, M_{q}^{\prime}\right\}
$$

of the protocol $P$, of the clauses described in Fig. 9 .
We prove by induction in Appendix C that $\mathcal{H}$ verifies the required property, which concludes the proof of Lemma 57.

Remark. The predicate symbols of $\mathcal{H}$ are the $P_{\text {Sort }}, P_{\geqslant p}$ introduced by Lemma 58 and three distinct predicate symbols: $I, I_{0}$ and Sym.

Example 60. We present here the clauses $\mathcal{H}_{P}$ corresponding to our running example. For simplicity, we factorize the rules whose premises are identical, letting them contain several conclusions (though it must be kept in mind that these are Horn clauses).

## Initialization rule:

$$
\frac{P_{\text {Agent }}(A) \quad P_{\text {Agent }}(B)}{I\left(s_{1}(A,\langle A, B, s\rangle)\right) \quad I\left(s_{1}(B,\langle B, s\rangle)\right), \quad I\left(s_{1}(s, s)\right)}
$$

Rule 1:

$$
\begin{array}{ll}
P_{\text {Agent }}(A) \quad P_{\text {Agent }}(B) & I\left(s_{1}(A,\langle A, B, s\rangle)\right) \\
\hline I\left(s_{2}(A,\langle A, B, s\rangle)\right) & I(\langle A, B\rangle)
\end{array}
$$

Rule 2:

$$
\frac{P_{\text {Agent }}(A) \quad P_{\text {Agent }}(B) \quad I\left(s_{1}(s, s)\right) \quad I(\langle A, B\rangle)}{I\left(s_{2}(s, s)\right) \quad I\left(\left\{B, K(A, B),\left\{x_{A}, K(A, B)\right\}_{\operatorname{shr}(B)}\right\}_{\operatorname{shr}(A)}\right)}
$$

Rule 3:

$$
\begin{gathered}
P_{\text {Agent }}(A) P_{\text {Agent }}(B) P_{\text {Key }}(X) P_{\text {Message }}(Y) \\
I\left(s_{2}(A,\langle A, B, s\rangle)\right) I\left(\{B, X, Y\}_{\operatorname{shr}(A))}\right) \\
I\left(s_{3}(A,\langle A, B, s, X, m(A, B)\rangle)\right) \quad I\left(\left\{\{m(A, B)\}_{X}, Y\right\rangle\right)
\end{gathered}
$$

and similarly for Rule 4 .
Step 2: We can start the second part of the proof of Theorem 56 . We write $\mathcal{H} \vdash^{\star} E$ when $E$ is derivable from $\mathcal{H}$.
Let $P$ be a protocol. Its secrecy policy $S_{0}$ is defined by

$$
S_{0}=\bigcup_{j=1}^{s}\left\{t_{1}^{j}, \ldots, t_{n_{j}}^{j} \mid x_{1}^{j}, \ldots, x_{k_{j}}^{j} \in \mathcal{A}_{k}\right\},
$$

where the $t_{i}^{j}$ are message schemes with free variables $x_{1}^{j}, \ldots, x_{k_{j}}^{j}$. By definition of secrecy, $P$ is not secure iff there exists a reachable $H$ such that fake $\left(\operatorname{Cont}(H) \cup I_{0}\right) \cap S_{0}=\emptyset$, i.e., iff there exists a reachable $H^{\prime}$ such that $\operatorname{Cont}\left(H^{\prime}\right) \cap S_{0}=\emptyset$, i.e., iff there exists a reachable $H^{\prime}, \exists i, j, \exists a_{i_{1}}, \ldots, a_{i_{p}} \in \mathcal{A}_{h}$ such that $t_{i}^{j}\left(a_{i_{1}}, \ldots, a_{i_{p}}\right) \in H^{\prime}$ iff, by Lemma $57, \exists i, j, \exists a_{i_{1}}, \ldots, a_{i_{p}} \in \mathcal{A}_{h}$ such that $\mathcal{H} \vdash^{\star} I\left(t_{i}^{j}\left(a_{i_{1}}, \ldots, a_{i_{p}}\right)\right)$.

Thus, we are left to prove that $\exists i, j, \exists a_{i_{1}}, \ldots, a_{i_{p}} \in \mathcal{A}_{h}$ such that $\mathcal{H} \vdash^{\star} I\left(t_{i}^{j}\left(a_{i_{1}}, \ldots\right.\right.$, $\left.a_{i_{p}}\right)$ ) is decidable.

We express the set $\mathcal{H}$ of Horn clauses as a set constraint: the set of definite set constraint corresponding to $\mathcal{H}_{\mathrm{msg}}$ is the set $\mathcal{C}_{\mathrm{msg}}$ described in Section 6.1 augmented with the inclusions:

$$
f(M s g, \ldots, M s g) \subseteq X_{S y m}
$$

for every function symbol $f \in \mathcal{F}, f \neq \mathrm{pub}, f \neq \mathrm{prv}$.
The definite set constraint $\mathcal{C}_{I}$ corresponding to $\mathcal{H}_{I}$ is described in Fig. 10. The set of definite set constraint $\mathcal{C}_{I_{0}}$ corresponding to $\mathcal{H}_{I_{0}}$ is constructed similarly.

We associate with each Horn clause described in Fig. 9 the following ET-constraint:

$$
\begin{equation*}
\left\langle s_{i}(A, \widetilde{M}) \cap I, \widetilde{M_{1}} \cap I, \ldots, \widetilde{M_{p}} \cap I, s_{i+1}\left(A, \widetilde{M^{\prime}}\right), \widetilde{M_{1}^{\prime}}, \ldots, \widetilde{M_{q}^{\prime}}{ }^{c_{r l}} \subset I,\right. \tag{9}
\end{equation*}
$$

where $c_{r l}$ and $\sim$ are defined in Section 6.1. The union of these ET-constraint is denoted by $\mathcal{C}_{P}$.


Fig. 10. Set constraints corresponding to the intruder capabilities.

Example 61. The Horn clause corresponding to rule 3 of our running example is expressed by

$$
\begin{aligned}
& \left\langle s_{2}(A,\langle A, A, s\rangle) \cap I,\left\{A, X_{\mathrm{Key}}, M s g\right\}_{\mathrm{shr}(A)} \cap I\right. \\
& \left.\quad s_{3}\left(A,\left\langle A, A, s, X_{\mathrm{Key}}, m(A, A)\right\rangle\right),\left\{\{m(A, A)\}_{X_{\mathrm{Key}}}, M s g\right\rangle\right\rangle^{c_{r l_{3}}} \subseteq I
\end{aligned}
$$

where $c_{r l 3}$ is the equality constraint described in Eq. (5), Section 6.1.
Let $\mathcal{C}$ be the union of $\mathcal{C}_{\mathrm{msg}}, \mathcal{C}_{I_{0}}, \mathcal{C}_{I}$ and $\mathcal{C}_{P}$. The set variables of $\mathcal{C}$ are the variables $X_{\text {sort }}$ for each sort sort and the four additional variables: $X_{\geqslant p}$, which corresponds to $P_{\geqslant p}, I, I_{0}$ and $X_{\text {Sym }}$.
$\mathcal{C}$ is a faithful representation of $\mathcal{H}$ :
Lemma 62. Let $\mathcal{M}$ be a collection of sets $S_{Q}$ for every (unary) predicate symbol $Q$. Then $\mathcal{M}$ is a model of $\mathcal{H}$ iff the substitution $\sigma_{\mathcal{M}}$ assigning $X_{Q}$ to $S_{Q}$ is a solution of $\mathcal{C}$.

Proof. Let $\mathcal{M}$ be a model of $\mathcal{H}$ and let us show that $\sigma_{\mathcal{M}}$ satisfies $\mathcal{C}$.
The only nonobvious part of the proof is to show that $S_{I}$ satisfies the set constraint defined in Eq. (9). Let $r l$ be a rule of the protocol, let

$$
t \in\left\langle s_{i}(A, \widetilde{M} \sigma) \cap S_{I}, \widetilde{M_{1}} \sigma \cap S_{I}, \ldots, \widetilde{M_{p}} \sigma \cap S_{I}, s_{i+1}\left(A, \widetilde{M^{\prime}} \sigma\right), \widetilde{M_{1}^{\prime}} \sigma, \ldots, \widetilde{M_{q}^{\prime}} \sigma\right\rangle
$$

such that $t$ satisfies $c_{r l} . t=\left\langle s_{i}(a, m), m_{1}, \ldots, m_{p}, s_{i+1}\left(a, m^{\prime}\right), m_{1}^{\prime}, \ldots, m_{q}^{\prime}\right\rangle$ and $s_{i}(a, m), m_{1}, \ldots, m_{p} \in S_{I}$. Since $t$ satisfies $c_{r l}$, by applying the clause defined in

Fig. 9 , we deduce $s_{i+1}\left(a, m^{\prime}\right), m_{1}^{\prime}, \ldots, m_{q}^{\prime} \in S_{I}$, thus, by applying the clause

$$
\frac{I\left(m_{1}\right) I\left(m_{2}\right)}{I\left(\left\{m_{1}, m_{2}\right\rangle\right)}, t \text { is in } S_{I} \text {, }
$$

thus $S_{I}$ satisfies constraint 9 .
Conversely, let $\sigma_{\mathcal{M}}$ be a model of $\mathcal{C}$. The only nonobvious part of the proof is to show that $S_{I}$ satisfies the Horn clause defined in Fig. 9. Let $m=M \sigma^{\prime}, m=M^{\prime} \sigma$, $m_{i}=M_{i} \sigma^{\prime}, m_{i}^{\prime}=M_{i}^{\prime} \sigma, a=A \sigma^{\prime}$ such that $s_{i}(a, m), m_{1}, \ldots, m_{p} \in S_{I}$ and $\sigma^{\prime}$ satisfies the conditions $\operatorname{cond}\left(A, M, M_{1}, \ldots, M_{p}, M_{1}^{\prime}, \ldots, M_{q}^{\prime}\right)$. Thus for every variable $X$ of $A, M, M_{1}, \ldots, M_{p}, M_{1}^{\prime}, \ldots, M_{q}^{\prime}, \sigma^{\prime}(X) \in \mathcal{I}\left(S_{\tilde{X}}\right): \sigma^{\prime}$ respects the type of the variables. Let $t=\left\langle s_{i}(a, m), m_{1}, \ldots, m_{p}, s_{i+1}\left(a, m^{\prime}\right), m_{1}^{\prime}, \ldots, m_{q}^{\prime}\right\rangle$, then

$$
\begin{gathered}
t \in\left\langle s_{i}(A, \tilde{M} \sigma) \cap S_{I}, \widetilde{M_{1}} \sigma \cap S_{I}, \ldots, \widetilde{M_{p}} \sigma \cap S_{I},\right. \\
\left.s_{i+1}\left(A, \widetilde{M^{\prime}} \sigma\right), \widetilde{M_{1}^{\prime}} \sigma, \ldots, \widetilde{M_{q}^{\prime}} \sigma\right\rangle
\end{gathered}
$$

and, by construction of $c_{r l}, t$ satisfies $c_{r l}$. Thus $t$ is in $S_{I}$. Applying the set constraint $\left\rangle_{i}^{-1}\left(S_{I}\right) \subseteq S_{I}\right.$, we get $s_{i+1}\left(a, m^{\prime}\right), m_{1}^{\prime}, \ldots, m_{q}^{\prime} \in S_{I}$, thus $S_{I}$ satisfies the Horn clause defined in Fig. 9.

Then

$$
\exists i, \exists a_{i_{1}}, \ldots, a_{i_{p}} \in \mathcal{A}_{h} \text { such that } \mathcal{H} \vdash^{\star} I\left(t_{i}\left(a_{i_{1}}, \ldots, a_{i_{p}}\right)\right)
$$

if and only if

$$
\mathcal{C} \bigcup_{i} I \cap t_{i}\left(X_{\text {Ah }}, \ldots, X_{\text {Ah }}\right) \subseteq \perp
$$

is not satisfiable.
First, if $\exists i, \exists a_{i_{1}}, \ldots, a_{i_{p}} \in \mathcal{A}_{h}$ such that $\mathcal{H} \vdash^{\star} I\left(t_{i}\left(a_{i_{1}}, \ldots, a_{i_{p}}\right)\right)$ then

$$
\mathcal{C} \cup_{i} I \cap t_{i}\left(X_{\text {Ah }}, \ldots, X_{\text {Ah }}\right) \subseteq \perp \text { is clearly not satisfiable. }
$$

Now, assume $\forall i, \forall a_{i_{1}}, \ldots, a_{i_{p}} \in \mathcal{A}_{h} \quad \mathcal{H} \quad K^{\star} I\left(t_{i}\left(a_{i_{1}}, \ldots, a_{i_{p}}\right)\right)$ and consider the least Herbrand model $\mathcal{M}$ of $\mathcal{H}$. Then $\mathcal{M}$ also models every negative clause of the form $\neg I\left(t_{i}\left(a_{i_{1}}\right.\right.$, $\left.\ldots, a_{i_{p}}\right)$ ). Applying Lemma $62, \sigma_{\mathcal{M}}$ is a solution for:

$$
\mathcal{C} \cup_{i} I \cap t_{i}\left(X_{\text {Ah }}, \ldots, X_{\text {Ah }}\right) \subseteq \perp
$$

Assume now that $P$ satisfies the basicity condition as in Theorem 56, then the set $\mathcal{C}$ as constructed above is a set of ET-constraints. Thus

$$
\mathcal{C} \bigcup_{i} I \cap t_{i}\left(X_{\text {Ah }}, \ldots, X_{\text {Ah }}\right) \subseteq \perp
$$

is also a set of ET-constraints. Then, thanks to Theorem 52, the satisfiability of this constraint is decidable, which completes the proof of Theorem 56.

