Automated verification of equivalence properties of cryptographic protocols

Rohit Chadha, Stefan Ciobaca, Steve Kremer

Automated verification of equivalence properties of cryptographic protocols*

Rohit Chadha[1], Ştefan Ciobâcă[1], and Steve Kremer[1,2]

[1] LSV, ENS Cachan & CNRS & INRIA
[2] INRIA Nancy - Grand-Est

Abstract. Indistinguishability properties are essential in formal verification of cryptographic protocols. They are needed to model anonymity properties, strong versions of confidentiality and resistance against offline guessing attacks, which can be conveniently modeled using process equivalences. We present a novel procedure to verify equivalence properties for bounded number of sessions of cryptographic protocols. As in the applied pi-calculus, our protocol specification language is parametrized by a first-order sorted term signature and an equational theory which allows formalization of algebraic properties of cryptographic primitives. Our procedure is able to verify trace equivalence for determinate cryptographic protocols. On determinate protocols, trace equivalence coincides with observational equivalence which can therefore be automatically verified for such processes. When protocols are not determinate our procedure can be used for both under- and over-approximations of trace equivalence, which proved successful on examples. The procedure can handle a large set of cryptographic primitives, namely those which can be modeled by an optimally reducing convergent rewrite system. The procedure is based on a fully abstract modelling of the traces of a bounded number of sessions of the protocols in first-order Horn clauses on which a dedicated resolution procedure is used to decide equivalence properties. Although, we were unable to prove termination of the resolution procedure, the procedure has been implemented in a prototype tool A-KiSs (Active Knowledge in Security Protocols) and has been effectively tested on examples some of which were outside the scope of existing tools, including checking anonymity of an electronic voting protocol.

1 Introduction

Cryptographic protocols are distributed programs which rely on the use of cryptography to secure electronic transactions such as those that arise in electronic commerce and wireless communication. They are also being applied in new domains such as in Internet voting—legally binding political elections in Estonia, Norway and Switzerland offer the possibility for Internet voting in 2011. This has led to increasing demands on the complexity of desired security properties, leading to more complex cryptographic protocols. Given the socio-economic-political consequences and the history of incorrect design of cryptographic protocols, the need for formal proofs of correctness of protocols is of great importance and has been widely recognized. Formal reasoning about cryptographic protocols is challenging as one has to reason against all potentially malicious behavior—all communication between protocol participants is assumed to be under the control of an adversary.

In order to make the task of formal analysis amenable to automation, usually the assumption of back-box cryptography and unbounded computational power on the part of the adversary is made. This adversarial model is often called the Dolev-Yao model as it is derived from the positions that Dolev and Yao took in their seminal paper [36]. It has proved extremely successful, and there are several automated tools [13, 6, 29, 38] that can automatically check trace-properties such as (weak forms of) confidentiality and authentication. While these trace-based properties are certainly important, many crucial security properties can only be expressed in terms of *indistinguishability* (or equivalence). They include strong flavors of confidentiality [14]; resistance to guessing attacks in password based protocols [10]; and anonymity properties in private authentication [3], electronic voting [33, 9], vehicular networks [30, 31] and RFID protools [5, 18]. More generally,

* This work was partially supported by ANR projects ARA SESUR AVOTÉ and JCJC VIP no 11 JS02 006 01.

indistinguishability allows to model security by the means of ideal systems, which are correct by construction [4, 32]. Indistinguishability properties of cryptographic protocols are naturally modeled by the means of *observational* and *testing equivalences* in cryptographic extensions of process calculi, e.g., the spi [4] and the applied-pi calculus [2]. While we have good tools for automated verification of trace properties, the situation is different for indistinguishability properties. This paper is an attempt to address this concern.

State-of-the-art. Many results have been obtained in the restricted case of a pure eavesdropper, i.e., a passive adversary: for *static equivalence* many decidability results have been shown [1, 27, 7] and exact [11, 25] and approximate [15] tools exist for a variety of cryptographic primitives. In the case we consider indistinguishability in the presence of an active adversary, who can interact in an arbitrary way with honest participants less results are known. Hüttel [41] showed undecidability of observational equivalence in the spi calculus, even for the finite control fragment, as well as decidability for the finite, i.e., replication-free, fragment of the spi calculus. The decidability result however only holds for a fixed set of cryptographic primitives and does not yield a practical algorithm. Current results [15] allow to approximate observational equivalence for an unbounded number of sessions. However, this approximation does not suffice to conclude for many applications, *e.g.*, [33, 5]. Our approach overcomes these limitations for some applications in [33]. We still cannot conclude for the e-passport example in [5], albeit for a different reason: our procedure does not currently handle else branches in protocols.

Symbolic bisimulations have also been devised for the spi [17, 16, 47] and applied pi calculus [34, 43] to avoid unbounded branching due to adversary inputs. However, only [34, 47] and [17] yield a decision procedure, but again only approximating observational equivalence. The results of [34] have been further refined to show a decision procedure on a restricted class of *simple* processes [28]. In particular they rely on a procedure deciding the equivalence of constraint systems, introduced by Baudet [10], for the special case of verifying the existence of guessing attacks. Baudet's procedure allows arbitrary cryptographic primitives that can be modeled as a subterm convergent rewrite systems [1]. An alternate procedure achieving the same goal was proposed by Chevalier and Rusinowitch [21]. However, both procedures are highly non-deterministic and do not yield a reasonable algorithm which could be implemented. Therefore, Cheval *et al.* [19] have designed a new procedure and a prototype tool to decide the equivalence of constraint systems, but only for a fixed set of primitives. Tools have also been implemented for checking testing equivalence [37], open bisimulation [47] and trace equivalence [20] for a bounded number of sessions but only a limited set of primitives. One may note that [20] is the only decision procedure to consider negative tests, i.e., else branches, which are crucial in several case studies [5, 3].

Our contribution. In this paper we introduce a new procedure for verifying equivalence properties for processes specified in a cryptographic process calculus (without replication). The messages in the calculus are modeled as terms equipped with an equational theory, similar to the applied pi calculus. Our main contributions are as follows.

- Our procedure checks for two equivalences which over- and under-approximate the standard notion of trace equivalence $\approx_t$ for cryptographic protocols: the under-approximation can be used to prove protocols correct while the over-approximation can be used to rule out incorrect protocols.
- Cortier and Delaune have shown in [28] that observational equivalence coincides with $\approx_t$ for the class of *determinate* processes. They also give a decision procedure for a strict sub-class of determinate processes, namely, *simple* processes. We show that the coarser relation coincides with $\approx_t$, and thus our procedure can be used to verify observational equivalence for the whole class of determinate processes.
- A novelty of our procedure is that it is based on a *fully abstract* modeling of symbolic traces for a *bounded* number of sessions in *first-order Horn clauses*. This is in contrast to the constraint-solving techniques employed by Tiu *et al.* [47], Cheval *et al.* [19, 20], Baudet [10] and Chevalier et al. [21] for verifying under-approximations of observational equivalence. Techniques based on Horn clauses have been extensively used, e.g., by Blanchet [13], Weidenbach [48] and Goubault [40], in the case of an unbounded number of sessions. Of these tools, only Blanchet [13] can verify an equivalence property, which happens to be an under-approximation of observational equivalence. Horn clause modeling of an unbounded number of

sessions of security protocols may allow false attacks. On the other hand, we have proven our modeling of a bounded number of sessions to be precise.
- Our modelling is fully abstract for arbitrary cryptographic primitives that can be modeled as a convergent rewrite system which has the *finite variant property*. Not only this strictly includes the class of primitives that can be modeled as subterm convergent rewrite systems, this allows us to handle a larger class of cryptographic primitives than [47, 19, 20, 10, 21, 13]. For example, this allows us to handle trapdoor commitment as used by Okamoto for electonic voting in [46]. Although we were unable to prove termination of our procedure, we conjecture it to terminate for the class of cryptographic primitives that can be modeled as subterm convergent rewrite systems. Our conjecture is supported by experimental evidence.
- Our procedure is implemented in the AKISS (Active Knowledge in Security protocols) prototype tool and used it among others to successfully prove anonymity in an electronic voting protocol [39]. For this electronic voting protocol, this is the first automated proof.