## 7. Conclusion

Let us summarize the contributions of the paper (roughly in increasing order of significance) and discuss their meaning and possible further developments.

1. The security of a protocol $P$ is undecidable, even for a restricted class in which there are no nonces, no compound keys and there is at least one honest instance of $P$. This is Theorem 6. This shows that the source of undecidability does not come from nonces, but from the memorization and copying facilities of the agents.
2. The satisfiability of intersection constraints with nonemptiness guards is DEXPTIMEcomplete. This is Theorem 11. It is a slight extension of results about set constraints.
3. We introduced the new class of tree automata with one memory and we showed that the emptiness is DEXPTIME-complete for this class. This is Theorem 15. This result is interesting in itself. One open question is its generalization with disequality tests (and not only equality tests between memory contents).
4. We introduced a class of set constraints with equality tests, in which the tests are not restricted to brother positions. We showed the decidability of constraints in this class by a reduction to tree automata with one memory. This is Theorem 52. It must be emphasized that we did not use the full power of automata with one memory here.
Interpreting Lemma 39 in the context of cryptographic protocols, it shows that, for basic variables, we may restrict our attention to finitely many instances (the representatives w.r.t. an appropriate equivalence relation). This shows in particular that we can assume w.l.o.g. that there is a bounded number of principals (the bound is given by the index of the equivalence relation).
One possible research direction is to investigate generalizations of this lemma, for instance in the context of nonces: is there an equivalence relation (preserving the solutions) which reduces the general case to the case of finitely many nonces? Such a result would not necessarily contradict the undecidability result of [18] since the protocol resulting from the coding of that paper does not satisfy the basicness hypothesis. In other words, as suggested by Theorem 6, the key for deciding the secrecy of cryptographic protocols might be to limit the copying facilities of the agents, not the number of sessions or nonces they generate.
5. We showed the decidability of secrecy for a class of cryptographic protocols, without any assumptions on the number of sessions (whether parallel or not). This is Theorem 56. This result is obtained by a reduction to set constraints with equality tests, but we did not use the full power of such constraints.
The use of set constraints, abstracting away the order in which messages have been sent over the network is proved to be relevant. Also, the ability of agents to recognize different types of data appeared clearly as a simplification factor, which can be tuned so that we fall in or out of the decidable class. We have showed the relationship between this ability and the copying facilities of the agents: the more they are able to distinguish between different data types, the more they are allowed to copy blindly pieces of messages, without escaping from the decidable class.
There are still several weaknesses in our paper. First, the constraint solving technique is too complex: we conjecture that our algorithm is in DEXPTIME, though we only showed a doubly exponential upper bound. It is also too complicated for the applications we have
in mind. That is mainly because we tried to be as general as possible. However, most of the time, we do not need such general constraints. In particular, we can avoid the most complicated step (Lemma 28) simply by designing normalized constraints only.

Finally the big open question is the extension of these results to an unbounded number of nonces.

## Appendix A. Proof of Lemma 28

Lemma 28 (Every expression). e which satisfies the basicity condition can be transformed into a normal expression $e^{\prime}$ such that, for every $\sigma, \llbracket e \rrbracket_{\sigma}=\llbracket e^{\prime} \rrbracket_{\sigma}$ and, moreover $\left|e^{\prime}\right|_{\mathcal{F}}$ and $\left|e^{\prime}\right|_{t}$ are polynomially bounded by $|e|_{t}$ and $\left|e^{\prime}\right|_{\mathcal{F}}$.

Note first that condition 2 is initially satisfied by all expressions since we assumed that any two expressions in $P(c)$ are incomparable with respect to the prefix ordering.

We are going to perform successive transformations, verifying more and more conditions, while preserving those which have already been reached. Initially, we only assume that condition 1 is satisfied, as stated in the hypothesis of the lemma.

Step 1: The goal of this step is to ensure, in addition to properties 1 and 2 a property, which implies condition 3.

If $c$ is an equality test and $p$ is a position, we write $p \cdot c$ the equality test $\bigwedge_{q \sim_{c} r} p \cdot q=p \cdot r$. In addition to condition 3 , we want to ensure that in any expression $g_{-}^{c}\left(\ldots, e_{i}, \ldots\right)$ such that, for some $c_{1}, i \cdot c_{1} \subseteq c$, if $e_{i}=f^{c^{\prime}}\left(\overrightarrow{e^{\prime}}\right) \cap e^{\prime \prime}$, then $P\left(c_{1}\right) \subseteq \Pi\left(f^{c^{\prime}}\left(\overrightarrow{e^{\prime}}\right)\right)$.

To define our rule, we first need to introduce a new equivalence relation:
Definition 63. Given an equality test $c$ such that $c$ is satisfiable, we define $\simeq_{c}$ and $S_{c}$ to be the least set $S$ and equivalence relation $\simeq$ such that:

$$
\left.\begin{array}{l}
p \sim_{c} q \Rightarrow p, q \in S \text { and } p \simeq q \\
p \sim_{c^{\prime}} q \Rightarrow p, q \in S \text { and } p \simeq q \\
p \simeq q \Rightarrow q \simeq p \\
p \simeq q, q \simeq r \Rightarrow p \simeq r \\
p \cdot q \in S \\
p \simeq p^{\prime}
\end{array}\right\} \Rightarrow\left\{\begin{array}{l}
p^{\prime} \cdot q \in S \\
p \cdot q \simeq p^{\prime} \cdot q
\end{array}\right.
$$

Lemma 64. The fixed point for $\simeq_{c}$ and $S_{c}$ is reached after a finite number of steps. In addition, there exists an order $\leqslant_{c}$ on the equivalence classes of $\simeq_{c}$ such that $u \leqslant_{c} u^{\prime}$ implies that no position of $u$ is a prefix of a position of $u$ '.

Proof (sketch). First, if there are two nonempty positions such that $p \cdot q \simeq p$ then $c$ is unsatisfiable: by induction (on the fixed point computation of $S, \simeq$ ), if $t \vDash c \wedge c^{\prime}$, then for every $p \simeq q, t \vDash p=q$. Since we are only considering finite terms, we cannot have $\left.t\right|_{p}=\left.t\right|_{p \cdot q}$.

Consider the DAG $G$ whose vertices are elements of $P(c) \cup P\left(c^{\prime}\right)$ and (labeled) edges $p \xrightarrow{i} p \cdot i$. Then, for each $p \sim_{c_{i}} p^{\prime}$, merge the two corresponding vertices. We get a


Fig. 11. $G$ and $G_{f}$ for $c$.
new graph $G_{f}$, whose set of vertices is contained in the original set of vertices. Then, by induction (on the fixed point computation of $S, \simeq$ ), $S$ is included in the set of paths of in $G_{f}$ and if $p \simeq q$, then the path labeled by $p$ and those labeled by $q$ leads to the same vertex (starting from the vertex $\varepsilon$ ).
As we have seen above, $G_{f}$ is acyclic, thus $S$ and $\simeq$ are finite, thus the fixed point for $S, \simeq$ is reached after a finite number of steps. Note that the number of vertices of $G_{f}$ is smaller or equal to the number of vertices of $G$ (Fig. 11).

Since $G_{f}$ is acyclic, $G_{f}$ induces an order $\leqslant$ on its vertices such that if $v \leqslant v^{\prime}$ than no path leading to the vertex $v$ is a prefix of a path leading to the vertex $v^{\prime}$. Each equivalence class $u$ of $\simeq$ is included in the set of paths of one of the vertex $v_{u}$ of the graph. We first order arbitrarily the equivalence classes which lead to the same vertices and then we extend this order by $u \leqslant{ }_{c} u^{\prime}$ if $v_{u} \leqslant v_{u^{\prime}}$ and $v_{u} \neq v_{u^{\prime}}$. Then $u \leqslant{ }_{c} u^{\prime}$ implies that no position of $u$ is a prefix of a position of $u^{\prime}$.

Example 65. Consider $c \stackrel{\text { def }}{=} 1=21 \wedge 3=41 \wedge 2=31$. Then, the graphs $G$ and $G_{f}$ are pictured in Fig. 11 and

$$
\simeq_{c}=\{1=21,3=41,2=31,21=311,31=411,311=4111\} .
$$

In addition, we define the equality test $c_{e}$ of an expression $e$ by induction on $e$ by:

$$
\begin{aligned}
c_{e_{1} \wedge \ldots \wedge e_{n}} & =c_{e_{1}} \wedge \ldots \wedge c_{e_{n}} \\
c_{f^{c}\left(e_{1}, \ldots, e_{n}\right)} & =c \wedge 1 \cdot c_{e_{1}} \wedge \ldots \wedge n \cdot c_{e_{n}} \wedge 1=1 \wedge \ldots \wedge n=n
\end{aligned}
$$

Let $\simeq_{c_{e}}$ be the equivalence relation corresponding to $c_{e}$ and $u_{1}, \ldots, u_{n}$ be its equivalence classes numbered in such a way that $u_{i} \leqslant c_{e} u_{j}$ implies $i \leqslant j$ (this is possible thanks to Lemma 64).

Then, $\left(N_{1}\right)$ is the successive application of $N_{u_{1}}, \ldots, N_{u_{n}}$ where

$$
\left(\begin{array}{ll}
\left.N_{u}\right) & e \rightarrow e
\end{array} \bigcap_{p^{\prime} \in u \cap \Pi(e)}^{\left.e^{\prime \prime} \in e\right|_{p^{\prime}}} e_{p \in u \cap \Pi(e)} e^{\prime \prime}\right]
$$

Note. The rule $\left(N_{1}\right)$ is obviously terminating since it requires at most as many steps as the number of classes modulo $\simeq_{c_{e}}$. However, its complexity is unclear. We conjecture for instance that $\left|e^{\prime}\right|_{\mathcal{F}}$ is polynomially bounded by $|e|_{\mathcal{F}}$ and $|e|_{t}$.

Lemma 66. ( $N_{1}$ ) preserves the semantics as well as properties 1 and 2 .
Proof. For every equivalence class $u$ and for every $p_{1}, p_{2} \in u \cap \Pi(e)$, by construction, $c_{e} \vDash p_{1}=p_{2}$. Hence the rule preserves the semantics.

Now, it preserves properties 1 and 2 since it consists in repeatedly replacing an expression $e^{\prime}$ with the intersection of $e^{\prime}$ and other expressions, without changing the tests and, if an expression is basic, then its intersection with any other expression is also basic.

Lemma 67. If $e^{\prime}$ is the result applying $\left(N_{1}\right)$ to $e$, then:

- $\forall p \in S_{c_{e}} \cap \Pi\left(e^{\prime}\right),\left.e^{\prime}\right|_{p}$ is a singleton.
- $\forall p \simeq_{c_{e}} p^{\prime},\left.e^{\prime}\right|_{p}=\left.e^{\prime}\right|_{p^{\prime}}$.
- if $\left.e^{\prime}\right|_{p}=f^{c}\left(e_{1}, \ldots, e_{n}\right) \cap e^{\prime \prime}$, then $p \cdot c \subseteq \simeq_{c_{e}}$ and for all $1 \leqslant i \leqslant n, p \cdot i \in S_{c_{e}}$.
- if $p \in \Pi\left(e^{\prime}\right)$ and $\left.p\right\rangle p^{\prime}$ for some $p^{\prime} \in u_{i}$ where $u_{i}$ is minimal for $\leqslant_{c}$, then $\left.e^{\prime}\right|_{p} \subseteq$ $\left.\cup_{q \in \Pi(e)} e\right|_{q}$.
- if $\left.e^{\prime}\right|_{p}=f^{c}\left(e_{1}, \ldots, e_{n}\right) \cap f^{c^{\prime}}\left(e_{1}^{\prime}, \ldots, e_{n}^{\prime}\right) \cap e^{\prime \prime}$ then for every $i=1, . ., n, e_{i}=e_{i}^{\prime}$.