2 Preliminaries

2.1 Terms

Let $\mathcal{F}$ be a signature, i.e., a finite set of function symbols and let ar be a function which assigns to each function symbol a natural number. Given a function symbol $f \in \mathcal{F}$, we say $ar(f) \in \mathbb{N}$ is the arity of f. A function symbol of arity 0 is called a *constant*. Given a set of *atoms* $\mathcal{A}$ and a signature $\mathcal{F}$, we denote by $\mathcal{T}_{\mathcal{F},\mathcal{A}}$ the set of terms built inductively from $\mathcal{A}$ by applying functions symbols in $\mathcal{F}$. Given sets of atoms $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$, we denote the set $\mathcal{T}_{\mathcal{F},\cup_{1\leq i\leq n}\mathcal{A}_i}$ by $\mathcal{T}_{\mathcal{F},\mathcal{A}_1,\mathcal{A}_2,\ldots\mathcal{A}_n}$. We assume that we have the following countably infinite pairwise disjoint sets: a set $\mathcal{N}$ of *private names*, $\mathcal{M}$ of *public names*, a set $\mathcal{C}$ of public *channel names*, a set $\mathcal{W}$ of *parameters*, and a set $\mathcal{X}$ of message variables. Intuitively, elements of the set $\mathcal{N}$ represent nonces generated by honest principals of a protocol, elements of $\mathcal{M}$ represent nonces available both to the adversary and to the honest participants and elements of $\mathcal{C}$ represent names of public channel (e.g. the name of a public network). Elements of $\mathcal{W}$ are pointers used by the adversary to refer to messages output by the honest participants in a protocol. We fix an enumeration $w_1, w_2, \ldots$ of the elements of $\mathcal{W}$. We let x, y, z range over $\mathcal{X}$. We also define the following set of terms:

- Terms denotes the set of all terms $\mathcal{T}_{\mathcal{F},\mathcal{N},\mathcal{M},\mathcal{W},\mathcal{X}}$
- $\mathsf{Messages}$ denotes the set of *messages* $\mathcal{T}_{\mathcal{F},\mathcal{N},\mathcal{M}}$
- $\mathsf{SMessages}$ denotes the set of *symbolic messages* $\mathcal{T}_{\mathcal{F},\mathcal{N},\mathcal{M},\mathcal{X}}$.

If t is a term, we denote by $vars(t)$ the set of variables appearing in t, by $names(t)$ the set of names (public or private) appearing in t and $\mathsf{st}(t)$ the set of all subterms of t. The functions $vars$, $names$ and st are extended as expected to sequences and sets of terms. A *position* is a string of positive natural numbers and ϵ denotes the empty string. The set $pos(t)$ of positions of a term t is defined as usual. If $p \in pos(t)$ then $t|_p$ is the subterm of t at position p.

Example 1. Consider the signature $\mathcal{F} = \{\mathsf{enc}, \mathsf{dec}, \mathsf{pair}, \mathsf{fst}, \mathsf{snd}\}$ where $ar(\mathsf{enc}) = 3$, $ar(\mathsf{dec}) = ar(\mathsf{pair}) = 2$ and $ar(\mathsf{fst}) = ar(\mathsf{snd}) = 1$. The term $t = \mathsf{pair}(\mathsf{enc}(a, k_1, r_1), \mathsf{enc}(b, k_2, r_2))$ models the pair of the encryptions of public names a and b with keys k_1, resp. k_2 and randomness r_1, resp. r_2. The set of positions $pos(t) = \{\epsilon, 1, 11, 12, 13, 2, 21, 22, 23\}$ and $t_\epsilon = t$, $t_1 = \mathsf{enc}(a, k_1, r_1)$ and $t_{23} = r_2$.

Substitutions. A substitution is a partial function $\sigma : \mathcal{W} \cup \mathcal{X} \to \mathsf{Terms}$. We only consider substitutions which map elements of $\mathcal{W}$ to elements in $\mathsf{Messages}$, elements of $\mathcal{X}$ to elements of $\mathsf{SMessages}$. The domain of σ shall be denoted by $dom(\sigma)$. For our purposes, we only consider substitutions with finite domains. We let $range(\sigma) = \{\, \sigma(u) \in \mathcal{T} \mid u \in dom(\sigma) \,\}$. If $dom(\sigma) = \{\, u_1, u_2, \ldots, u_n \,\}$ and $t_i = \sigma(u_i)$ for each $1 \leq i \leq n$ then we shall write σ as $\{\, u_1 \mapsto t_1, \ldots, u_n \mapsto t_n \,\}$. σ is said to be *ground* if $range(\sigma) \subseteq \mathsf{Messages}$. The notation $names(\sigma)$ will denote the set $names(range(\sigma))$. As usual, a substitution extends homomorphically

to a function $\mathsf{apply}_\sigma : \mathsf{Terms} \to \mathsf{Terms}$ obtained by "applying" σ. Given $t \in \mathsf{Terms}$, we denote $\mathsf{apply}_\sigma(t)$ by $t\sigma$. If σ is a substitution and $X \subseteq dom(\sigma)$, we denoted by $\sigma[X]$ the substitution whose domain is restricted to X.

2.2 Rewriting and unification

Two terms s and t are (syntactically) *unifiable* if there exists a substitution σ such that $s\sigma = t\sigma$. We denote by mgu a function which associates to any two unifiable terms s and t a most general unifier σ of s and t such that $\sigma = \sigma[vars(s,t)]$. It is well known [8] that for any two unifiable terms s and t, there is a most general unifier, unique up to variable renaming.

A rewrite system R is a set of rewrite rules of the form $\ell \to r$ where $\ell, r \in \mathsf{Terms}$, $names(l,r) = \emptyset$ and $vars(r) \subseteq (\ell)$. A term t can be rewritten in one step to u, denoted $t \to_\mathsf{R} u$, if there exist a position $p \in pos(t)$, a rule $\ell \to r$ in R and a substitution σ such that $t|_p = \ell\sigma$ and u is as t where the subterm $t|_p$ is replaced by $r\sigma$. $\to_\mathsf{R}^*$ denotes the transitive and reflexive closure of $\to_\mathsf{R}$. A rewrite system is said to be confluent if for any t, t_1, t_2 such that $t \to_\mathsf{R}^* t_1$ and $t \to_\mathsf{R}^* t_2$ there exists u such that $t_1 \to_\mathsf{R}^* u$ and $t_2 \to_\mathsf{R}^* u$. A rewrite system is said to be terminating if it does not admit any infinite sequence $t_0 \to_\mathsf{R} t_1 \to_\mathsf{R} t_2 \to_\mathsf{R} \ldots$. It is convergent if it is both confluent and terminating. We denote by $t{\downarrow_\mathsf{R}}$ the normal form of a term t. When R is clear from the context or unimportant we simply write $t{\downarrow}$. Two terms s and t are said to be equal modulo R, written $s =_\mathsf{R} t$, if $s{\downarrow_\mathsf{R}} = t{\downarrow_\mathsf{R}}$. Given a substitution σ we denote by $\sigma{\downarrow}$ a substitution such that $dom(\sigma{\downarrow}) = dom(\sigma)$ and for all $u \in dom(\sigma)$ $\sigma{\downarrow}(u) = \sigma(u){\downarrow}$.

Example 2. Continuing Example 1, consider the rewrite system $\mathsf{R} = \{\mathsf{dec}(\mathsf{enc}(x,y,z),y) \to x, \mathsf{fst}(\mathsf{pair}(x,y)) \to x, \mathsf{snd}(\mathsf{pair}(x,y)) \to y\}$. The first rewrite rule models that a message can be decrypted, provided decryption uses the same key (represented by variable y) as encryption. The two last rules model projection of the first and second component of a pair. Then we have that $t = \mathsf{fst}(\mathsf{pair}(\mathsf{dec}(\mathsf{enc}(a,k,r),k),b)) \to_\mathsf{R} \mathsf{fst}(\mathsf{pair}(a,b)) \to_\mathsf{R} a = t{\downarrow_\mathsf{R}}$.

Given a convergent rewrite system, we now define the notion of complete set of variants, which was introduced by Common-Lundh and Delaune [26].

Definition 1. *A set of substitutions* variants$(t_1, \ldots, t_k)$ *is called a* complete set of variants *of terms* $t_1, \ldots, t_k$ *if for any substitution* ω *there exist* $\sigma \in$ variants$(t_1, \ldots, t_k)$ *and a substitution* τ *such that for all* $1 \leq j \leq k$ *we have that* $\omega[vars(t_j)]{\downarrow} = (\sigma{\downarrow}\tau)[vars(t_j)]$ *and* $(t_j\omega){\downarrow} = (t_j\sigma){\downarrow}\tau$.

Intuitively the set of variants of t represents a pre-computation such that any instance of t in normal form is *syntactically* equal to an instance of $t\sigma_i{\downarrow}$ for some i, without the need to apply further rewrite steps. A rewrite system has the *finite variant property* if for any sequence of terms a finite, complete set of variants exists. In [23], Ciobâcă presents an algorithm for computing such complete sets of variants which is correct whenever the rewrite system is *optimally reducing* [45]. Optimally reducing rewrite systems include subterm convergent systems [1] (and hence the classical Dolev Yao theories for encryption, signatures and hash functions), as well as a theory for modeling blind signatures [42]. Moreover, complete sets of variants can be used to perform unification modulo R [23].

Definition 2. *A set of substitutions* $\mathrm{mgu}_\mathsf{R}(\{s_i \stackrel{?}{=} t_i\}_{i \in I})$ *is called a complete set of unifiers modulo* R *of the system of equations* $\{s_i \stackrel{?}{=} t_i\}_{i \in I}$ *if:*

1. $dom(\sigma) \subseteq vars(X)$ *(for all* $\sigma \in \mathrm{mgu}_\mathsf{R}(\{s_i \stackrel{?}{=} t_i\}_{i \in I})$*),*
2. $s_i\sigma =_\mathsf{R} t_i\sigma$ *(for all* $i \in I$ *and for all* $\sigma \in \mathrm{mgu}_\mathsf{R}(\{s_i \stackrel{?}{=} t_i\}_{i \in I})$*) and*
3. *any* θ *such that* $s_i\theta =_\mathsf{R} t_i\theta$ *for all* $i \in I$ *is such that* $\exists \sigma \in \mathrm{mgu}_\mathsf{R}(\{s_i \stackrel{?}{=} t_i\}_{i \in I}) \exists \tau : \theta[X] =_\mathsf{R} (\sigma\tau)[X]$.

where $X = vars(\{s_i, t_i\}_{i \in I})$.

For singleton systems, we also write $\mathrm{mgu}_\mathsf{R}(s,t)$ instead of $\mathrm{mgu}_\mathsf{R}(\{s,t\})$.

From now on we assume that the rewrite system is convergent and has the finite variant property.

2.3 Frames, deducibility and static equivalence

Recall that we have fixed an enumeration $w_1, w_2, \ldots$ of the elements of the set $\mathcal{W}$. We will use the notion of a *frame* to represent messages which have been recorded by an attacker.

Definition 3. *A* frame φ *is a substitution* $\{w_1 \mapsto t_1, \ldots, w_n \mapsto t_n\}$ *where* $t_i \in \mathsf{Messages}$ *(*$1 \leq i \leq n$*).*

Note that in our definition, every frame with $|dom(\varphi)| = n$ has $dom(\varphi) = \{w_1, \ldots, w_n\}$. We denote the set of all frames as Frames. The adversary can use the messages learnt from the run of a protocol to construct new messages. This is modeled as the deducibility relation.

Definition 4. *Any term in* $\mathcal{T}_{\mathcal{F},\mathcal{M},\mathcal{W}}$ *is said to be a recipe. We say that* a message t is deducible from φ with a recipe r *(written as* $\varphi \vdash^r t$*) if* $t \in \mathsf{Messages}$ *and* $r\varphi =_{\mathsf{R}} t$*. We write* $\mathsf{Recipes}$ *for the set* $\mathcal{T}_{\mathcal{F},\mathcal{M},\mathcal{W}}$*.*

A frame $\varphi' = \{w_1 \mapsto t'_1, \ldots, w_m \mapsto t'_m\}$ *extends* a frame $\varphi = \{w_1 \mapsto t_1, \ldots, w_n \mapsto t_n\}$ if $m \geq n$ and if $t'_i = t_i$ for all $1 \leq i \leq n$. It is easy to see that if φ' extends φ and if $\varphi \vdash^r t$ then $\varphi' \vdash^r t$.

Example 3. Consider the signature $\mathcal{F}$ and the rewrite system R in Example 2. Let $\varphi\{w_1 \mapsto \mathsf{enc}(s,k,r), w_2 \mapsto k\}$ where $s, k, r \in \mathcal{N}$ are private names. Then we have that $\varphi \vdash^{\mathsf{dec}(w_1,w_2)} s$. Note that $\mathsf{dec}(w_1, k) \notin \mathsf{Recipes}$ as $k \in \mathcal{N}$. If we had that $s \in \mathcal{M}$ we would also have that $\varphi \vdash^s s$ reflecting that public names are always deducible.

We now define *static equivalence* of frames to capture indistinguishability of sequences of messages.

Definition 5. *Let* $r_1, r_2 \in \mathsf{Recipes}$*. A* test $r_1 \stackrel{?}{=} r_2$ holds *in a frame* φ *(written* $(r_1 = r_2)\varphi$*) if* $\varphi \vdash^{r_1} t$ *and* $\varphi \vdash^{r_2} t$ *for some* t*,* i.e.*,* r_1 *and* r_2 *are recipes for the same term in* φ*.*

A frame φ_1 *is* statically included *in* φ_2 *(written* $\varphi_1 \sqsubseteq_s \varphi_2$*) iff for all* $r_1, r_2 \in \mathsf{Recipes}$ *we have that* $(r_1 = r_2)\varphi_1$ *implies* $(r_1 = r_2)\varphi_2$*. Two frames* φ_1 *and* φ_2 *are* statically equivalent *(written* $\varphi_1 \approx_s \varphi_2$*) iff* $\varphi_1 \sqsubseteq_s \varphi_2$ *and* $\varphi_2 \sqsubseteq_s \varphi_1$*.*

Example 4. Let $a, b \in \mathcal{M}$ and $r, k, k' \in \mathcal{N}$. We have that $\{w_1 \mapsto \mathsf{enc}(a,k,r), w_2 \mapsto k\} \not\approx_s \{w_1 \mapsto \mathsf{enc}(b,k,r), w_2 \mapsto k\}$ because the test $(dec(w_1, w_2) = a)$ distinguishes both frames. However, $\{w_1 \mapsto \mathsf{enc}(a,k,r), w_2 \mapsto k'\} \approx_s \{w_1 \mapsto \mathsf{enc}(b,k,r), w_2 \mapsto k'\}$. Moreover, we have that $\{w_1 \mapsto a, w_2 \mapsto b\} \sqsubseteq_s \{w_1 \mapsto a, w_2 \mapsto a\}$ while $\{w_1 \mapsto a, w_2 \mapsto a\} \not\sqsubseteq_s \{w_1 \mapsto a, w_2 \mapsto b\}$.

3 A cryptographic process calculus

We shall assume that cryptographic protocols are modeled using a simple process calculus which has similarities with the applied pi-calculus [2]. Applied pi-calculus has proved to be useful for specifying and verifying cryptographic protocols; and there are tools that automate verification of protocols in this model [13]. We shall further restrict our attention to the finite, *i.e.*, replication-free fragment of applied pi-calculus. This restriction is important because observational equivalence becomes undecidable with replication [41]. However, even with the restriction, one can model a bounded number of protocol instances.

We define our process calculus in this section. We begin by defining its syntax.

Syntax. Recall that we have fixed a first-order signature $\mathcal{F}$, a set $\mathcal{N}$ of *private names*, $\mathcal{M}$ of *public names*, a set $\mathcal{C}$ of public channel names, a set $\mathcal{W}$ of *parameters*, and a set $\mathcal{X}$ of message variables (see Section 2). The terms of the set $\mathcal{T}_{\mathcal{F},\mathcal{N},\mathcal{M},\mathcal{W},\mathcal{X}}$ are also identified modulo a fixed subterm convergent rewrite system R (see Section 2).

We model a bounded number of instances of a cryptographic protocol as a *finite* set of traces. Traces are defined using sequences of *actions* generated by the following grammar (note that here **in** and **out** are fresh symbols not occurring in $\mathcal{F}$):

$$
\begin{array}{rll}
a ::= & \mathbf{in}(c, x) & \text{receive action} \\
& \mathbf{out}(c, t) & \text{send action} \\
& [s \stackrel{?}{=} t] & \text{test action}
\end{array}
$$

where $x \in \mathcal{X}, s, t \in \mathsf{SMessages}, c \in \mathcal{C}$. A *trace* T is a sequence of actions $T = a_1.a_2.\ldots.a_n$. As usual, a receive action $\mathbf{in}(c, x)$ acts as a binding construct for the variable x. We assume the usual definitions of free and bound variables for traces. We also assume that each variable is bound at most once. A trace is *ground* if it does not contain any free variables. The set of ground traces shall be represented as $\mathsf{GndTraces}$. We also assume the usual definition of a name occurring in a trace.

A *process* P is defined to be a set of traces $P = \{T_1, \ldots, T_n\}$. We say that a process is ground if all of its traces are ground. We identify traces with singleton processes.

Remark 1. We do not have an ν operator: the binding happens implicitly by the use of private names in $\mathcal{N}$. We have also not explicitly included the parallel operator $|$ and the choice operator $+$. One could include these and generate the corresponding set of traces. Thus, there is no loss in expressivity. However, an explicit enumeration of the traces can result in an exponential number of traces.

Semantics. The semantics of a process is defined using the semantics of its traces. The semantics of a trace is given in terms of a labeled transition system T. We assume that all interactions between protocol participants are mediated by the adversary. The labeled transition system records the interaction of the protocol participants with the adversary. The set of labels of T is defined using the set $\mathsf{Recipes}$. Recall that the set $\mathsf{Recipes}$ is the set $\mathcal{T}_{\mathcal{F},\mathcal{M},\mathcal{W}}$ (see Section 2). The set of labels, Labels, is

$$
\mathsf{Labels} = \{\, \mathbf{in}(c, r), \mathbf{out}(c), \mathbf{test} \mid r \in \mathsf{Recipes}, c \in \mathcal{C} \,\}.
$$

The labeled transition system T is a subset of $(\mathsf{GndTraces} \times \mathsf{Frames}) \times \mathsf{Labels} \times (\mathsf{GndTraces} \times \mathsf{Frames})$ and we shall write $(T, \varphi) \xrightarrow{\ell} (T', \varphi')$ whenever $((T, \varphi), \ell, (T', \varphi')) \in \mathsf{T}$. The frame in the transition system is used to record the messages that the protocol participants have sent in the past. The relation $\xrightarrow{\ell}$ is defined as follows:

$$
\textsc{Receive}\ \frac{\varphi \vdash^r t}{(\mathbf{in}(c, x).T, \varphi) \xrightarrow{\mathbf{in}(c,r)} (T\{x \mapsto t\}, \varphi)}
$$

$$
\textsc{Send}\ \frac{}{(\mathbf{out}(c, t).T, \varphi) \xrightarrow{\mathbf{out}(c)} (T, \varphi \cup \{w_{|dom(\varphi)|+1} \mapsto t\})}
$$

$$
\textsc{Test}\ \frac{s =_{\mathsf{R}} t}{([s \stackrel{?}{=} t].T, \varphi) \xrightarrow{\mathbf{test}} (T, \varphi)}
$$

The label $\mathbf{in}(c, r)$ indicates a message sent by the adversary over the channel c and r is the recipe that adversary uses to create this message. The label $\mathbf{out}(c)$ indicates a message sent over the public channel c and transition rule Send records the message sent in the frame. Finally, the rule Test is an internal action.