Proof. We prove by induction on $k$ that if $e^{\prime}$ is the result of applying $N_{u_{1}} \ldots N_{u_{k}}$ to $e$, then for all $i \leqslant k$,

1. $\forall p \in u_{i} \cap \Pi\left(e^{\prime}\right),\left.e^{\prime}\right|_{p}$ is a singleton.
2. $\forall p, p^{\prime} \in u_{i} \cap \Pi\left(e^{\prime}\right),\left.e^{\prime}\right|_{p}=\left.e^{\prime}\right|_{p^{\prime}}$.
3. if $\left.e^{\prime}\right|_{p}=f^{c}\left(e_{1}, \ldots, e_{n}\right) \cap e^{\prime \prime}$, then $p \cdot c \subseteq \simeq_{c_{e}}$ and for all $1 \leqslant i \leqslant n, p \cdot i \in S_{c_{e}}$.
4. if $p \in \Pi\left(e^{\prime}\right)$ and $\left.p\right\rangle p^{\prime}$ for some $p^{\prime} \in u_{i} \cap \Pi\left(e^{\prime}\right), i \leqslant k$ where $u_{i}$ is minimal for $\leqslant_{c}$, then $\left.e^{\prime}\right|_{p} \subseteq S(e)$, where $\left.S(e) \stackrel{\text { def }}{=} \bigcup_{q \in \Pi(e)} e\right|_{q}$.
If $k=0$, i.e., no rule has been applied, then 1,2 and 4 are true. 3 is true by construction of $S_{c_{e}}$ and $\simeq_{c_{e}}$.
Assume it is proved for $k$ and let us prove the property for $k+1$. We consider $e^{\prime \prime}$ the result of $e^{\prime}$ by $N_{u_{k+1}}$ where $e^{\prime}$ is the result of $e$ by the application of $N_{u_{1}} \ldots N_{u_{k}}$.

$$
e^{\prime \prime}=e^{\prime}\left[\bigcap_{\substack{\left.p^{\prime} \in u_{k+1} \cap \Pi\left(e^{\prime}\right) \\ e^{\prime \prime \prime} \in e^{\prime}\right|_{p^{\prime}}}}^{e^{\prime \prime \prime}}\right]_{p \in u_{k+1} \cap \Pi\left(e^{\prime}\right)}
$$

Consider $q, q^{\prime} \in u_{i} \cap \Pi\left(e^{\prime \prime}\right), i\langle k+1$.

- either $q$ is incomparable with the paths of $u_{k+1} \cap \Pi(e)$, then $\left.e^{\prime \prime}\right|_{q}=\left.e^{\prime}\right|_{q}$ and, by induction hypothesis, $\left.e\right|_{q}$ is a singleton.
- or there exists $p \in u_{k+1} \cap \Pi(e)$ such that $q \geqslant p$, i.e. $q=p \cdot q_{1}$, then

$$
\left.e^{\prime \prime}\right|_{q}=e^{\prime}\left[\bigcap_{\substack{p^{\prime} \in u_{k+1} \cap \Pi\left(e^{\prime}\right) \\ e^{\prime \prime \prime}\left|q_{1} \in e^{\prime}\right| p_{p^{\prime}}}} e^{\prime \prime \prime}\right]_{p \in u_{k+1} \cap \Pi\left(e^{\prime}\right)} .
$$

By construction of our equivalence relation, since $p \simeq p^{\prime}$ and $p \cdot q_{1} \in u_{i}$, we have $p^{\prime} \cdot q_{1} \in u_{i}$. Thus, by induction hypothesis $\left.e^{\prime}\right|_{p^{\prime} \cdot q_{1}}=\left.e^{\prime}\right|_{p \cdot q_{1}}$ and is a singleton, thus $\left.e^{\prime \prime}\right|_{q}=\left.e^{\prime}\right|_{q}$.

- or there exists $p \in u_{k+1} \cap \Pi(e)$ such that $q\langle p$, which is impossible by the choice of the order on the equivalence classes.
Conclusion: in any cases, we have that $\left.e^{\prime \prime}\right|_{q}=\left.e^{\prime}\right|_{p}$ is a singleton and $\left.e^{\prime \prime}\right|_{q}=\left.e^{\prime \prime}\right|_{q^{\prime}}$. Assume $p\rangle q$ for some $p \in \Pi\left(e^{\prime \prime}\right)$ and assume $u_{i}$ is minimal w.r.t. $\leqslant_{c}$, then $\left.e^{\prime \prime}\right|_{q}=\left.e^{\prime}\right|_{p} \subseteq S(e)$ by induction.
Consider $q, q^{\prime} \in u_{k+1} \cap \Pi\left(e^{\prime \prime}\right)$. Assume $q \notin \Pi\left(e^{\prime}\right)$. Then $q$ is a path created by application of $N_{u_{k+1}}$. This means that there exists $p \in u_{k+1} \cap \Pi\left(e^{\prime}\right)$ such that $p$ is a strict prefix of $q$ which contradicts $G_{f}$ acyclic.
Thus $q, q^{\prime}$ are in $u_{k+1} \cap \Pi\left(e^{\prime}\right)$, thus by construction

$$
\left.e^{\prime \prime}\right|_{q}=\left.e^{\prime \prime}\right|_{q^{\prime}}=\bigcap_{\substack{\left.p^{\prime} \in u_{k+1} \cap \Pi\left(e^{\prime}\right) \\ e^{\prime \prime \prime} \in e^{\prime}\right|_{p^{\prime}}}} e^{\prime \prime \prime}
$$

is a singleton. Assume $p\rangle q$, i.e. $p=q \cdot p_{1}$, for some $p \in \Pi\left(e^{\prime \prime}\right)$ and assume $u_{i}$ is minimal w.r.t. $\leqslant_{c}$, then for all $p^{\prime} \in u_{k+1} \cap \Pi\left(e^{\prime}\right)$, we have $\left.e^{\prime}\right|_{p^{\prime}}=\left.e\right|_{p^{\prime}}$, since by minimality of $u_{k+1}$ no rule can have been applied above $p^{\prime}$ for $p^{\prime} \in u_{k+1} \cap \Pi\left(e^{\prime}\right)$. Thus

$$
\left.e^{\prime \prime}\right|_{p}=\left.\bigcup_{\substack{\left.p^{\prime} \in u_{k+1} \cap \Pi(e) \\ e^{\prime \prime \prime} \in e\right|_{p^{\prime}}}}^{\bigcup} e^{\prime \prime \prime}\right|_{p_{1}} \subseteq S(e)
$$

It remains to prove 3 : assume $\left.e^{\prime \prime}\right|_{q}=f^{c}\left(e_{1}, \ldots, e_{n}\right) \cap e^{4}$. Then, either $q$ is incomparable with the paths of $u_{k+1} \cap \Pi\left(e^{\prime}\right)$, then $\left.e^{\prime \prime}\right|_{q}=\left.e^{\prime}\right|_{q}$ and we can apply the induction hypothesis. Or $q\left\langle p\right.$ for some $p \in u_{k+1} \cap \Pi\left(e^{\prime}\right)$, then $\left.\left.e^{\prime}\right|_{q}=f^{c}\left(e_{1}^{\prime}, \ldots, e_{n}^{\prime}\right) \cap e^{4^{\prime}}\right)$ and 3 is ensured by induction. Or (last case) $q \geqslant p$ for some $p \in u_{k+1} \cap \Pi\left(e^{\prime}\right)$, then $q=p \cdot q_{1}$ and there exists $p^{\prime} \in u_{k+1} \cap \Pi\left(e^{\prime}\right)$ such that $\left.e^{\prime \prime}\right|_{q}=\left.e^{\prime}\right|_{p^{\prime} \cdot q_{1}}$. Thus, by induction hypothesis $p^{\prime} \cdot c \in \simeq_{c_{e}}$ and for every $1 \leqslant i \leqslant n, p^{\prime} \cdot i \in S_{c_{e}}$. Now, by construction of $\simeq_{c_{e}}$ and $S_{c_{e}}$, since $p \simeq_{c_{e}} p^{\prime}$, we have $p \cdot c \in \simeq_{c_{e}}$ and for every $1 \leqslant i \leqslant n, p \cdot i \in S_{c_{e}}$.

Now consider the last property. $c_{e}$ contains the identities $p=p$ for $p \in \Pi(e)$. Hence, if $\left.f^{c}\left(e_{1}, \ldots, e_{n}\right) \cap f^{c^{\prime}}\left(e_{1}^{\prime}, \ldots, e_{n}^{\prime}\right) \cap e^{\prime \prime} \in e^{\prime}\right|_{p}$, then $e_{i}=e_{i}^{\prime}$ is the intersection of expressions $\left.e\right|_{p^{\prime}}$ such that $p \cdot i \simeq_{c_{e}} p^{\prime}$ (note that $e_{i},\left.e_{i}^{\prime} \in e^{\prime}\right|_{p \cdot i}$.

Thanks to Lemma 67, we have the required properties:
Corollary 68. If $e^{\prime}$ is the result of $e$ by the application of $\left(N_{1}\right)$, then for every equality test, if $p \sim_{c} q$ then the expressions at positions $p$ and $q$ in $e^{\prime}$ are identical. In addition, in any expression $g_{-}^{c}\left(\ldots, e_{i}, \ldots\right)$ such that, for some $c_{1}, i \cdot c_{1} \subseteq c$, if $e_{i}=f^{c^{\prime}}\left(\overrightarrow{e^{\prime}}\right) \cap e^{\prime \prime}$, then $P\left(c_{1}\right) \subseteq \Pi\left(f^{c^{\prime}}\left(\overrightarrow{e^{\prime}}\right)\right)$.

Example 69. Consider

$$
e=f^{1=21 \wedge 3=41}(X, g(X), Y, g(Y)) \cap f^{2=31}\left(X_{1}, Z, g(Z), X_{2}\right)
$$

$$
\begin{aligned}
& e \xrightarrow{N_{u_{1}}} e_{1} \stackrel{\text { def }}{=} f^{1=21 \wedge 3=41}\left(X \cap X_{1}, g\left(X \cap X_{1}\right), Y, g(Y)\right) \\
& \cap f^{2=31}\left(X \cap X_{1}, Z, g(Z), X_{2}\right) \\
& e_{1} \xrightarrow{N_{u_{2}}} e_{2} \stackrel{\text { def }}{=} f^{1=21 \wedge 3=41}\left(X \cap X_{1}, Z \cap g\left(X \cap X_{1}\right), Y, g(Y)\right) \\
& \cap f^{2=31}\left(X \cap X_{1}, Z \cap g\left(X \cap X_{1}\right), g\left(Z \cap g\left(X \cap X_{1}\right)\right), X_{2}\right) \\
& e_{2} \xrightarrow{N_{u_{3}}} e_{3} \xrightarrow{\text { def }} f^{1=21 \wedge 3=41}\left(X \cap X_{1}, Z \cap g\left(X \cap X_{1}\right), Y \cap g\left(Z \cap g\left(X \cap X_{1}\right)\right),\right. \\
& \left.g\left(Y \cap g\left(Z \cap g\left(X \cap X_{1}\right)\right)\right)\right) \\
& \cap f^{2=31}\left(X \cap X_{1}, Z \cap g\left(X \cap X_{1}\right), Y \cap g\left(Z \cap g\left(X \cap X_{1}\right)\right), X_{2}\right) \\
& e_{3} \xrightarrow{N_{u_{3}}} e_{3} \stackrel{\text { def }}{=} f^{1=21 \wedge 3=41}\left(X \cap X_{1}, Z \cap g\left(X \cap X_{1}\right), Y \cap g\left(Z \cap g\left(X \cap X_{1}\right)\right),\right. \\
& \left.X_{2} \cap g\left(Y \cap g\left(Z \cap g\left(X \cap X_{1}\right)\right)\right)\right) \\
& \cap f^{2=31}\left(X \cap X_{1}, Z \cap g\left(X \cap X_{1}\right), Y \cap g\left(Z \cap g\left(X \cap X_{1}\right)\right),\right. \\
& \left.X_{2} \cap g\left(Y \cap g\left(Z \cap g\left(X \cap X_{1}\right)\right)\right)\right)
\end{aligned}
$$

Fig. 12. Reduction of $e$ by $\left(N_{1}\right)$.

Then $c_{e} \xlongequal{\text { def }} 1=21 \wedge 3=41 \wedge 2=31$ and the equivalence classes of $\simeq_{c_{e}}$ are:

$$
\begin{aligned}
& u_{1}=\{1,2 \cdot 1,3 \cdot 1 \cdot 1,4 \cdot 1 \cdot 1 \cdot 1\}, u_{2}=\{2,3 \cdot 1,4 \cdot 1 \cdot 1\}, u_{3}=\{3.4 \cdot 1\}, \\
& u_{4}=\{4\}
\end{aligned}
$$

The successive applications of $N_{u_{1}}, N_{u_{2}}, N_{u_{3}}$ and $N_{u_{4}}$ are described in Fig. 12.
Note that, now, "every expression in $\left.e\right|_{p}$ is basic" is equivalent to "there is an expression in $\left.e\right|_{p}$ which is basic" since the intersection of a basic expression with any other expression yields a basic expression. That is why, from now on, we may say, by abuse of language that " $\left.e\right|_{p}$ is basic" to mean either of the two above versions.

We consider, in addition to the rule ( $N_{1}$ ), the following "cleaning" rules:

$$
\begin{aligned}
& \left(N_{2}\right) f^{c}\left(e_{1}, \ldots, e_{n}\right) \cap f\left(e_{1}^{\prime}, \ldots, e_{n}^{\prime}\right) \rightarrow f^{c}\left(e_{1} \cap e_{1}^{\prime}, \ldots, e_{n} \cap e_{n}^{\prime}\right) \\
& \left(N_{3}\right) \quad f^{c}\left(e_{1}, \ldots, e_{n}\right) \cap g^{c^{\prime}}\left(e_{1}^{\prime}, \ldots, e_{m}^{\prime}\right) \rightarrow \perp \text { if } f \neq g .
\end{aligned}
$$

Lemma 70. The rules $\left(N_{2}\right),\left(N_{3}\right)$, applied to normal forms w.r.t. ( $N_{1}$ ), preserve the semantics as well as properties 1,2,3 and the properties described in Lemma 67.

Proof. First, by Lemma 67, in any application of ( $N_{2}$ ), we must have $e_{i}=e_{i}^{\prime}$. Then, it is sufficient to notice that $c_{e}$ is unchanged by application of $\left(N_{2}\right)$, hence its application does not trigger $\left(N_{1}\right)$ and preserves the properties of Lemma 67.

Step 2: We start with some properties of equality tests.
Lemma 71. Let $j \cdot c_{1}$ be the subset of $c_{2}$ containing all equalities whose both sides are prefixed by $j$. If $c_{2}$ satisfies the basicity condition in e and e satisfies condition 3 , then $c_{1}$ satisfies the basicity condition in every expression belonging to $\left.e\right|_{j}$.

Proof. Let $p \cdot i_{1} \cdot q \sim_{c_{1}} p^{\prime}, i_{1} \neq i_{2}$ and $p^{\prime} \ngtr{ }_{\text {pref }} p$. Then $j \cdot p \cdot i_{1} \cdot q \sim_{c_{2}} j \cdot p^{\prime}$ and $j \cdot p^{\prime} \not \not_{\text {pref }} j \cdot p$. By basicity of $c_{2}$, either $\left.e\right|_{j \cdot p^{\prime}}$ is basic or $\left.e\right|_{j \cdot p \cdot i_{2}}$ contains basic expressions only or else $j \cdot p \cdot i_{2} \cdot w \sim_{c_{2}} j \cdot p^{\prime}$ for some $w$, which implies $p \cdot i_{2} \cdot w \sim_{c_{1}} p^{\prime}$.

Lemma 72. If $c$ satisfies the basicity condition in $e$ and $c_{1}$ is an equivalence class of $c$, then $c \backslash c_{1}$ satisfies the basicity condition in $e$.

Proof. Let $p \cdot i_{1} \cdot q \sim_{c \backslash c_{1}} p^{\prime}, i_{1} \neq i_{2}$ and $p^{\prime} \ngtr{ }_{\text {pref }} p$. Then $p \cdot i_{1} \cdot q \sim_{c} p^{\prime}$ and, by basicity of $c$, either $\left.e\right|_{p^{\prime}}$ or every expression in $\left.e\right|_{p \cdot i_{2}}$ is basic or else $p \cdot i_{2} \cdot w \sim_{c} p^{\prime}$ for some $w$. In the latter case, since $c_{1}$ is an equivalence class, $p \cdot i_{2} \cdot w \sim_{c \backslash c_{1}} p^{\prime}$.

Now, we use the following two transformation rules:
$\left(N_{4}\right) f^{c \wedge j \cdot c_{1}}\left(\ldots, e_{j}, \ldots\right) \rightarrow f^{c \wedge j \cdot c_{1}}\left(\ldots, e_{j} \cap g^{c_{1}}\left(e_{j}^{1}, \ldots, e_{j}^{k_{j}}\right), \ldots\right)$ if $e_{j}=g^{c^{\prime}}\left(e_{j}^{1}, \ldots, e_{j}^{k_{j}}\right) \cap e^{\prime \prime}$ and $c$ does not contain any test whose both sides are prefixed by $j$.
$\left(N_{5}\right) f^{c \wedge p_{1} \cdot c_{1} \wedge \ldots \wedge p_{n} \cdot c_{n}}(\vec{e}) \rightarrow f^{c}(\vec{e})$
if $p_{1} \cdot c_{1} \cup \ldots p_{n} \cdot c_{n}$ is a union of equivalence classes, every $p_{i}$ is non empty and, for every $i$, there is an expression $e_{i}$ at position $p_{i}$ in $f^{c \wedge \ldots}(\vec{e})$ such that $e_{i}=g^{c_{i}^{\prime}}\left(\overrightarrow{e_{i}^{\prime}}\right) \cap e_{i}^{\prime \prime}$ and $c_{i}^{\prime} \vDash c_{i}$. (In words: we may remove classes which are consequences of equality tests lower in the expression).

Lemma 73. ( $N_{4}$ ) and ( $N_{5}$ ) preserve conditions 3,1 and 2 . An expression which is unchanged by application of these two rules satisfies condition 4. Moreover, the size (w.r.t. $\mathcal{F}$ ) is preserved by the two rules, the size (w.r.t. t) is reduced by the second rule and, using repeatedly the first rule in an expression e results in an expression $e^{\prime}$ such that $\left|e^{\prime}\right|_{t} \leqslant|e|_{t}^{2}$.

Proof (sketch). By Lemmas 71 and 72, these transformations preserve condition 1 and they preserve trivially condition 3 .

Condition 2 is also preserved since we did not merge any equality test so far.
The satisfaction of condition 4 follows from an inspection of the expressions which are left unchanged by any application of these rules.

The preservation of $|e|_{\mathcal{F}}$ follows from the definition. If $c$ yields a new test $c^{\prime}$, possibly after repeated applications of $\left(N_{4}\right)$, then there is a $p_{1}$ such that $c^{\prime}$ consists in equalities $p=q$ such that $p_{1} \cdot p \sim_{c} p_{1} \cdot q$. If we fix the size of $p_{1}$, then the sum of sizes of such $c^{\prime}$ is bounded by $|c|$. Hence the total size of the new tests is bounded by $|c|^{2}$.

Step 3: The purpose of this step is to show how to satisfy in addition condition 5, while preserving properties $3,1,4,2$. In what follows, integers $i, j \ldots$ are always assumed range over a finite set $1 . . n$ which is consistent with the arity of function symbols.

Let $e$ be an expression $f^{c}\left(\vec{e}_{1}\right) \cap f^{c^{\prime}}\left(\overrightarrow{e_{2}}\right) \cap \ldots$ If $e$ is in normal form w.r.t. $\left(N_{i}\right), i \leqslant 5$, then we may assume that $\overrightarrow{e_{1}}=\overrightarrow{e_{2}}$. Indeed, this is true of normal forms w.r.t. $\left(N_{1}\right),\left(N_{2}\right),\left(N_{3}\right)$ thanks to Lemmas 67 and 70, and such a property is trivially preserved by the rules


Fig. 13. A representation of $Q\left(c_{0}, c^{\prime}\right)$ and $Q_{m}\left(c_{0}, c^{\prime}\right)$.
$\left(N_{4}\right),\left(N_{5}\right)$. We will however only assume in what follows the weaker property $\Pi\left(f^{c}\left(\vec{e}_{1}\right)\right)=$ $\Pi\left(f^{c^{\prime}}\left(\overrightarrow{e_{2}}\right)\right)$.

We define $c \sqcap c^{\prime}$ (relatively to $e$ ) as follows: first, if $c \wedge c^{\prime}$ is unsatisfiable, we replace it with $\perp$. Otherwise, for every nontrivial equivalence class $c_{0}$ for $c$, let

$$
Q\left(c_{0}, c^{\prime}\right)=\left\{w\left|\exists p \in P\left(c_{0}\right), p \cdot w \in P\left(c^{\prime}\right), e\right|_{p \cdot w} \text { not basic }\right\}
$$

and

$$
\begin{aligned}
Q_{m}\left(c_{0}, c^{\prime}\right)= & \left\{w \cdot i \mid \exists w^{\prime} \in Q\left(c_{0}, c^{\prime}\right), w\left\langle_{\text {pref }} w^{\prime}, \forall w^{\prime} \in Q\left(c_{0}, c^{\prime}\right)\right.\right. \\
& \left.\times w \cdot i \wedge_{\text {pref }} w^{\prime}\right\}
\end{aligned}
$$

Intuitively, $Q_{m}\left(c_{0}, c^{\prime}\right)$ is the border of $Q\left(c_{0}, c^{\prime}\right)$ with its maximal elements. See Fig. 13.
Each time $Q\left(c_{0}, c^{\prime}\right)$ is empty, we let $\overline{c_{0}}=c_{0}$ and, otherwise:

$$
\bar{c}_{0} c^{\prime} \stackrel{\text { def }}{=} \bigwedge_{p \sim}^{\sim_{c_{0}} q} \bigwedge_{w \in Q_{m}\left(c_{0}, c^{\prime}\right)} p \cdot w=q \cdot w
$$

Then

$$
\bar{c}^{c^{\prime}} \stackrel{\text { def }}{=} \bigwedge_{c_{0} \text { a class of } c}{\overline{c_{0}}}^{c^{\prime}}
$$

We define now the sequence $c_{n}$ as follows: $c_{1}=c, c_{1}^{\prime}=c^{\prime}$ and $c_{n+1} \stackrel{\text { def }}{=} \overline{c_{n}} \overline{c_{n}^{\prime}}$ and $c_{n+1}^{\prime} \stackrel{\text { def }}{=} \overline{c_{n}^{\prime}} \overline{c_{n}}$.

Note that if $p \cdot w=q \cdot w \in \bar{c}^{c^{\prime}}$, then $p \cdot w, q \cdot w \in \Pi\left(f^{c}\left(\vec{e}_{1}\right)\right)=\Pi\left(f^{c^{\prime}}\left(\overrightarrow{e_{2}}\right)\right)$. Moreover, if $c \neq \bar{c}$, then $|\bar{c}|\rangle|c|$. It follows that the sequences $c_{n}$ and $c_{n}^{\prime}$ are ultimately stationary: let $c_{\infty}$ and $c_{\infty}^{\prime}$ be the respective limit values of $c_{n}$ and $c_{n}^{\prime}$. We define

$$
c \sqcap c^{\prime} \stackrel{\text { def }}{=} c_{\infty} \wedge c_{\infty}^{\prime}
$$

Example 74. Let us consider $c \stackrel{\text { def }}{=} 1=2$ and $c^{\prime} \stackrel{\text { def }}{=} 211=3 \wedge 11=4$ and assume that every position of $c$ or $c^{\prime}$ is not basic. Then $c \sqcap c^{\prime}$ is computed in two steps:
Step 1: $Q\left(c, c^{\prime}\right)=\{11,1\}$ thus $Q_{m}\left(c, c^{\prime}\right)=\{11\}$.

$$
\begin{aligned}
& Q\left(c^{\prime 1}, c\right)=Q\left(c^{\prime 2}, c\right)=\emptyset \text { where } c^{\prime 1}=211=3 \text { and } c^{\prime 2}=11=4 \\
& \text { Thus } c_{2} \stackrel{\text { def }}{=} \bar{c}^{c^{\prime}}=111=211 \text { and } c_{2}^{\prime} \stackrel{\text { def }}{=}{\overline{c^{\prime}}}^{c}=c^{\prime}=211=3 \wedge 11=4
\end{aligned}
$$

Step 2: $Q\left(c_{2}, c_{2}^{\prime}\right)=\emptyset$.
Let $c_{2}^{1^{\prime}} \stackrel{\text { def }}{=} 211=3$ and $c_{2}^{2^{\prime}} \stackrel{\text { def }}{=} 11=4$, then $Q\left(c_{2}^{1^{\prime}}, c_{2}\right)=\emptyset$ and $Q\left(c_{2}^{2^{\prime}}, c_{2}\right)=\{1\}$
thus $Q_{m}\left(c_{2}^{2^{\prime}}, c_{2}\right)=\{1\}$.
Thus $c_{3} \stackrel{\text { def }}{=} \bar{c}_{2} c_{2}^{\prime}=c_{2}=111=211$ and $c_{3}^{\prime} \stackrel{\text { def }}{=} \overline{c_{2}^{\prime}}{ }^{c_{2}}=211=3 \wedge 111=41$.
$\bar{c}_{3}^{c_{3}^{\prime}}=c_{3},{\overline{c_{3}^{\prime}}}^{c_{3}}=c_{3}^{\prime}$, thus $c \sqcap c^{\prime} \xlongequal{\text { def }} 111=211 \wedge 211=3 \wedge 111=41$.
Let us analyze a bit more precisely the complexity: for every pair $p \sim_{\subset \sqcap c^{\prime}} q$, there is a position $r \in P(c) \cup P\left(c^{\prime}\right)$ such that $p=p_{1} \cdot i$ and $r=p_{1} \cdot r_{1}$. Then, if $a$ is the maximal arity of a function symbol, then

$$
\left|P\left(c \sqcap c^{\prime}\right)\right| \leqslant\left(|c|+\left|c^{\prime}\right|\right) \times a,
$$

where $a$ is the maximal arity of a function symbol and since the number of possible choices for $p_{1}$ is bounded by $|c|+\left|c^{\prime}\right|$. It follows that

$$
\left|c \sqcap c^{\prime}\right| \leqslant\left(|c|+\left|c^{\prime}\right|\right) \times M\left(c, c^{\prime}\right) \times a \leqslant a \times\left(|c|+\left|c^{\prime}\right|\right)^{2},
$$

where $M\left(c, c^{\prime}\right)$ is the maximal length of a position in $P(c) \cup P\left(c^{\prime}\right)$.
Then we use the following transformation rule:

$$
\left(N_{6}\right) f^{c}\left(e_{1}, \ldots, e_{n}\right) \cap f^{c^{\prime}}\left(e_{1}^{\prime}, \ldots, e_{n}^{\prime}\right) \rightarrow f^{c \sqcap c^{\prime}}\left(e_{1} \cap e_{1}^{\prime}, \ldots, e_{n} \cap e_{n}^{\prime}\right)
$$

if $\Pi\left(f^{c}(\vec{e})\right)=\Pi\left(f^{c^{\prime}}\left(\overrightarrow{e^{\prime}}\right)\right)$.
First, $P(c), P\left(c^{\prime}\right) \subseteq \Pi(f(\vec{e}))$, thanks to property 3. It follows that $P\left(c \sqcap c^{\prime}\right) \subseteq \Pi(f(\vec{e}))$. The semantics is also preserved since, as long as all positions $p \cdot i$ and $q \cdot i$ are in $\Pi\left(f^{c}(\vec{e})\right)$, and the top symbols at positions $p$ and $q$ are identical, an equality test $p=q \in c$ is equivalent to the conjunction of equality tests $p \cdot i=q \cdot i$.