As usual, we shall write $(T_0, \varphi_0) \xrightarrow{\ell_1,\ldots,\ell_n} (T_n, \varphi_n)$ when $(T_0, \varphi_0) \xrightarrow{\ell_1} (T_1, \varphi_1) \ldots \xrightarrow{\ell_n} (T_n, \varphi_n)$ and we say that $\ell_1 \ldots \ell_n$ is a *run* of (T_0, φ_0). We shall write $(T, \varphi) \stackrel{\ell}{\Rightarrow} (T', \varphi')$ when either $(T, \varphi) \xrightarrow{\mathbf{test}^*,\ell,\mathbf{test}^*} (T', \varphi')$ and $\ell \neq \mathbf{test}$ or $(T, \varphi) \xrightarrow{\mathbf{test}^*} (T', \varphi')$ and $\ell = \mathbf{test}$, where $\mathbf{test}^*$ denotes an arbitrary number of $\mathbf{test}$ actions. We write $(T, \varphi) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T_n, \varphi_n)$ when $(T, \varphi) \stackrel{\ell_1}{\Rightarrow} (T_1, \varphi_1) \stackrel{\ell_2}{\Rightarrow} \ldots \stackrel{\ell_n}{\Rightarrow} (T_n, \varphi_n)$. If $P = \{T_1, \ldots, T_m\}$ is a process, we write $(P, \varphi) \xrightarrow{\ell_1,\ldots,\ell_n} (T', \varphi')$ (resp. $\stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T', \varphi')$) if there exists a trace $T \in P$ such that $(T, \varphi) \xrightarrow{\ell_1,\ldots,\ell_n} (T', \varphi')$ (resp. $(T, \varphi) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T', \varphi')$).

Process equivalences. In this section we will define different flavors of trace equivalence which will be useful in this paper. We first recall the standard definition of trace equivalence in cryptographic process algebras.

Definition 6. ***(Trace equivalence)*** *A ground process P is said to be* trace-included *in a ground process Q (written $P \sqsubseteq_t Q$) if whenever $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T, \varphi)$ then there exist T', φ' such that $(Q, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T', \varphi')$ and $\varphi \approx_s \varphi'$. Two processes P and Q are* trace-equivalent *(written $P \approx_t Q$) if $P \sqsubseteq_t Q$ and $Q \sqsubseteq_t P$.*

We will also define two other notions of trace equivalence, one coarser and one more fine-grained. The coarser trace equivalence, which we denote by $\approx_{ct}$ is the trace equivalence that can actually be verified by our procedure.

Definition 7. *Given ground processes P and Q, we say that $P \sqsubseteq_{ct} Q$ if whenever $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T, \varphi)$ then there exist T', φ' such that $(Q, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T', \varphi')$ and $\varphi \sqsubseteq_s \varphi'$. We say that $P \approx_{ct} Q$ if $P \sqsubseteq_{ct} Q$ and $Q \sqsubseteq_{ct} P$.*

The following example illustrates the difference between $\approx_t$ and $\approx_{ct}$.

Example 5. Let P and Q be the ground processes defined as follows: $P = \{\, \mathbf{out}(c, a).\mathbf{out}(c, a) \,\}$ and $Q = \{\, \mathbf{out}(c, a).\mathbf{out}(c, a), \mathbf{out}(c, a).\mathbf{out}(c, b) \,\}$. Clearly $P \sqsubseteq_{ct} Q$. Observe also that $Q \sqsubseteq_{ct} P$. This is because $\{w_1 \mapsto a, w_2 \mapsto b\} \sqsubseteq_s \{w_1 \mapsto a, w_2 \mapsto a\}$. Thus, $P \approx_{ct} Q$. But $P \not\approx_t Q$.

We will however show that these two notions coincide for a class of *determinate processes*. In the context of the applied pi calculus determinate processes were previously studied by Cortier and Delaune in [28].

Definition 8. ***(Determinate process)*** *We say that a ground process P is determinate if whenever $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T, \varphi)$ and $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T', \varphi')$ then $\varphi \approx_s \varphi'$.*

Intuitively, determinate processes are processes in which the adversary's static knowledge at any instance is completely determined by its past interaction with the protocol participants. The following is immediate from the definition.

Proposition 1. *A ground trace, i.e, a ground process consisting of single trace, is determinate.*

As already mentioned above, it was demonstrated in [28] that trace equivalence coincides with observational equivalence for determinate processes. We show that $\approx_t$ and $\approx_{ct}$ also coincide for this class of processes.

Theorem 1. *If P and Q are ground processes then $P \approx_t Q$ implies $P \approx_{ct} Q$. Furthermore if P and Q are determinate, then $P \approx_{ct} Q$ implies $P \approx_t Q$.*

Proof. Let P and Q be determinate processes.

($\Rightarrow$) Follows immediately from definition of $\approx_t$ and $\approx_{ct}$.

($\Leftarrow$) We need to show that $P \approx_{ct} Q$ implies $P \approx_t Q$. We proceed by contradiction. Suppose that $P \approx_{ct} Q$ and $P \not\approx_t Q$. We suppose $P \not\sqsubseteq_t Q$ (the case of $Q \not\sqsubseteq_t P$ being symmetric). As $P \not\sqsubseteq_t Q$ we have that there exist $\ell_1, \ldots, \ell_n$, T, φ, such that $(P, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T, \varphi)$ and

1. either there exist no φ', T' such that $(Q, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T', \varphi')$,
2. or for all φ', T' such that $(Q, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T', \varphi')$ we have that $\varphi \not\approx_s \varphi'$.

In the first case, $P \not\approx_{ct} Q$, contradicting our hypothesis. In the second case, as $\varphi \not\approx_s \varphi'$, there exist r, r' such that $(r = r')\varphi$ and $(r \neq r')\varphi'$ (or vice-versa, the other case is symmetric). As $P \sqsubseteq_w Q$, we have that there exist T'', φ'' such that $(Q, \emptyset) \stackrel{\ell_1,\ldots,\ell_n}{\Longrightarrow} (T'', \varphi'')$ and $\varphi \sqsubseteq_s \varphi''$. Hence, we have $(r = r')\varphi''$. As Q is determinate, we have that $\varphi' \approx_s \varphi''$. This yields a contradiction, as $(r \neq r')\varphi'$ and $(r = r')\varphi''$ would imply $\varphi' \not\approx_s \varphi''$. □

Additionally, we introduce a more fine-grained notion of trace equivalence, denoted $\approx_{ft}$.

Definition 9. *Given ground processes P and Q, we say that $P \sqsubseteq_{ft} Q$ whenever for all trace $T \in P$ there exists a trace $T' \in Q$ such that $T \approx_t T'$. We say that $P \approx_{ft} Q$ if $P \sqsubseteq_{ft} Q$ and $Q \sqsubseteq_{ft} P$.*

It follows directly form the definition that $\approx_{ft} \subset \approx_t$. The difference between these two relations is illustrated by the following example.

Example 6. Let P and Q be ground processes defined as follows:

$$\begin{aligned} P = \{ & \mathbf{out}(c, enc(a,k)).\mathbf{out}(c, enc(b,k)).\mathbf{in}(c,x).[x = enc(a,k)]\mathbf{out}(c,k), \\ & \mathbf{out}(c, enc(a,k)).\mathbf{out}(c, enc(b,k)).\mathbf{in}(c,x).[x = enc(b,k)]\mathbf{out}(c,k)\} \\ Q = \{ & \mathbf{out}(c, enc(a,k)).\mathbf{out}(c, enc(b,k)).\mathbf{in}(c,x).[x = enc(dec(x,k),k)]\mathbf{out}(c,k)\} \end{aligned}$$

where $k \in \mathcal{N}$ is a private name and a, b are constants. The test $x = enc(dec(x,k),k)$ simply checks whether x is an encryption with key k. It is not difficult to see that $P \approx_t Q$ but $P \not\approx_{ft} Q$.

As already mentioned our procedure is able check $\approx_{ct}$ which coincides with $\approx_t$ when processes are determinate. In the case where processes are not determinate we can use our procedure to check $\approx_{ct}$ and $\approx_{ft}$ in order to over- and under-approximate $\approx_t$. Indeed, as traces are determinate processes a procedure for checking $\approx_{ct}$ can be used to verify $\approx_{ft}$.

4 Modeling traces as Horn clauses

Our decision procedure is based on a fully abstract modelling of a trace in first-order Horn clauses. We give the details of this modelling; we start by giving some definitions that we need for defining the predicates used in the logic.