Condition 3 is also trivially satisfied. Remains to verify the preservation of the other ones: condition 4 is shown to be preserved in Lemma 75, condition 1 in Lemma 76, condition 2 in Lemma 77.

Lemma 75. The rule preserves property 4.
Proof. Actually, every $\overline{c_{0}}$ and $\overline{c_{0}^{\prime}}$ satisfy property 4 . Indeed, if, in the class $c_{0}$ of $p$ (in $\left.c\right), q$ does not share any prefix with $p$, then in the class of $p \cdot w$ (w.r.t. $\overline{c_{0}}$ ), $q \cdot w$ does not share any prefix with $p \cdot w$.

Lemma 76. The transformation preserves the basicity condition.
Proof. It suffices to show that $\bar{c} \wedge \overline{c^{\prime}}$ satisfies the basicity condition, whenever $c, c^{\prime}$ do. Then we use an induction on the fixed point computation for $c \sqcap c^{\prime}$.
Assume $p \cdot i \cdot q \sim_{\bar{c} \wedge \overline{c^{\prime}}} p^{\prime}, p^{\prime} \not ¥_{\text {pref }} p, j \neq i$ and, for every $w, p \cdot j \cdot w \not_{\bar{c} \wedge \overline{c^{\prime}}} p^{\prime}$. Then $p$ is not empty. Assume w.l.o.g that $p \cdot i \cdot q, p^{\prime} \in P\left(\overline{c_{0}}\right)$ where $c_{0}$ is an equivalence class of $c$. (If this is not the case, exchange the roles of $c$ and $c^{\prime}$ ). Then, by Lemma 75 , there is a $p_{1}^{\prime} \sim_{c_{0}} p \cdot i \cdot q$ such that $p$ is not a prefix of $p_{1}^{\prime}$. Hence we may assume w.l.o.g. that $p$ is not a prefix of $p^{\prime}$ (possibly after replacing $p^{\prime}$ with some $p_{1}^{\prime}$ ).

If $p \cdot i \cdot q \sim_{c} p^{\prime}$, then the result follows from the basicity property of $c$ : in such a case, we must have $\overline{c_{0}}=c_{0}$ (since, in any case, either $\overline{c_{0}}=c_{0}$ or $P\left(\overline{c_{0}}\right) \cap P\left(c_{0}\right)=\emptyset$ ), hence $p \cdot j \cdot w \sim_{c} p^{\prime}$ iff $p \cdot j \cdot w \sim_{\bar{c} \wedge \overline{c^{\prime}}} p^{\prime}$.

Let us assume now that this is not the case: $\overline{c_{0}} \neq c_{0}, p \cdot i \cdot q=p_{0} \cdot w_{0} \cdot i_{0}$ with $w_{0} \cdot i_{0} \in Q_{m}\left(c_{0}, c^{\prime}\right)$ and $p_{0} \sim_{c} q_{0}$.

$$
\begin{aligned}
& p \cdot i \cdot q=p_{0} \cdot w_{0} \cdot i_{0} \\
& p^{\prime}=q_{0} \cdot w_{0} \cdot i_{0} \\
& p_{0} \sim_{c} q_{0} \sim_{c} p_{0}^{\prime} \\
& p_{0}^{\prime} \cdot p_{1} \sim_{c^{\prime}} q_{1} \\
& \left.e\right|_{p_{0}^{\prime} \cdot p_{1}} \text { is not basic }
\end{aligned} \text { and }
$$

In addition, we may assume, thanks to property 4 again that $p_{0}$ and $q_{0}$ on one hand and $p_{0}^{\prime}$ and $q_{1}$ on the other hand do not share any nontrivial prefix.
Case 1: $p\left\langle_{\text {pref }} p_{0}\right.$. There is a $q^{\prime}$ such that $p_{0}=p \cdot i \cdot q^{\prime}$. We use the basicity property of $c$, considering the equivalence $p \cdot i \cdot q^{\prime} \sim_{c} q_{0}$ : either $\left.e\right|_{p \cdot j}$ is basic or $\left.e\right|_{p_{0}}$ is basic or $p \cdot j \cdot w \sim_{c} p_{0}$ for some $w$. In the first two cases, we get what we want $\left(\left.e\right|_{p_{0}}\right.$ basic implies $e_{p \cdot i \cdot q}$ basic). In the last case, $p \cdot j \cdot w \cdot w_{0} \cdot i_{0} \sim_{\overline{c_{0}}} p^{\prime}$ by construction, hence contradicting the hypothesis $\forall w^{\prime} \cdot p \cdot j \cdot w^{\prime} \chi_{\bar{c} \wedge \overline{c^{\prime}}} p^{\prime}$.
Case 2: $\left.p_{0} \cdot w_{0}\right\rangle_{\text {pref }} p \geqslant_{\text {pref }} p_{0}$. Since $w_{0} \cdot i_{0} \ell_{\text {pref }} p_{1}$ and $w_{0} \leqslant_{\text {pref }} p_{1}, p_{1}=w_{0} \cdot j_{0} \cdot \alpha$ for some $j_{0} \neq i_{0}$ and $\alpha$. We apply now the basicity property of $c^{\prime}$, considering the equivalence $p_{0}^{\prime} \cdot w_{0} \cdot j_{0} \cdot \alpha \sim_{c^{\prime}} q_{1}$ (recall that $p_{0}^{\prime} \cdot p_{1} \sim_{c^{\prime}} q_{1}$ ). Since $\left.e\right|_{p_{0}^{\prime} \cdot p_{1}}$ is not basic, only two cases can occur:
Case 2.1: $\left.e\right|_{p_{0}^{\prime} \cdot w_{0} \cdot i_{0}}$ is basic which implies $\left.e\right|_{p_{0} \cdot w_{0} \cdot i_{0}}$ basic, hence the desired conclusion.
Case 2.2: $p_{0}^{\prime} \cdot w_{0} \cdot i_{0} \cdot w \sim_{c^{\prime}} q_{1} \sim_{c^{\prime}} p_{0}^{\prime} \cdot p_{1}$ for some $w$. Let $p=p_{0} \cdot p_{2}, w_{0}=p_{2} \cdot i \cdot w_{1}$. Then $p_{0}^{\prime} \cdot p_{2} \cdot i \cdot w_{1} \cdot i_{0} \cdot w \sim_{c^{\prime}} q_{1}$ and, since $i \neq j$, by the basicity of $c^{\prime}$, either $e_{p_{0}^{\prime} \cdot p_{2} \cdot j}$ is basic or $\left.e\right|_{q_{1}}$ is basic or else there is a $w_{2}$ such that $p_{0}^{\prime} \cdot p_{2} \cdot j \cdot w_{2} \sim_{c^{\prime}} q_{1}$.

In the first case, $\left.e\right|_{p_{0} \cdot p_{2} \cdot j}$ is also basic (i.e. $\left.e\right|_{p \cdot j}$ is basic) and we conclude.
The second case contradicts the hypothesis that $\left.e\right|_{p_{0}^{\prime} \cdot p_{1}}$ is not basic.
In the third case, $p_{0}^{\prime} \cdot p_{2} \cdot j \cdot w_{2} \sim_{c^{\prime}} p_{0}^{\prime} \cdot p_{1} \sim_{c^{\prime}} p_{0}^{\prime} \cdot w_{0} \cdot i_{0}$. Then, by construction, $p_{0} \cdot w_{0} \cdot i_{0} \sim_{\bar{c} \wedge \overline{c^{\prime}}} p_{0} \cdot p_{2} \cdot j \cdot w_{2}=p \cdot j \cdot w_{2}$, which contradicts again the hypothesis. Case 3: $p=p_{0} \cdot w_{0}, i=i_{0}$ and $q$ is empty. $p_{1}=w_{0} \cdot j_{0} \cdot \alpha$ with $j_{0} \neq i_{0}$. If $j=j_{0}$, from $\left.e\right|_{p_{0}^{\prime} \cdot p_{1}}$ is not basic, we conclude the desired result. Otherwise, $j \neq j_{0}$ and we use the basicity property of $c^{\prime}$, considering again $p_{0}^{\prime} \cdot w_{0} \cdot j_{0} \cdot \alpha \sim_{c^{\prime}} q_{1}, j \neq j_{0}$. Either $\left.e\right|_{p_{0}^{\prime} \cdot w_{0} \cdot j}$ is basic (in which case we also conclude) or there is $w_{1}^{\prime} p_{0}^{\prime} \cdot p_{1} \sim_{c^{\prime}} p_{0}^{\prime} \cdot w_{0} \cdot j \cdot w_{1}^{\prime}$.

In the latter case, we use again the basicity property on $c^{\prime}$, considering $p_{0}^{\prime} \cdot w_{0} \cdot j_{0} \cdot \alpha \sim_{c^{\prime}}$ $q_{1}, i \neq j_{0}$. Either $\left.e\right|_{p_{0}^{\prime} \cdot w_{0} \cdot i_{0}}$ is basic, in which case we conclude, or else there is a $w_{2}^{\prime}$ such that $p_{0}^{\prime} \cdot p_{1} \sim_{c^{\prime}} p_{0}^{\prime} \cdot w_{0} \cdot i_{0} \cdot w_{2}^{\prime}$. Now, let us recall that, by construction, $\forall w^{\prime} \in Q\left(c_{0}, c^{\prime}\right)$, $w_{0} \cdot i_{0} \Lambda_{\text {pref }} w^{\prime}$. Since $w_{0} \cdot i_{0} \cdot w_{1}^{\prime} \in Q\left(c_{0}, c^{\prime}\right)$, we must have $w_{1}^{\prime}$ empty.
$\left.e\right|_{p_{0}^{\prime} \cdot w_{0} \cdot i_{0}}$ is basic (in which case we conclude) or
It follows that $p_{0}^{\prime} \cdot w_{0} \cdot i_{0} \sim_{c^{\prime}} p_{0}^{\prime} \cdot w_{0} \cdot j \cdot w_{2}^{\prime}$, which contradicts $p \cdot i \not \chi_{\bar{c} \wedge \overline{c^{\prime}}} p \cdot j \cdot w_{2}^{\prime}$.

Lemma 77. The transformation $\left(N_{6}\right)$ preserves property 2.

Proof. We prove, by induction on $n$ that

1. for every $p \cdot q, p \in P\left(c_{n}\right) \cup P\left(c_{n}^{\prime}\right)$ such that $q$ is not empty, either $\left.e\right|_{p \cdot q}$ is a basic expression or else $c_{n+1} \neq c_{n}$ or $c_{n+1}^{\prime} \neq c_{n}^{\prime}$.
2. $c_{n}$ and $c_{n}^{\prime}$ (individually) satisfy property 2 .

The lemma will follow.
For every $n$, if $p \in P\left(c_{n}\right)$ and if $p \cdot q \in P\left(c_{n}^{\prime}\right)$ and if $\left.e\right|_{p \cdot q}$ is not basic, then $q \in Q\left(\gamma_{0}, c_{n}^{\prime}\right)$ by definition ( $\gamma_{0}$ is the equivalence class of $p$ ) and, since $q$ is not empty, a suffix of $q$ belongs to $Q_{m}\left(\gamma_{0}, c_{n}^{\prime}\right)$. It follows that $\bar{\gamma}_{0}^{c_{n}^{\prime}} \neq \gamma_{0}$ and therefore $c_{n+1} \neq c_{n}$. So, from now on we may assume w.l.o.g that both $p$ and $p \cdot q$ are in $P\left(c_{n}\right)$.

In the base case, $c_{n}=c$ and $c_{n}^{\prime}=c^{\prime}$ and $c, c^{\prime}$ satisfy condition 2. Then, if $p, p \cdot q \in P(c)$, $\left.e\right|_{p \cdot q}$ is basic.

Consider now the induction step and let $p \in P\left(\bar{\gamma}_{0}^{c_{n}^{\prime}}\right), p \cdot q \in P\left(\bar{\gamma}_{1}^{c_{n}^{\prime}}\right), q$ is not empty and $\gamma_{0}, \gamma_{1}$ are classes of $c_{n}$.
Assume first that $\bar{\gamma}_{0}^{c_{n}^{\prime}} \neq \gamma_{0}$. Then, according to the definition of $c_{n+1}$, there are positions such that:

$$
\begin{aligned}
& p=p_{1} \cdot w \cdot i \\
& w\left\langle_{\text {pref }} w^{\prime}\right. \\
& p_{2} \sim_{c_{n}} p_{1} \\
& p_{2} \cdot w^{\prime} \in P\left(c_{n}^{\prime}\right) \\
& \left.e\right|_{p_{2} \cdot w^{\prime}} \text { is not basic } \\
& \forall w^{\prime \prime} \in Q\left(\gamma_{0}, c_{n}^{\prime}\right), w \cdot i \wedge_{\text {pref }} w^{\prime \prime}
\end{aligned}
$$

We consider now several cases for $p \cdot q$.
Case 1: $p \cdot q \in P\left(\bar{\gamma}_{1}^{c_{n}^{\prime}}\right), \bar{\gamma}_{1}^{c_{n}^{\prime}}=\gamma_{1} \subseteq c_{n}$. Then $p_{1}\left\langle_{\text {pref }} p \cdot q\right.$ and we conclude using property 2 on $c_{n}$ (induction hypothesis).
Case 2: $p \cdot q \in P\left(\overline{\gamma_{1}}\right), \overline{\gamma_{1}} \neq \gamma_{1} \subseteq c_{n}$. Then, by definition, there are positions such that:

$$
\begin{aligned}
& p \cdot q=p_{3} \cdot w_{1} \cdot i_{1} \\
& w_{1}\left\langle_{\text {pref }} w_{1}^{\prime}\right. \\
& p_{4} \sim_{c_{n}} p_{3} \\
& p_{4} \cdot w_{1}^{\prime} \in P\left(c_{n}^{\prime}\right) \\
& \left.e\right|_{p 4} \cdot w_{1}^{\prime} \text { is not basic } \\
& \forall w_{1}^{\prime \prime} \in Q\left(\gamma_{1}, c_{n}^{\prime}\right), w_{1} \cdot i_{1}\left\langle_{\text {pref }} w_{1}^{\prime \prime}\right.
\end{aligned}
$$

In this case, $p \cdot q=p_{3} \cdot w_{1} \cdot i_{1}=p_{1} \cdot w \cdot i \cdot q$, hence $p_{3}$ and $p_{1}$ must be comparable w.r.t. the prefix ordering. If they are distinct, assume $p_{k}$ is the largest one, then, by property 2 on $c_{n},\left.e\right|_{p_{k}}$ must be basic, hence $\left.e\right|_{p}$ or $\left.e\right|_{p \cdot q}$ must be basic and we conclude. Otherwise, $p_{1}=p_{3}$ and $\gamma_{0}=\gamma_{1}$. By hypothesis, $\forall w^{\prime \prime} \in Q\left(\gamma_{0}, c_{n}^{\prime}\right), w \cdot i \Lambda_{\text {pref }} w^{\prime \prime}$, thus $w^{\prime \prime}=$ $w \cdot i \cdot q \notin Q\left(\gamma_{0}, c_{n}^{\prime}\right)$, which can only occur when $\left.e\right|_{p \cdot \cdot w \cdot i \cdot q}=\left.e\right|_{p \cdot q}$ is basic, and we conclude again.
Assume now $\bar{\gamma}_{0} c_{n}^{\prime}=\gamma_{0}$. If $\bar{\gamma}_{1}^{c_{n}^{\prime}}=\gamma_{1}$, we conclude by the induction hypothesis: property
2 holds on $c_{n}$.

Otherwise, there are positions such that:

$$
\begin{aligned}
& p \cdot q=p_{1} \cdot w_{1} \cdot i_{1} \\
& w_{1}\left\langle_{\text {pref }} w_{1}^{\prime}\right. \\
& p_{2} \sim_{c_{n}} p_{1} \\
& p_{2} \cdot w_{1}^{\prime} \in P\left(c_{n}^{\prime}\right) \\
& \left.e\right|_{p_{2} \cdot w_{1}^{\prime}} \text { is not basic } \\
& \forall w_{1}^{\prime \prime} \in Q\left(\gamma_{1}, c_{n}^{\prime}\right), w_{1} \cdot i_{1} \wedge_{\text {pref }} w_{1}^{\prime \prime}
\end{aligned}
$$

$\bar{\gamma}_{0}^{\prime}{ }_{n}^{\prime}=\gamma_{0}$ implies $p_{1} \Lambda_{\text {pref }} p$. If $\left.p_{1}\right\rangle_{\text {pref }} p$, from $p_{1} \sim_{c_{n}} p_{2}$, we conclude that $\left.e\right|_{p_{1}}$ is basic (hence $\left.e\right|_{p \cdot q}$ ), thanks to the induction hypothesis.

We are left to the case $p=p_{1}$. Then $p_{2} \sim_{c_{n}} p$ and $p_{2} \cdot w_{1}^{\prime} \in P\left(c_{n}^{\prime}\right)$ and $w_{1}^{\prime}$ is not empty, hence $\bar{\gamma}_{0}^{c_{n}^{\prime}} \neq \gamma_{0}$, a contradiction.

Lemma 78. Property 5 is satisfied for normal forms w.r.t. $\left(N_{6}\right)$.
Proof. At step 1, we ensured that, for every $p \in \Pi(e),\left.e\right|_{p}$ only contains expressions of the form $f^{c_{1}}(\vec{e}) \cap \ldots \cap f^{c_{n}}(\vec{e}) \cap X_{1} \cap \ldots \cap X_{m}$. The rule ( $N_{6}$ ) imposes $n=1$, hence property 5.

In this step, we did not increase $|e|_{\mathcal{F}}$ and $\left|c \sqcap c^{\prime}\right| \leqslant a \times\left(|c|+\left|c^{\prime}\right|\right)^{2}$, hence $\left|e^{\prime}\right|_{t} \leqslant a \times|e|_{t}^{2}$.
Step 4: The purpose of this last step is to rearrange the equality tests so that there are no overlapping tests except possibly for basic expressions. (More formally, we need to ensure condition 6).

Let us show first some examples of what we want.
Example 79. $e \stackrel{\text { def }}{=} f^{11=12=2}\left(f^{11=12=21=22}(f(X, X), f(X, X)), f(X, X)\right)$. $e$ contains overlapping tests. We can however use first the rules $\left(N_{4}\right),(N 2)$ and get

$$
f^{11=12=2}\left(f^{11=12=21=22}\left(f^{1=2}(X, X), f^{1=2}(X, X)\right), f(X, X)\right)
$$

Now, it turns out that the intermediate test is a consequence of the top one and the lowest ones, and it can be removed, yielding (after normalization w.r.t. ( $N 2$ )):

$$
f^{11=12=2}\left(f\left(f^{1=2}(X, X), f^{1=2}(X, X)\right), f^{1=2}(X, X)\right)
$$

for which there is no overlapping test.
In this example, pictured in Fig. 14, we see that we do not need to change the tests but only to reorganize them.

Example 80. Let

$$
e=f^{111=121=112=122=2}\left(f^{1=2}(g(X, X, Y), g(X, X, Y)), X\right),
$$

which contains overlapping tests. Using the rule $\left(N_{4}\right)$ we get

$$
\begin{aligned}
& f^{c}\left(f ^ { 1 = 2 } ( g ( X , X , Y ) , g ( X , X , Y ) ) \cap f ^ { 1 1 = 1 2 = 2 1 = 2 2 } \left(g^{1=2}(X, X, Y),\right.\right. \\
& \left.\left.\quad \times g^{1=2}(X, X, Y)\right), X\right)
\end{aligned}
$$



Fig. 14. The tests of Example 79.
with $c \stackrel{\text { def }}{=} 111=121=112=122=2$. Using rule $\left(N_{6}\right)$ we get

$$
f^{111=121=112=122=2}\left(f^{11=12=21=22 \wedge 13=23}\left(g^{1=2}(X, X, Y), g^{1=2}(X, X, Y)\right), X\right)
$$

Now, the class $11=12=21=22$ is a consequence of the top and low tests and it can be removed:

$$
f^{111=121=112=122=2}\left(f^{13=23}\left(g^{1=2}(X, X, Y), g^{1=2}(X, X, Y)\right), X\right)
$$

Finally the low tests can also be removed since they are consequences of the top one, yielding:

$$
f^{111=121=112=122=2}\left(f^{13=23}(g(X, X, Y), g(X, X, Y)), X\right)
$$

in which there is a remaining overlapping test. However, in $e, Y$ must be basic (thanks to the basicity condition) and thus the lower positions 13 and 23 correspond to basic expressions. In this example, pictured in Fig. 15, we need to push some tests down.

So, the idea is to first inherit the constraints thanks to rule $\left(N_{4}\right)$ (this has been done at step 2 ), next normalize w.r.t. ( $N_{6}$ ) (this has been done at step 3 ) and finally remove useless tests, which we do now.

$$
\left(N_{7}\right) f^{c_{1}}\left(\vec{e}_{1}\right)\left[g^{c_{2} \wedge c_{0}}\left(\vec{e}_{2}\right) \cap e_{2}^{\prime}\right]_{p_{1}} \rightarrow f^{c_{1}}\left(\vec{e}_{1}\right)\left[g^{c_{2}}\left(\vec{e}_{2}\right) \cap e_{2}^{\prime}\right]_{p_{1}}
$$

If

- $c_{0}$ is an equivalence class in $c_{2} \wedge c_{0}$,

$$
\text { - } c_{1} \wedge p_{1} \cdot \underbrace{g^{\left.c_{2} \wedge c_{0}\left(\vec{e}_{2}\right)\right|_{r}=h^{c_{r}}\left(\vec{e}_{r}\right) \cap e_{r}^{\prime}}}_{\substack{r \in \Pi\left(g_{\begin{subarray}{c}{\left.c^{2} \wedge \wedge_{0} \\
r \neq \varepsilon \\
\left(\vec{e}_{2}\right)\right)} }}\right.}\end{subarray}}
$$

- $e=f^{c_{1}}\left(\vec{e}_{1}\right)\left[g^{c_{2} \wedge c_{0}}\left(\vec{e}_{2}\right) \cap e_{2}^{\prime}\right]_{p_{1}}$,


Fig. 15. The tests of Example 80.

- $\vDash_{e}$ is the consequence relation according to the following rules:
- reflexivity, symmetry and transitivity
- right compatibility: $p=q \vDash_{e} p_{-} \cdot r=q \cdot r$
$\circ$ folding (w.r.t. $e$ ): if $\left.e\right|_{p}=f^{c}\left(\overrightarrow{e^{\prime}}\right) \cap e^{\prime \prime}$ and $f$ has arity $n$, then $p \cdot 1=q \cdot 1 \wedge \ldots \wedge p \cdot n=$ $q \cdot n F_{e} p=q$.
- conjunction introduction:

$$
\left.\begin{array}{l}
c_{1} \vDash_{e} c_{1}^{\prime} \\
c_{2} \vDash_{e} c_{2}^{\prime}
\end{array}\right\} \Rightarrow c_{1} \wedge c_{2} \vDash_{e} c_{1}^{\prime} \wedge c_{2}^{\prime} .
$$

We must be careful on how to apply this rule. Consider the following example
Example 81. $e \stackrel{\text { def }}{=} f^{111=12=2}\left(f^{11=12=2}\left(f^{1=2}\left(g^{\top}(X), g^{\top}(X)\right), g^{\top}(X)\right), g^{\top}(X)\right)$. This expression is in normal form w.r.t. the previous transformations. There are two ways of applying rule $\left(N_{7}\right)$ : we can remove the constraint $1=2$ since $11=12=2 \wedge 11 \cdot \mathrm{~T} \wedge$ $12 \cdot T \vDash 1 \cdot(1=2)$. Then the expression is in normal form for $N_{7}$ and there are still some overlapping tests. The other possibility is to apply $\left(N_{7}\right)$ to $11=12=2: 111=12=$ $2 \wedge 1 \cdot 1 \cdot(1=2) \vDash 1 \cdot(11=12=2)$ and there is no longer any overlapping tests in the expression.

We assume that the previous steps have been completed and use the rule $\left(N_{7}\right)$ top-down.
Lemma 82. ( $N_{7}$ ) (applied top-down) is terminating, it preserves the semantics and the properties 5, 3, 1, 4, 2.

Proof. The termination is straightforward: the size of the expression is strictly decreasing (and the resulting expression $e^{\prime}$ satisfies $\left|e^{\prime}\right|_{\mathcal{F}}=|e|_{\mathcal{F}}$ and $\left|e^{\prime}\right|_{t} \leqslant|e|_{t}$.)

The condition of the rule ensures the preservation of interpretations. Property 5 is preserved since we do not change the term structure of the expression. Property 3 is not necessarily preserved by one-step application of $\left(N_{7}\right)$. However, if $p_{0} \cdot p_{1}=p_{0} \cdot p_{1}^{\prime}$ is checked higher up in the expression, then the expressions at positions $p_{1}$ and $p_{1}^{\prime}$ must be identical (by property 3 ) and the rule ( $N_{7}$ ) will be applied twice to these expressions, yielding removal of $c_{0}$ for both occurrences.

Lemma 72 ensures the preservation of property 1 . Also, property 4 is preserved since we remove an equivalence class and property 2 is preserved since we remove some tests.

Lemma 83. Normal forms w.r.t. rule $\left(N_{7}\right)$ (applied top-down) satisfy condition 6.
Proof. Assume that we are in the situation of property 6: $e=f^{c}\left(e_{1}, \ldots, e_{n}\right) \cap e^{\prime}, p_{1} \cdot p_{2} \sim_{c}$ $q, p_{1}, p_{2}$ are nonempty, $\left.f^{c}\left(e_{1}, \ldots, e_{n}\right)\right|_{p_{1}}=g^{c^{\prime}}\left(e_{1}^{\prime}, \ldots, e_{m}^{\prime}\right) \cap e^{\prime \prime}$ and $p^{\prime} \in P\left(c^{\prime}\right)$.