Symbolic labels and symbolic runs. We define the set of *symbolic labels* as

$$\mathsf{SLabels} = \{\mathbf{in}(c,t), \mathbf{out}(c), \mathbf{test} \mid t \in \mathsf{SMessages}, c \in \mathcal{C}\}$$

and the set of *symbolic runs* as the set of finite sequences of symbolic labels (see Figure 1). The empty sequence is denoted by ϵ. We will often be lazy and write $\ $ (empty space) for ϵ. Intuitively, a symbolic label stands for a set of possible labels, and a symbolic run stands for a set of possible runs of the protocol.

Symbolic Recipes. We assume a set $\mathcal{Y}$ of *recipe variables* disjoint from $\mathcal{X}$. The set of terms $\mathcal{T}_{\mathcal{F},\mathcal{M},\mathcal{W},\mathcal{Y}}$ shall be called *symbolic recipes* and denoted by $\mathsf{SRecipes}$. We use capital letters X, Y, Z to range over $\mathcal{Y}$. Intuitively, a symbolic recipe stands for a set of recipes.

We can extend the definition of substitutions to include variables from $\mathcal{Y}$ in its domain. However, we only consider substitutions that map variables in $\mathcal{Y}$ to $\mathsf{SRecipes}$. A ground substitution must map variables in $\mathcal{Y}$ to $\mathsf{Recipes}$. The notion of most general unifiers is extended to symbolic recipes as expected.

Predicates. The predicates used in our modelling and the semantics of the predicates are given in Figure 1. The predicates are interpreted over a triple– a trace T, a frame φ and a substitution σ. We have four kinds of predicates of the logic, all of whom have a symbolic recipe as an argument. Intuitively, the *reachability predicate* r_w says that each run represented by w is possible. The intruder knowledge predicate $\mathsf{k}_w(R,t)$ says that whenever a run represented by w happens, the (symbolic) message t can be constructed by the intruder using the (symbolic) recipe R. The identity predicate $\mathsf{i}_w(R,R')$ says that whenever the (symbolic) run SR happens, the (symbolic) recipes R and R' are recipes for the same (symbolic) term. The reachable identity predicate $\mathsf{ri}_w(R,R')$ is a short form for the conjunction of the predicates r_w and $\mathsf{i}_w(R,R')$.

Symbolic Runs ($\ell \in \mathsf{SLabels}$):
$u, v, w := \epsilon \mid \ell, w$

Predicates ($w \in \mathsf{SRuns}, R \in \mathsf{SRecipes}, t \in \mathsf{SMessages}$) :

r_w	(Reachability predicate)
$\mathsf{k}_w(R,t)$	(Intruder knowledge predicate)
$\mathsf{i}_w(R,R')$	(Identity predicate)
$\mathsf{ri}_w(R,R')$	(Reachable identity predicate)

Semantics ($\ell_i \in \mathsf{SLabels}, R \in \mathsf{SRecipes}, t \in \mathsf{SMessages}, T \in \mathsf{GndTraces}, \varphi \in \mathsf{Frames}, \sigma$ a ground substitution):

$(T,\varphi_0,\sigma) \models \mathsf{r}_{\ell_1,\ldots,\ell_i}$	if $(T,\varphi_0) \xrightarrow{L_1} (T_1,\varphi_1) \xrightarrow{L_2} \ldots \xrightarrow{L_n} (T_n,\varphi_n)$ such that $\ell_i\sigma =_{\mathsf{R}} L_i\varphi_{i-1}$ for all $1 \le i \le n$
$(T,\varphi_0,\sigma) \models \mathsf{k}_{\ell_1,\ldots,\ell_i}(R,t)$	if when $(T,\varphi_0) \xrightarrow{L_1} (T_1,\varphi_1) \xrightarrow{L_2} \ldots \xrightarrow{L_n} (T_n,\varphi_n)$ such that $\ell_i\sigma =_{\mathsf{R}} L_i\varphi_{i-1}$ for all $1 \le i \le n$ then $\varphi'_n \vdash^{R\sigma} t\sigma$
$(T,\varphi_0,\sigma) \models \mathsf{i}_{\ell_1,\ldots,\ell_i}(R,R')$	if there exists t s.t. $(T,\varphi_0,\sigma) \models \mathsf{k}_{\ell_1,\ldots,\ell_i}(R,t)$ and $(T,\varphi_0,\sigma) \models \mathsf{k}_{\ell_1,\ldots,\ell_i}(R',t)$
$(T,\varphi_0,\sigma) \models \mathsf{ri}_{\ell_1,\ldots,\ell_i}(R,R')$	if $(T,\varphi_0,\sigma) \models \mathsf{r}_{\ell_1,\ldots,\ell_i}$ and $(T,\varphi_0,\sigma) \models \mathsf{i}_{\ell_1,\ldots,\ell_i}(R,R')$

Fig. 1: Predicates

Formulas and statements. We consider first-order formulas built using the above predicates and the usual connectives (conjunction, disjunction, negation, implication, existential and universal quantification). As in the case of predicates, a formula is interpreted over a triple consisting of a trace T, a frame φ and a substitution σ; and the semantics is defined as expected. For ground formulas we do not need the substitution σ and when a formula f is ground we simply write $(T,\varphi) \models f$ to denote that this formula holds for (T,φ). If moreover, $dom(\varphi) = \emptyset$, we simply write $T \models f$ for $(T,\emptyset) \models f$.

We now identify a subset of the formulas, which we shall call statements. Statements will take the form of Horn clauses, and we shall be mainly concerned with them.

Definition 10. *A* statement *is a Horn clause of the form $H \Leftarrow B_1,\ldots,B_n$ where:*

1. $H \in \{\mathsf{r}_{l_1,\ldots,l_k}, \mathsf{k}_{l_1,\ldots,l_k}(R,t), \mathsf{i}_{l_1,\ldots,l_k}(R,R'), \mathsf{ri}_{l_1,\ldots,l_k}(R,R')\}$
2. *For each* $1 \le i \le n, B_i = \mathsf{k}_{l_1,\ldots,l_{j_i}}(X_i,t_i)$

for some $l_1,\ldots l_k \in \mathsf{SLabels}$, $t \in \mathsf{SMessages}$, $R,R' \in \mathsf{SRecipes}$, $j_i \le k$, $t_1,\ldots,t_n \in \mathsf{SMessages}$ and $X_1,\ldots,X_n \in \mathcal{Y}$. Furthermore $X_1,\ldots,X_n$ are distinct variables and if $H = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t)$ then $vars(t) \subseteq vars(t_1,\ldots,t_n)$.

We implicitly assume that in a Horn clause all variables are universally quantified. Hence, all statements are closed formulas.

4.1 The set of seed statements

As mentioned above, our decision procedure is based on a fully abstract modelling of a trace in first-order Horn clauses. In this section, given a trace T we will give a set of statements $\mathsf{seed}(T)$ which will serve as a starting point for the modelling. We shall also establish that the set of statements $\mathsf{seed}(T)$ is a sound and (partially) complete abstraction of the trace T. In order to formally define $\mathsf{seed}(T)$, we start by fixing some conventions.

Let $T = a_1.a_2.\ldots.a_n$ be a ground trace. We assume w.l.o.g. the following naming conventions:

1. if a_i is a receive action then $a_i = \mathbf{in}(c_i,x_i)$.
2. $x_i \neq x_j$ for any $i \neq j$.

$\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow} \Leftarrow \{\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow)\}_{j\in R(m)}$
for all $0 \le m \le n$
for all $\sigma \in \mathrm{mgu}_{\mathsf{R}}(\{s_k = t_k\}_{k\in T(m)})$
for all $\tau \in \mathrm{variants}(\ell_1\sigma, \ldots, \ell_m\sigma)$

$\mathsf{k}_{\ell_1\tau\downarrow,\ldots,\ell_m\tau\downarrow}(w_{|S(m)|}, t_m\tau\downarrow) \Leftarrow \{\mathsf{k}_{\ell_1\tau\downarrow,\ldots,\ell_{j-1}\tau\downarrow}(X_j, x_j\tau\downarrow)\}_{j\in R(m)}$
for all $m \in S(n)$
for all $\tau \in \mathrm{variants}(\ell_1, \ldots, \ell_m, t_m)$

$\mathsf{k}(c, c) \Leftarrow$
for all public names $c \in \mathcal{M}_0$

$\mathsf{k}_{\ell_1,\ldots,\ell_m}(f(Y_1, \ldots, Y_k), f(y_1, \ldots, y_k)\tau\downarrow) \Leftarrow \{\mathsf{k}_{\ell_1,\ldots,\ell_m}(Y_j, y_j\tau\downarrow)\}_{j\in\{1,\ldots,k\}}$
for all $0 \le m \le n$
for all function symbols f of arity k
for all $\tau \in \mathrm{variants}(f(y_1, \ldots, y_k))$.

Fig. 2: Seed statements

3. if a_i is a send action then $a_i = \mathbf{out}(c_i, t_i)$.
4. if a_i is a test actions then $a_i = [s_i \stackrel{?}{=} t_i]$.

Moreover, for each $1 \le i \le n$ let $\ell_i \in \mathsf{SLabels}$ be as follows:

$$\ell_i = \begin{cases} \mathbf{in}(c_i, x_i) & \text{if } a_i = \mathbf{in}(c_i, x_i) \\ \mathbf{out}(c_i) & \text{if } a_i = \mathbf{out}(c_i, t_i) \\ \mathbf{test} & \text{if } a_i = [s_i \stackrel{?}{=} t_i] \end{cases}.$$

For each $0 \le m \le n$, let the sets $R(m)$, $S(m)$ and $T(m)$ respectively denote the indices of the receive actions, send actions and test actions amongst $a_1, \ldots, a_m$. Formally,

$$\begin{cases} R(m) = \{i \mid 1 \le i \le m, a_i = \mathbf{in}(c_i, x_i)\} \\ S(m) = \{i \mid 1 \le i \le m, a_i = \mathbf{out}(c_i, t_i)\} \\ T(m) = \{i \mid 1 \le i \le m, a_i = [s_i \stackrel{?}{=} t_i]\} \end{cases}.$$

Given a set of public names $\mathcal{M}_0 \subseteq \mathcal{M}$, *set of seed statements* associated to T and $\mathcal{M}_0$, denoted $\mathsf{seed}(T, \mathcal{M}_0)$, is defined to be the set of statements given in Figure 2. If $\mathcal{M}_0 = \mathcal{M}$, then $\mathsf{seed}(T, \mathcal{M})$ is said to be the set of seed statements associated to T and in this case we write $\mathsf{seed}(T)$ as a shortcut for $\mathsf{seed}(T, \mathcal{M})$.

Remark 2. Please note that while constructing the set of seed statements, we apply the most general unifier modulo R to all tests. In addition, we also apply finite variants. This allows us to *get rid* of rewriting in our procedure.

We shortly show that the set of seed statements is a sound and (partially) complete modelling of a trace. However, we need one more definition to state this fact.

Definition 11. *Let K be a set of statements. We define $\mathcal{H}(K)$ to be the smallest set of ground terms such that:*

$$\textsc{Simple Consequence}\ \frac{f = \Big(H \Leftarrow B_1, \ldots, B_n\Big) \in K \qquad \sigma \text{ grounding for } f \qquad B_1\sigma \in \mathcal{H}(K) \qquad \ldots \qquad B_n\sigma \in \mathcal{H}(K)}{H\sigma \in \mathcal{H}(K)}$$

$$\textsc{ExtendK}\ \frac{\mathsf{k}_u(R, t) \in \mathcal{H}(K)}{\mathsf{k}_{uv}(R, t) \in \mathcal{H}(K)} \qquad \textsc{ExtendI}\ \frac{\mathsf{i}_u(R, R') \in \mathcal{H}(K)}{\mathsf{i}_{uv}(R, R') \in \mathcal{H}(K)}$$

(Equivalently, $\mathcal{H}(K)$ is the least Herbrand model of $K \cup \{\mathsf{k}_{\ell_1,\ldots,\ell_{n+1}}(X,x) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_n}(X,x)\}_{n\in\mathbb{N}} \cup \{\mathsf{i}_{\ell_1,\ldots,\ell_{n+1}}(X_1,X_2) \Leftarrow \mathsf{i}_{\ell_1,\ldots,\ell_n}(X_1,X_2)\}_{n\in\mathbb{N}}$.)

We show that as far as reachability predicates and intruder knowledge predicates are concerned, the set $\mathsf{seed}(T)$ is a complete abstraction of a trace.

Theorem 2. *Let T be a ground trace.*

- *(Soundness.) For any statement $f \in \mathsf{seed}(T) \cup \mathcal{H}(\mathsf{seed}(T))$, $T \models f$.*
- *(Completeness.) If $(T,\emptyset) \xrightarrow{L_1,\ldots,L_m} (S,\varphi)$ then:*
  1. *$\mathsf{r}_{L_1\varphi\downarrow,\ldots,L_m\varphi\downarrow} \in \mathcal{H}(\mathsf{seed}(T))$.*
  2. *if $\varphi \vdash^R t$ then $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_m\varphi\downarrow}(R,t\downarrow) \in \mathcal{H}(\mathsf{seed}(T))$.*

Remark 3. Please note that the set $\mathsf{seed}(T)$ is only partially complete in that we have not shown that in Theorem 2 that if $\varphi \vdash^R t$ and $\varphi \vdash^{R'} t$ then $\mathsf{i}_{L_1\varphi\downarrow,\ldots,L_m\varphi\downarrow} \in \mathcal{H}(\mathsf{seed}(T))$.

We will shortly show how the completeness of $\mathsf{seed}(T)$ can be built upon to achieve a) full abstraction of the trace T and a b) procedure for checking equivalences $\approx_{ct}$ and $\sqsubseteq_{ft}$.

5 Procedure for deciding trace equivalence

We shall now show give a procedure for deciding trace equivalence. At a high level, this consists of two steps.

1. A saturation procedure which constructs a set of *simple* statements from the set $\mathsf{seed}(T)$ which we will call *solved* statements. The saturation procedure ensures that the set of solved statements is a complete abstraction of T.
2. Given two processes P and Q, we saturate the set of seed statements for traces of P and Q and then use the solved statements to decide whether P and Q are trace equivalent.

We shall now give the details of the procedure. We start by the saturation procedure.

5.1 Knowledge bases and saturation

The saturation procedure manipulates a set of statements called a knowledge base:

Definition 12. *Given a statement $f = H \Leftarrow B_1,\ldots,B_n$,*

- *f is said to be* solved *if for all $1 \le i \le n$, $B_i = \mathsf{k}_{\ell_1,\ldots,\ell_{j_i}}(X_i,x_i)$ for some variables $x_i \in \mathcal{X}, X_i \in \mathcal{Y}$.*
- *f is said to be* well-formed *if whenever it is solved and $H = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t)$, we have that $t \notin \mathcal{X}$.*

A set of well-formed *statements is called a* knowledge base. *If K is a knowledge base, we define $K_{\mathsf{solved}} = \{f \in K \mid f \text{ is solved }\}$ to be the knowledge base restricted to the solved statements.*

Given an initial knowledge base K, the saturation procedure procedure produces another knowledge base $\mathsf{sat}(K)$. The saturation procedure proceeds as follows. First new statements are *generated* and then the knowledge base is *updated* with the new statements. This two-step process continues until a fixed-point is achieved. We describe the two steps in the procedure.

Generating new statements. Given a knowledge base K, new statements f are generated by applying the rules in Figure 3.

$$\textsc{Resolution}\ \frac{\begin{array}{c} f \in K, g \in K_{\mathsf{solved}}, \qquad f = \Big(H \Leftarrow \mathsf{k}_{uv}(X,t), B_1, \ldots, B_n\Big) \\ g = \Big(\mathsf{k}_w(R,t') \Leftarrow B_{n+1}, \ldots, B_m\Big) \qquad \sigma = \mathrm{mgu}(\mathsf{k}_u(X,t), \mathsf{k}_w(R,t')) \qquad t \notin \mathcal{X} \end{array}}{K = K \oplus h \text{ where } h = \Big((H \Leftarrow B_1, \ldots, B_m)\sigma\Big)}$$

$$\textsc{Equation}\ \frac{\begin{array}{c} f, g \in K_{\mathsf{solved}}, \\ f = \Big(\mathsf{k}_u(R,t) \Leftarrow B_1, \ldots, B_n\Big) \qquad g = \Big(\mathsf{k}_{u'v'}(R',t') \Leftarrow B_{n+1}, \ldots, B_m\Big) \qquad \sigma = \mathrm{mgu}(\mathsf{k}_u(\_,t), \mathsf{k}_{u'}(\_,t')) \end{array}}{K = K \oplus h \text{ where } h = \Big((\mathsf{i}_{u'v'}(R,R') \Leftarrow B_1, \ldots, B_m)\sigma\Big)}$$

$$\textsc{Test}\ \frac{f, g \in K_{\mathsf{solved}}, \qquad f = \Big(\mathsf{i}_u(R,R') \Leftarrow B_1, \ldots, B_n\Big) \qquad g = \Big(\mathsf{r}_{u'v'} \Leftarrow B_{n+1}, \ldots, B_m\Big) \qquad \sigma = \mathrm{mgu}(u,u')}{K = K \oplus h \text{ where } h = \Big((\mathsf{ri}_{u'v'}(R,R') \Leftarrow B_1, \ldots, B_m)\sigma\Big)}$$

Fig. 3: Saturation rules

Update. The first step while updating the knowledge base by f is to convert f into a canonical form:

Definition 13. *Given a solved deduction statement f, we define the* canonical form *of f to be the statement $f\Downarrow$ obtained by first applying Rule* Rename *below as many times as possible and then applying Rule* Remove *below as many times as possible:*

$$\textsc{Rename}\ \frac{H \Leftarrow \mathsf{k}_u(X,x), \mathsf{k}_{uv}(Y,x), B_1, \ldots, B_n}{(H \Leftarrow \mathsf{k}_u(X,x), B_1, \ldots, B_n)\{Y \mapsto X\}}$$

$$\textsc{Remove}\ \frac{H \Leftarrow \mathsf{k}_u(X,x), B_1, \ldots, B_n \qquad x \notin \mathit{vars}(H)}{H \Leftarrow B_1, \ldots, B_n}$$

For any other type of statement f, the canonical form $f\Downarrow$ is defined to be equal to f.