Assume that there is no position $p_{2}^{\prime}$ such that $p_{1} \cdot p_{2}^{\prime} \sim_{c} p_{1} \cdot p_{2}$ and $p_{2}^{\prime}$ is comparable with $p^{\prime}$ w.r.t. $\geqslant_{\text {pref. }}$. Let $p_{1} \cdot p_{2}^{\prime} \sim_{c} p_{1} \cdot p_{2}$ such that $p_{2}^{\prime}$ is the position which shares the longest prefix with $p^{\prime}$. Then we can write $p^{\prime}=w \cdot j \cdot w^{\prime}$ and $p_{2}^{\prime}=w \cdot i \cdot w^{\prime \prime}$ with $i \neq j$. By condition 1 (for $c$ ), considering $p_{1} \cdot w \cdot i \cdot w^{\prime \prime} \sim_{c} p_{1} \cdot p_{2}$, either $\left.e\right|_{p_{1} \cdot p_{2}}$ is basic or $\left.e\right|_{p_{1} \cdot p^{\prime}}$ is basic or $\exists w_{1}, p_{1} \cdot p_{2} \sim_{c} p_{1} \cdot w \cdot j \cdot w_{1}$. This last case contradicts the maximal shared prefix hypothesis.

We are left to the case where $p^{\prime}$ is comparable w.r.t. $\geqslant_{\text {pref }}$ with some position $p_{2}^{\prime}$ such that $p_{1} \cdot p_{2}^{\prime} \sim_{c} p_{1} \cdot p_{2}$.

If $p^{\prime} \sim_{c^{\prime}} q^{\prime}$ and $q^{\prime}$ does not share any nontrivial prefix with $p^{\prime}$ (this is possible thanks to property 4 on $c^{\prime}$ ), then a similar property holds for $q^{\prime}$ : we assume now that $p^{\prime} \sim_{c^{\prime}} q^{\prime}$ and there are positions $p_{2}^{\prime}$ and $p_{2}^{\prime \prime}$ such that $p_{1} \cdot p_{2} \sim_{c} p_{1} \cdot p_{2}^{\prime} \sim_{c} p_{1} \cdot p_{2}^{\prime \prime}$ and $p^{\prime}$ and $p_{2}^{\prime}$ on one hand and $q^{\prime}$ and $p_{2}^{\prime \prime}$ on the other hand are comparable w.r.t. the prefix ordering. $p_{2}^{\prime}$ must be distinct from $p_{2}^{\prime \prime}$ since $p^{\prime}$ and $q^{\prime}$ do not share any prefix.

By rules $\left(N_{4}\right),\left(N_{6}\right), c^{\prime}=c_{1} \sqcap \ldots \sqcap c_{n}$ and $c_{1} \vDash p_{2}^{\prime}=p_{2}^{\prime \prime}$. Now, we consider a number of cases, depending on the comparisons between $p^{\prime}, q^{\prime}, p_{2}^{\prime}, q_{2}^{\prime}$ :
Case 1: $p^{\prime}\left\langle_{\text {pref }} p_{2}^{\prime}\right.$.
In this case, by definition of $\Pi,\left.e\right|_{p_{1} \cdot p_{2}^{\prime}}$ must be basic: $c^{\prime}$ has to contain a suffix of $p_{2}^{\prime}$ and we use property 2 on $c^{\prime}$. It follows that $\left.e\right|_{p_{1} \cdot p_{2}}$ is basic.
Case 2: $q^{\prime}\left\langle_{\text {pref }} p_{2}^{\prime \prime}\right.$. This is similar to the first case.
Case 3: $p^{\prime} \geqslant \operatorname{pref} p_{2}^{\prime}$ and $q^{\prime} \geqslant \operatorname{pref} p_{2}^{\prime \prime}$.
Let $p^{\prime}=p_{2}^{\prime} \cdot q_{2}^{\prime}$. By definition of $\square, p^{\prime} \sim_{c^{\prime}} p_{2}^{\prime \prime} \cdot q_{2}^{\prime}$. Then $q^{\prime}$ must be equal to $p_{2}^{\prime \prime} \cdot q_{2}^{\prime}$ and $c \vDash p_{1} \cdot\left(p^{\prime}=q^{\prime}\right)$. Next, equalities $p^{\prime} \sim_{c^{\prime}} q^{\prime \prime}$, in which $p^{\prime}$ and $q^{\prime \prime}$ share a non-trivial prefix must be consequences (w.r.t. $\vDash_{e}$ ) of equality tests on subexpressions: this is true for normal forms w.r.t. ( $N_{4}$ ) and this is an invariant of ( $N_{6}$ ) since $c \sqcap c^{\prime} \vDash_{e} c \wedge c^{\prime}$.

Then, rule $N_{7}$ can be applied (contradiction).

## Appendix B. SET-constraints and automata with one memory

We consider a satisfiable SET-constraint $S$ and we assume that $A_{S}$ is constructed as described in Section 5.3.

We can first note that if $e=f^{c}\left(e_{1}, \ldots, e_{n}\right)$ occurs in $S$ and if $p$ is a non-root position of $e$, then $c \downarrow p$ has only one nontrivial equivalence class. This is ensured by conditions 1,4 and 2 of normal expressions (see Definition 27).

We prove by induction on the size of $t$ that, if $\alpha$ is the solution of $\operatorname{solved}(S)$, then for every $t$ accepted in $q_{X}, H(t) \in \alpha(X)$.

For the sake of simplicity, we will say shortly that " $t$ is accepted in $\langle q, m\rangle$ " instead of "there is a computation of the automaton on $t$ yielding the configuration $\langle q, m\rangle$.

## Lemma 84.

- if $t$ is accepted in $\left\langle q_{e, p}, \tau\right\rangle$, then $H(t) \in \alpha\left(\left.e\right|_{p}\right), H(t) \vDash c \downarrow p$ and $\tau=\left.H(t)\right|_{p \cdot p^{\prime}}$ for some $p^{\prime}$ tested by $c \downarrow p$ (since $H(t) \vDash c \downarrow p$ and since $c \downarrow p$ has only one nontrivial equivalence class, for every $p^{\prime}, p^{\prime \prime}$ tested by $c \downarrow p$, we have $\left.\left.H(t)\right|_{p \cdot p^{\prime}}=\left.H(t)\right|_{p \cdot p^{\prime \prime}}\right)$.
- if $t$ is accepted in $\left\langle q_{e}, \tau\right\rangle$, then $H(t) \in \alpha(e)$ and $\tau=a$.
- if t is accepted in $\left\langle q_{X}, \tau\right\rangle$, then $H(t) \in \alpha(X)$ and $\tau=H(t)$.
- if t is accepted in $\left\langle q_{\phi}, \tau\right\rangle$, then $\alpha \vDash \phi$.
- if $t$ is accepted in $\left\langle q_{a}, \tau\right\rangle$, then $t=a$ and $\tau=a$.

Proof (sketch). If $|t|=1$, then $t=b$ for some constant in $\mathcal{F}$ and the only transition leading to $b$ is $b \underset{b}{\top} q_{b}$.

Assume Lemma 84 is satisfied for every $t$ of size $\leqslant n$ and consider $t$ such that $|t|=n+1$. If $t$ is accepted in $\left\langle q_{e, p}, \tau\right\rangle$, then $t=g\left(t_{1}, \ldots, t_{k}\right)$ such that $t_{i}$ is accepted in $\left\langle q_{e, p \cdot i}, \tau_{i}\right\rangle$. By induction hypothesis, $H\left(t_{i}\right) \in \alpha\left(\left.e\right|_{p \cdot i}\right), H\left(t_{i}\right) \vDash c \downarrow p \cdot i$ and $\tau=\left.H\left(t_{i}\right)\right|_{p \cdot i \cdot p^{\prime}}$ for some $p^{\prime}$ tested by $c \downarrow p \cdot i . H(t)=g\left(H\left(t_{1}\right), \ldots, H\left(t_{n}\right)\right)$, thus $H(t) \in \alpha\left(\left.e\right|_{p}\right)$. We have also that
 checked by $c$, thus $\tau=\left.H(t)\right|_{p \cdot p^{\prime}}$ for some $p^{\prime}$ tested by $c \downarrow p$.
The other cases are proved similarly.
Conversely, if $t \in \alpha(X)$ where $\alpha$ is the minimal solution of $\operatorname{solved}(S)$, then there exists $t^{\prime}$ such that $H(t)=t^{\prime}$ and $t^{\prime}$ is accepted in state $q_{X}$.

## Lemma 85.

1. if $t \in \alpha(X)$, then there exists $t^{\prime}$ such that $H\left(t^{\prime}\right)=t$ and $t^{\prime}$ is accepted in $\left\langle q_{X}, t^{\prime}\right\rangle$.
2. if $t \in \alpha\left(\left.e\right|_{p}\right)$ and $t \vDash c \downarrow p$, then there exists $t^{\prime}$ such that $H\left(t^{\prime}\right)=t$ and $t^{\prime}$ is accepted in $\left\langle q_{e, p},\left.t\right|_{p \cdot p^{\prime}}\right\rangle$ for some $p^{\prime}$ tested by $c \downarrow p$.
3. if $t \in \alpha(e)$, then there exists $t^{\prime}$ such that $H\left(t^{\prime}\right)=t$ and $t^{\prime}$ is accepted in $\left\langle q_{e}, a\right\rangle$.

Proof (sketch). We prove that if $T_{\text {solved }(S)}^{n}(\emptyset)$ satisfies the properties of Lemma 85, then $T_{\text {solved }(S)}^{n+1}(\emptyset)$ also satisfies the properties of Lemma 85 . The result follows by minimality of $\alpha$.

Assume $T_{\text {solved }(S)}^{n}(\emptyset)$ satisfies the properties of lemma 85. First, we can verify that if $T_{\text {solved }(S)}^{n}(\emptyset) \vDash \phi$ then there exists $t^{\prime}$ such that $t^{\prime}$ is accepted in state $q_{\phi}$. Assume now that $\left.T_{\text {solved }(S)}^{n+( }\right)$ satisfies property 1 , then, by well-founded induction on $\geqslant$ (the reverse prefix order), we show that $T_{\text {solved }(S)}^{n+1}(\emptyset)$ satisfies properties 2 and 3 .
Thus, it is sufficient to prove that $T_{\text {solved }(S)}^{n+1}(\emptyset)$ satisfies property 1 : assume $t \in \llbracket X \rrbracket_{n}$, then there exists a clause $\phi \Rightarrow e \subseteq X$ such that $T_{\text {solved }(S)}^{n}(\emptyset) \vDash \phi$ and $t \in \llbracket e \rrbracket_{n}$. Applying the induction hypothesis and the rules of the automaton, we deduce that there exists $t^{\prime}$ such that $H\left(t^{\prime}\right)=t$ and $t^{\prime}$ is accepted in $\left\langle q_{X}, t^{\prime}\right\rangle$.

## Appendix C. Protocols and Horn Clauses

Let $P$ be a protocol. We assume $\mathcal{H}_{M s g}, \mathcal{H}_{I}$ and $\mathcal{H}_{P}$ constructed as described in Section 6.2.

We first construct $\mathcal{H}_{I_{0}}$ such that the maximal initial knowledge of the intruder $I_{0}$ is a minimal interpretation of the predicate $I_{0}$ which satisfies $\mathcal{H}_{I_{0}}$.

Lemma 58. Let $t_{1}, \ldots, t_{n}$ be message schemes with the free variables $x_{1}, \ldots, x_{k}$. Then, there exists a set of Horn clauses $\mathcal{H}_{I_{0}}$ such that $I_{0}(m)$ is derivable from $\mathcal{H}_{I_{0}} \cup \mathcal{H}_{\mathrm{msg}}$ if and only if $\operatorname{parts}(m) \cap\left\{t_{i} \sigma \mid 1 \leqslant i \leqslant n, \sigma\left(x_{i}\right) \in \mathcal{A}_{h}\right\}=\emptyset$.

Proof. Let $p$ be the maximal depth of the terms $t_{1}, \ldots, t_{n}$. Let $S=\left\{t_{i} \sigma \mid 1 \leqslant i \leqslant n, \sigma\left(x_{i}\right) \in\right.$ $\left.\mathcal{A}_{h}\right\}$. We introduce a new predicate $P_{\geqslant p}$ such that $P_{\geqslant p}$ accepts the terms of depth larger or equal to $p$. The clauses $\mathcal{H}_{I_{0}}$ for $P_{\geqslant p}$ and $I_{0}$ are described in Fig. 16. $|m|_{d}$ denotes the depth of the term $m$. Let $\mathcal{H}^{\prime}=\mathcal{H}_{M s g} \cup \mathcal{H}_{I_{0}}$.
By construction, $\mathcal{H}^{\prime} \vdash P_{\geqslant p}(m)$ if and only if the depth of $m$ is greater or equal to $p$.
Let us show by induction on the number of rules which have been applied that if $\mathcal{H}^{\prime} \vdash I_{0}(m)$ then parts $(m) \cap S=\emptyset$. Let $m$ be a term such that $\mathcal{H}^{\prime} \vdash I_{0}(m)$ and let us consider the last rule which has been applied:
Rule 1. $\operatorname{parts}(m) \cap S=\emptyset$ by definition of rule (1).
Rule 4. $m=f\left(m_{1}, \ldots, m_{n}\right)$ such that $f \in \mathcal{I} \mathcal{F}, \mathcal{H}^{\prime} \vdash I_{0}\left(m_{1}\right), \ldots, I_{0}\left(m_{n}\right)$ and there exists $i$ such that $\mathcal{H}^{\prime} \vdash P \geqslant{ }^{\prime}\left(m_{i}\right)$.

$$
\operatorname{parts}(m)=\{m\} \cup \bigcup_{1 \leqslant i \leqslant n} \operatorname{parts}\left(m_{i}\right)
$$

By induction hypothesis, parts $\left(m_{i}\right) \cap S=\emptyset$. In addition, there exists $i$ such that $\left|m_{i}\right|_{d} \geqslant p$, thus $\left.|m|_{d}\right\rangle p$ which implies that $m \notin S$. Thus parts $(m) \cap S=\emptyset$.
Rule 5. $m=f\left(m_{1}, \ldots, m_{n}\right)$ such that $f \in \mathcal{O} \mathcal{F}$ and there exists $i$ such that $\mathcal{H}^{\prime} \vdash P \geqslant p\left(m_{i}\right)$, thus $\left.|m|_{d}\right\rangle p$. Since parts $(m)=\{m\}$ and $\left.|m|_{d}\right\rangle p$, we have parts $(m) \cap S=\emptyset$.
Rule 6. this case is similar to the previous ones.
Conversely, an induction on the depth of $m$ proves that if parts $(m) \cap S=\emptyset$, then $\mathcal{H}^{\prime} \vdash I_{0}(m)$.