It is easy to see that any fact f can be converted into a canonical form. After a canonical form has been obtained, we perform another check before $f\Downarrow$ can be added to the knowledge base. Intuitively, this check ensures that we add enough identity predicates in the the knowledge base.

Definition 14. *The set of* consequences *of a knowledge base K, denoted* $\mathbf{conseq}(K)$*, is the smallest set such that:*

$$\textsc{Axiom}\ \frac{}{\mathsf{k}_{uv}(R,t) \Leftarrow \mathsf{k}_u(R,t), B_1, \ldots, B_m \in \mathbf{conseq}(K)}$$

$$\textsc{Res}\ \frac{\begin{array}{c} H \Leftarrow B_1, \ldots, B_n \in K \qquad \sigma \text{ a substitution} \\ B_1\sigma \Leftarrow C_1, \ldots, C_m \in \mathbf{conseq}(K) \qquad \ldots \qquad B_n\sigma \Leftarrow C_1, \ldots, C_m \in \mathbf{conseq}(K) \end{array}}{H\sigma \Leftarrow C_1, \ldots, C_m \in \mathbf{conseq}(K)}$$

Given a knowledge base K and a statement f, the *update of K by f*, denoted $K \oplus f$, is defined to be $K \cup \{f\Downarrow\}$ if the head of f is not of the form $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t)$. Otherwise, let

$$f\Downarrow = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,t_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,t_n)$$

and

$$K \oplus f = \begin{cases} K \cup \{f\Downarrow\} & \text{if } f \text{ is solved and for any } R' \text{ and } 1 \le i_1, i_2, \ldots, i_n \le k \\ & \mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, t_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, t_n) \notin K' \\ K \cup \{\mathsf{i}_{\ell_1,\ldots,\ell_k}(R, R') & \text{if } f \text{ is solved and } R' \text{ and } 1 \le i_1, i_2, \ldots, i_n \le k \text{ are such that} \\ \quad \Leftarrow \{\mathsf{k}_{\ell_1,\ldots,\ell_{i_j}}(X_j, t_j)\}_{j \in \{1,\ldots,n\}}\} & \mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, t_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, t_n) \in K' \\ K \cup \{f\Downarrow\} & \text{if } f \text{ is not solved} \end{cases}$$

where $K' = \mathbf{conseq}(K_{\mathsf{solved}})$.

Please note that update is not a function, namely that there may be several $R', i_1, \ldots, i_n$ such that $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, t_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, t_n) \in \mathbf{conseq}(K_{\mathsf{solved}})$. However, we need to compute only one such $R', i_1, \ldots, i_n$.

Initial knowledge base. One question that naturally arises is what is the initial knowledge base for the saturation procedure. Given a trace T, the initial knowledge base for the saturation procedure is defined as follows.

Definition 15. *Given a set of statements S, the* initial knowledge base *associated to S, denoted $K_i(S)$, is defined to be the empty knowledge base updated by the set S, i.e., $K_i(S) = \emptyset \oplus_{f \in S} f$. If T is a ground trace, we write $K_i(T)$ for $K_i(\mathsf{seed}(T))$.*

Please observe that $K_i(T)$ depends on the order in which statements in $\mathsf{seed}(T)$ are updated. The exact order, however, is not important and our results shall hold regardless of the order chosen. The saturation procedure takes $K_i(T)$ as an input and produces a knowledge base $\mathsf{sat}(K_i(T))$. The reason for choosing $K_i(T)$ instead of $\mathsf{seed}(T)$ as the starting point of the saturation procedure is that $\mathsf{seed}(T)$ may not be a knowledge base (recall that a knowledge base is a set of well-formed statements). The set $K_i(T)$ is, however, a knowledge base.

Proposition 2. *Given a trace T, the set $K_i(T)$ is a knowledge base.*

Soundness and completeness of the saturation procedure. We shall now show that the set of solved statements in $\mathsf{sat}(K_i(T))$ is a sound and complete abstraction of a trace T. We need one more definition which extends $\mathcal{H}(K)$ and allows us to establish that $\mathsf{sat}(K_i(T))$ is a complete abstraction of T.

Definition 16. *Let K be a set of statements. We define $\mathcal{H}_{\mathsf{e}}(K)$ to be the smallest set of ground terms such that $\mathcal{H}(K) \subseteq \mathcal{H}_{\mathsf{e}}(K)$ and:*

$$\textsc{Refl}\ \frac{}{\mathsf{i}_w(R, R) \in \mathcal{H}_{\mathsf{e}}(K)} \qquad \textsc{Sym}\ \frac{\mathsf{i}_w(R_1, R_2) \in \mathcal{H}_{\mathsf{e}}(K)}{\mathsf{i}_w(R_2, R_1) \in \mathcal{H}_{\mathsf{e}}(K)} \qquad \textsc{Tran}\ \frac{\mathsf{i}_w(R_1, R_2) \in \mathcal{H}_{\mathsf{e}}(K) \qquad \mathsf{i}_w(R_1, R_3) \in \mathcal{H}_{\mathsf{e}}(K)}{\mathsf{i}_w(R_1, R_3) \in \mathcal{H}_{\mathsf{e}}(K)}$$

$$\textsc{Cong}\ \frac{\mathsf{i}_w(R_1, R_1') \in \mathcal{H}_{\mathsf{e}}(K), \cdots, \mathsf{i}_w(R_n, R_n') \in \mathcal{H}_{\mathsf{e}}(K) \qquad f \in \mathcal{F}, ar(f) = n}{\mathsf{i}_w(f(R_1, \ldots R_n), f(R_1', \ldots R_n')) \in \mathcal{H}_{\mathsf{e}}(K)} \qquad \textsc{Extend}\ \frac{\mathsf{i}_u(R, R') \in \mathcal{H}_{\mathsf{e}}(K)}{\mathsf{i}_{uv}(R, R') \in \mathcal{H}_{\mathsf{e}}(K)}$$

$$\textsc{Equational Consequence}\ \frac{\mathsf{k}_w(R, t) \in \mathcal{H}(K) \qquad \mathsf{i}_w(R, R') \in \mathcal{H}_{\mathsf{e}}(K)}{\mathsf{k}_w(R', t) \in \mathcal{H}_{\mathsf{e}}(K)}$$

We have that the set of solved statements produced by the saturation procedure is a sound and complete abstraction of the trace T.

Theorem 3. *Let T be a ground trace and let $K = \mathsf{sat}(K_i(T))$.*

- *(Soundness.) For any $f \in K \cup \mathcal{H}_{\mathsf{e}}(K)$, $T \models f$.*
- *(Completeness.) If $(T, \emptyset) \xrightarrow{L_1,\ldots,L_n} (S, \varphi)$ then*
  1. *$\mathsf{r}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow} \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$.*
  2. *if $\varphi \vdash^R t$ then $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, t\downarrow) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$.*
  3. *if $\varphi \vdash^R t$ and $\varphi \vdash^{R'} t$, then $\mathsf{i}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, R') \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$.*

Effectiveness of the saturation procedure. We have shown that the set of solved statements in $\mathsf{sat}(K_i(T))$ form a sound and complete abstraction for the trace T. However this set is infinite and may not be effectively computable. This may be because of following reasons.

- The set $\mathsf{seed}(T)$ for a ground trace T is infinite. Hence the saturation procedure may continue forever. We will, however, shortly show that for the saturation procedure we only need to consider the saturation of the set $K_i(\mathsf{seed}(T, \mathcal{M}_0))$ where $\mathcal{M}_0$ is the set of public names occurring in T (see Lemma 1). The set $\mathsf{sat}(K_i(T))$ can then be computed from this set. Since the set $K_i(\mathsf{seed}(T, \mathcal{M}_0))$ is finite, this means that all intermediate knowledge bases in the saturation procedure are finite.
- For the update rule, we have to check that given a knowledge base K, term t, labels $\ell_1, \ldots, \ell_k$, indices $1 \leq i_1, \ldots i_n, \leq k$, variables $x_1, \ldots, x_n \in \mathcal{X}$ and recipe variables $X_1, \ldots, X_n \in \mathcal{Y}$, whether
$$\exists R.\ \mathsf{k}_{\ell_1,\ldots,\ell_k}(R, t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n) \in \mathbf{conseq}(K_{\mathsf{solved}}).$$
Furthermore, if the check succeeds then we have to compute one such R. We will show that can be achieved if K is finite (see Lemma 2).
- The saturation procedure may itself not terminate even if the initial knowledge base is finite. As pointed out in the Introduction, we conjecture that the saturation procedure terminates, but were unable to show the termination.

The following lemma allows us to compute the $\mathsf{sat}(K_i(T))$ from the set $\mathsf{sat}(K_i(seed(M_0, T)))$ where M_0 is the set of public names occurring in T.

Lemma 1. *Let T be a trace and $M_T \subseteq \mathcal{M}$ be the public names occurring in T. Let*
$$K_{\mathcal{M}} = \{\{\mathsf{k}(m, m) \Leftarrow\}_{m \in \mathcal{M}} \cup \{\mathsf{i}(m, m) \Leftarrow\}_{m \in \mathcal{M}} \cup \{\mathsf{ri}(m, m) \Leftarrow\}_{m \in \mathcal{M}}\}.$$
Then $\mathsf{sat}(K_i(T)) = \mathsf{sat}(K_i(seed(M_T, T))) \cup K_{\mathcal{M}}$.