We prove here Lemma 57 by proving the following stronger lemma.

Lemma 86. Let $m$ be a message and a an agent, $\mathcal{C} \vdash^{\star} I(\bar{m})$ iff there exists a reachable $H$ such that $[m] \in H$ and $\mathcal{C} \vdash^{\star} I\left(s_{i}(\bar{a}, \bar{m})\right)$ iff there exists a reachable $H$ such that $S(a, i, m) \in H$.

To prove this, we need few lemmas:
Lemma 87. If there exists a reachable $H_{1}$ such that $m_{1} \in H_{1}$ where $m_{1}$ is a message and if there exists a reachable $H_{2}$ such that $e_{2} \in H_{2}$ where $e_{2}$ is either a message or a state, then there exists a reachable $H$ such that $m_{1}, e_{2} \in H$.


Fig. 16. Horn clauses for $P_{\geqslant p}$ and $I_{0}$.

A transition $t$ of a protocol is applicable in $H$ provided $\operatorname{Pre}(t) \subseteq H$. Thus, if $t$ is applicable in $H$, then $t$ is applicable in $H^{\prime}$, for all $H^{\prime} \supseteq H$. In the same way, if $X \in$ fake $(\operatorname{Cont}(H) \cup I)$ then $X \in$ fake $\left(\operatorname{Cont}\left(H^{\prime}\right) \cup I\right)$, for all $H^{\prime} \supseteq H$.
Therefore, let $H$ be the global state obtained from $H_{1}$ by applying all the transitions used to obtain $H_{2}$. $e_{2}$ is in $H$ and $m_{1}$ is still in $H$ since the transitions do not remove any message.

Lemma 88. Let $S$ a set of messages such that $\forall m \in S \quad \mathcal{C} \vdash^{\star} \bar{m}$. Then $\forall m \in$ fake $(S)$, $\mathcal{C} \vdash^{\star} \bar{m}$.

Pairing, unpairing, encryption and decryption are simulated by the clauses in Fig. 8.
For Lemma 57, we first prove by induction on $n$ that if $\mathcal{C} \vdash^{n} I(\bar{m})$ then there exists a reachable $H$ such that $[m] \in H$ and if $\mathcal{C} \vdash^{n} I\left(s_{i}(\bar{a}, \bar{m})\right)$ then there exists a reachable $H$ such that $S(a, i, m) \in H$.

For $n=0$, it is true,
Assume the hypothesis is verified for $n$, and assume $\mathcal{C} \vdash^{n+1} I(\bar{m})$. The last deduction rule is either one of those presented in Fig. 8, in this case, by inspection of the deduction rules, using the induction hypothesis and Lemma 87, we conclude that there exists a reachable $H$ such that $[m] \in H$. Or the last deduction rule is one of those presented in Fig. 9. Then,

$$
\mathcal{C} \vdash^{n} I\left(s_{i}\left(\bar{a}, \overline{m_{0}}\right)\right), M \operatorname{sg}\left(\overline{m_{0}}\right), I\left(\overline{m_{1}}\right), M \operatorname{sg}\left(\overline{m_{1}}\right), \ldots, I\left(\overline{m_{p}}\right), M \operatorname{sg}\left(\overline{m_{p}}\right)
$$

where $\overline{m_{i}}=\overline{M_{i}} \sigma^{\prime}, m_{0}=\overline{M_{0}} \sigma^{\prime}$ and $\sigma^{\prime}$ preserves the type : if $\sigma^{\prime}\left(x_{A}\right)=\bar{t}$ where $A$ is an agent variable, then $\mathcal{C} \vdash^{n} A(\bar{t})$, thus $t$ is an agent. By application of the induction hypothesis and applying Lemma 87 , there exists a reachable $H$ such that $m, m_{1}, \ldots, m_{p} \in H$. Thus, the transition

$$
\begin{aligned}
t=\{ & \left\{S\left(A \sigma^{\prime}, i, M \sigma^{\prime}\right), M_{1} \sigma^{\prime}, \ldots, M_{p} \sigma^{\prime}\right\} \longrightarrow
\end{aligned} \quad\left\{\begin{array}{l}
\longrightarrow \\
\\
\end{array}\right.
$$

is applicable in $H$. Let $H^{\prime}$ the global state obtained from $H$ by applying $t, m$ is in $H^{\prime}$.
Assume $\mathcal{C} \vdash^{n+1} I\left(s_{i}(\bar{a}, \bar{m})\right)$. The only choice for the last deduction rule is one those presented in Fig. 9. The same reasoning as above allows us to conclude that there exists a reachable $H$ such that $S(a, i, m) \in H$.
Conversely, we prove by induction on $n$ that if there exists a $n$-reachable $H$ such that $[m] \in H$ or $S(a, i, m) \in H$ then $\mathcal{C} \vdash^{\star} I(\bar{m})$ or $\mathcal{C} \vdash^{\star} I\left(s_{i}(\bar{a}, \bar{m})\right)$ where $n$-reachable stands for "reachable with $n$ global transitions".

For $n=0, H=H_{0}$ and $H_{0}$ does not contain any message or state.
Assume the hypothesis is verified for $n$, and assume there exists a $n+1$-reachable $H$ such that $[m] \in H$. Thus, there exists a $n$-reachable $H_{1}$ such that $H$ is an honest or fake successor of $H^{\prime}$. If $[m] \in H^{\prime}$, we conclude immediately. Assume $[m] \notin H^{\prime}$ :
Honest successor. Let $t$ the applicable transition such that $H=\left(H^{\prime} \backslash\left(\operatorname{Pre}(t) \cap H^{\prime}\right) \cup\right.$ $\operatorname{Post}(t))$. By application of the induction hypothesis and applying the clause described in Fig. 9, we conclude $\mathcal{C} \vdash^{\star} I(\bar{m})$.
Fake successor. If $H=H^{\prime} \cup\{m\}$ where $H^{\prime}$ is $n$-reachable and $m \in$ fake $(\operatorname{Cont}(H) \cup I)$.
Lemma 88 and the induction hypothesis allows us to conclude.

## References

[1] M. Abadi, A. Gordon, A calculus for cryptographic protocols: the spi calculus, Inform. Comput. 148 (1) (1999) 1-70.
[2] A. Aiken, Introduction to set constraint-based program analysis, Sci. Comput. Program. 35 (1999) 79-111.
[3] R. Amadio, D. Lugiez, On the reachability problem in cryptographic protocols, in Proc. CONCUR'00, Lecture Notes in Computer Science, Vol. 1877, 2000.
[4] B. Bogaert, S. Tison, Equality and disequality constraints on brother terms in tree automata, in: A. Finkel (Ed.), Porc. 9th Symp. on Theoretical Aspects of Computer Science, Cachan, France, 1992.
[5] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, A. Scedrov, A meta-notation for protocol analysis, in: P. Syverson (Ed.), 12th IEEE Computer Security Foundations Workshop, IEEE Computer Society Press, Silver Spring, MD, 1999.
[6] W. Charatonik, L. Pacholski, Negative set constraints with equality, in: Proc. IEEE Symp. on Logic in Computer Science, Paris, 1994, pp. 128-136.
[7] W. Charatonik, A. Podelski, Set constraints with intersection, in: Proc. IEEE Symp. on Logic in Computer Science, Warsaw, 1997.
[8] J. Clarke, J. Jacobs, A survey of authentication protocol literature: Version 1.0, Draft paper, 1997.
[9] H. Comon, V. Cortier, Tree automata with one memory, set constraints and cryptographic protocols, Res. Report LSV-01-13, Laboratoire Spécification and Vérification, ENS de Cachan, France, December 2001.
[10] H. Comon, V. Cortier, J. Mitchell, Tree automata with one memory, set constraints and ping-pong protocols, in: Proc. 28th Int. Coll. Automata, Languages, and Programming (ICALP’01), Crete, Greece, July 2001.
[11] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi, Tree automata techniques and applications, available on: http://www.grappa.univ-lille3.fr/tata, 1997
[12] H. Comon, V. Shmatikov, Is it possible to decide whether a cryptographic protocol is secure or not?, J. Telecomm. Inform. Tech. 4 (2002) 5-15.
[13] V. Cortier, J. Millen, Harald Ruess, Proving Secrecy is easy enough, 14th IEEE Computer Security Foundations Workshop, Cape Breton, 2001, p. 97-108.
[14] N. Dershowitz, J. Jouannaud, Handbook of Theoretical Computer Science, Vol. B, Chap. Rewrite Systems, Elsevier, 1990, pp. 243-320.
[15] D. Dolev, S. Even, R. Karp, On the security of ping pong protocols, Inform. Control 55 (1982) 57-68.
[16] D. Dolev, A. Yao, On the security of public key protocols, in: Proc. IEEE Symp. on Foundations of Computer Science, 1981, p. 350-357.
[17] C. Dufourd, A. Finkel, Ph. Schnoebelen, Reset nets between decidability and undecidability, in: Proc. 25th Int. Coll. Automata, Languages, and Programming (ICALP'98), Aalborg, Denmark, July 1998, Lecture Notes in Computer Science, Vol. 1443, Springer, Berlin, 1998, pp. 103-115.
[18] N. Durgin, P. Lincoln, J. Mitchell, A. Scedrov, Undecidability of bounded security protocols, in: Proc. Workshop on Formal Methods in Security Protocols, Trento, Italy, 1999.
[19] S. Even, O. Goldreich, On the security of multi-party ping-pong protocols, Tech. Report 285, Technion, Haifa, Israel, 1983. Extended abstract appeared in IEEE Symp. Foundations of Computer Science, 1983.
[20] N. Heinze, J. Jaffar, A decision procedure for a class of set constraints, in: Proc. IEEE Symp. on Logic in Computer Science, Philadelphia, 1990, pp. 42-51.
[21] N. Heintze, J. Tygar, A model for secure protocols and their compositions, IEEE Trans. Software Eng. 22 (1) (1996).
[22] G. Lowe, Breaking and fixing the Needham-Schroeder public-key protocol using FDR, in: Proc. Tools and Algorithms for the Construction and Analysis of Systems (TACAS), T. Margaria, B. Steffen (Eds.), Lecture Notes in Computer Science, Vol. 1055, Springer-Verlag, Berlin, 1996, pp. 147-166.
[23] J. Millen, H. Ruess, Protocol-Independent Secrecy, in: Proc. Workshop on Formal Methods in Computer Security, Chicago, 2000.
[24] J. Millen, V. Shmatikov, Constraint solving for bounded-process cryptographic protocol analysis, in: Proc. 8th ACM Conf. on Computer and Communications Security, 2001, pp. 166-175.
[25] J. Mongy, Transformation de noyaux reconnaissables d'arbres. Forêts RATEG. Ph.D. thesis, Laboratoire d'Informatique Fondamentale de Lille, Université des Sciences et Technologies de Lille, Villeneuve d'Ascq, France, 1981.
[26] L. Paulson, The inductive approach to verifying cryptographic protocols, J. Comput. Security 6 (1998) 85-128.
[27] M. Rusinowitch, M. Turuani, Protocol insecurity with finite number of sessions is NP complete, in: Proc. 14th IEEE Computer Security Foundations Workshop, 2001, pp. 174-190.
[28] H. Seidl, Haskell overloading is DEXPTIME-complete, Inf. Process. Lett. 52 (1994) 57-60.


[^0]:    ${ }^{2}$ This is an extended version of a paper whose abstract appeared in Proc. ICALP 2001.

    * Corresponding author.

    E-mail addresses: comon@1sv.ens-cachan.fr (H. Comon), cortier@1sv.ens-cachan.fr (V. Cortier).
    ${ }^{1}$ Partially supported by a NATO grant. Partially supported by DoD MURI "Semantic Consistency in Information Exchange", ONR GrantN00014-97-1-0505, and NSF CCR-9629754. Partially supported by INRIA project SECSI and RNTL project EVA.
    ${ }^{2}$ Partially supported RNTL project EVA and ACI VERNAM.