The following lemma implies that the update step terminates if we only have a finite number of statements in the knowledge base.

Lemma 2. *Given a finite set of solved statements K, term t, labels $\ell_1, \ldots, \ell_k$, indices $1 \leq i_1, \ldots i_n \leq k$, variables $x_1, \ldots, x_n \in \mathcal{X}$ and recipe variables $X_1, \ldots, X_n \in \mathcal{Y}$, it is decidable if there is an R such that $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R, t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n) \in \mathbf{conseq}(K_{\mathsf{solved}})$. If the answer to the decision procedure is "Yes," then we can compute one such R.*

5.2 Algorithm

Theorem 4 directly yields an algorithm to decide trace inclusion for determinate processes.

Theorem 4. *Let T be a trace and let P be a determinate process. Let K be the set of solved statements from a saturated knowledge base associated to T. Then $T \sqsubseteq_w P$ iff the following tests hold:*

$$\textsc{Reachability}\ \frac{\begin{array}{c}\left(\mathsf{r}_{l_1,\ldots,l_n} \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{i \in \{1,\ldots,m\}}\right) \in K \\ c_1, \ldots, c_k \text{ fresh constants} \qquad \sigma : vars(l_1, \ldots, l_n) \to \{c_1, \ldots, c_k\} \text{ is a bijection} \\ \mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(K) \text{ for all } i \text{ such that } l_i = \boldsymbol{in}(t_i) \\ M_i = l_i \text{ if } l_i \in \{\boldsymbol{test}, \boldsymbol{out}(\_)\} \qquad M_i = \boldsymbol{in}(d_i, R_i) \text{ if } l_i = \boldsymbol{in}(d_i, t_i)\end{array}}{(P, \emptyset) \xrightarrow{M_1,\ldots,M_n} (T', \varphi)}$$

$$\textsc{Identity}\ \frac{\begin{array}{c}\left(\mathsf{ri}_{l_1,\ldots,l_n}(R, R') \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{i \in \{1,\ldots,m\}}\right) \in K \\ c_1, \ldots, c_k \text{ fresh constants} \qquad \sigma : vars(l_1, \ldots, l_n) \to \{c_1, \ldots, c_k\} \text{ is a bijection} \\ \mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(K) \text{ for all } i \text{ such that } l_i = \boldsymbol{in}(t_i) \\ M_i = l_i \text{ if } l_i \in \{\boldsymbol{test}, \boldsymbol{out}(\_)\} \qquad M_i = \boldsymbol{in}(d_i, R_i) \text{ if } l_i = \boldsymbol{in}(d_i, t_i)\end{array}}{(P, \emptyset) \xrightarrow{M_1,\ldots,M_n} (T', \varphi) \text{ such that } (R\omega = R'\omega)\varphi \text{ where } \omega = \{X_i \mapsto x_i\sigma\}}$$

Note that performing the tests requires deciding if, given t, and w, $\mathsf{k}_w(R,t) \in \mathcal{H}(K)$ for some recipe R for a knowledge base K containing only solved statements.

It is easy to see that this is equivalent to checking if $\left(\mathsf{k}_w(R,t) \Leftarrow\right) \in \mathbf{conseq}(K)$ and we have already shown that there is an effective procedure for this (which finds an R if such an R exists).

6 Prototype and case studies

We implemented the procedure for checking equivalence in a prototype, AKiSs (Active Knowledge in Security protocols). AKiSs is written in OCaml and has about 2000 lines of source code, including code for computing complete sets of finite variants and complete sets of equational unifiers. We used AKiSs to verify the equivalences in Examples 5 and 6. Using AKiSs we were able to verify strong secrecy for Denning-Sacco-Blanchet [14] and Needham-Schroeder-Lowe (NSL) [44], resistance to guessing attacks in the EKE protocol [12], and, more interestingly, anonymity of the FOO [39] and Okamoto [46] electronic voting protocols. [3] To our knowledge, AKiSs is the only tool that can verify FOO and Okamoto automatically. We discuss each of these examples in more details below. AKiSs along with all the discussed examples is available on: `http://www.lsv.ens-cachan.fr/~ciobaca/akiss/`.

To ease protocol specification, the process calculus syntax used for specifying protocol we allow for an operator *interleave*, denoted $\|$, which models parallel composition of processes and an operator *sequence*, denoted ;, for modeling protocols structured in phases. These constructs are merely syntactic sugar and are defined as follows. Given processes P and Q we define $P;Q$ as the sequential composition of each trace in P with each trace in Q, i.e.,

$$P;Q = \{T_1.T_2 \mid T_1 \in P, T_2 \in Q\}$$

Let ϵ denote the empty, a_1, a_2 be actions and T, T_1, T_2 traces. The parallel composition of two traces is the process defined inductively as

$$T \parallel \epsilon = \epsilon \parallel T = T$$
$$a_1.T_1 \parallel a_2.T_2 = \{a_1.a_2; (T_1 \parallel T_2), a_2.a_1; (T_1 \parallel T_2)\}$$

The parallel composition is then naturally lifted to process, i.e. $P \parallel Q = \{T_1 \parallel T_2 \mid T_1 \in P, T_2 \in Q\}$.

We now give more details about our case studies.

Strong flavors of confidentiality. The *strong secrecy* property was introduced by Blanchet in [14] and we rephrase it here in our setting. Let P be a protocol with x as the only free variable of P. Then x is said to be *strongly secret* if

$$\mathbf{in}(c, x_1).\mathbf{in}(c, x_2).(P\{x \mapsto x_1\}) \approx_t \mathbf{in}(c, x_1).\mathbf{in}(c, x_2).(P\{x \mapsto x_2\}).$$

Intuitively, the attacker cannot distinguish the processes using variables x_1 and x_2 even though it can choose arbitrary (public) values for these variables. The definition generalizes to multiple variables in the expected way. We illustrate this property on a Denning-Sacco-Blanchet protocol. Informally, the protocol can be described as follows.

$$\mathsf{A} \to \mathsf{B} : \mathsf{aenc}(\mathsf{sign}(\mathsf{pair}(\mathsf{pk}(ska), \mathsf{pair}(\mathsf{pk}(skb), k))), ska), \mathsf{pk}(skb))$$
$$\mathsf{B} \to \mathsf{A} : \mathsf{enc}(x, k)$$

A sends to B a fresh symmetric session key k together with A's and B's public keys. This is signed with A's secret key and (asymmetrically) encrypted with B's public key. Upon receiving this message, B decrypts it, checks the signature and uses the fresh session key to symmetrically encrypt a secret x. The detailed protocol model is given in Figure 4. We used AKiSs to verify this protocol for strong secrecy of x (with one session

[3] Please note that as defined in [46], modeling of Okamoto's protocol requires private channels. As we do not have private channels in our calculus, we transform the protocol so that every message sent by honest participants on a private channel is sent encrypted under a key not known to the adversary

Equational Theory:

$$\begin{array}{ccc} \mathsf{fst}(\mathsf{pair}(x,y)) \to x & \mathsf{adec}(\mathsf{aenc}(x,\mathsf{pk}(y)),y) \to x & \mathsf{check}(\mathsf{sign}(x,y),pk(y)) \to ok \\ \mathsf{snd}(\mathsf{pair}(x,y)) \to y & \mathsf{dec}(\mathsf{enc}(x,y),y) \to x & \mathsf{msg}(\mathsf{sign}(x,y)) \to x \end{array}$$

Processes:

$$\begin{aligned} P_i &= \mathit{Setup}; I_i \qquad (i \in \{1,2\}) \\ \mathit{Setup} &= \mathbf{out}(c,\mathsf{pk}(skA)).\mathbf{out}(c,\mathsf{pk}(ekB)).\mathbf{in}(c,x_1).\mathbf{in}(c,x_2) \\ I_i &= A \parallel B_i \\ A &= \mathbf{out}(c,\mathsf{aenc}(\mathsf{sign}(\mathsf{pair}(\mathsf{pk}(skA),\mathsf{pair}(\mathsf{pk}(ekB),k)),skA),\mathsf{pk}(ekB))) \\ B_i &= \mathbf{in}(c,z).[\mathsf{check}(\mathsf{adec}(z,ekB),\mathsf{pk}(skA)) \stackrel{?}{=} ok].[\mathsf{fst}(\mathsf{msg}(\mathsf{adec}(z,ekB))) \stackrel{?}{=} \mathsf{pk}(skA)]. \\ &\quad [\mathsf{fst}(\mathsf{snd}(\mathsf{msg}(\mathsf{adec}(z,ekB)))) \stackrel{?}{=} \mathsf{pk}(ekB)].\mathbf{out}(c,\mathsf{enc}(x_i,\mathsf{snd}(\mathsf{snd}(\mathsf{msg}(\mathsf{adec}(z,ekB)))))) \end{aligned}$$

Fig. 4: Formal description of the protocol by Blanchet [14]

of A and B). This protocol is determinate, and hence we used $\approx_{ct}$ to verify that $P_1 \approx_{ct} P_2$. The verification succeeds as expected.

A variant of the protocol [14] consists in letting A also send out a secret y encrypted with k changing the first message to

$$\mathsf{A} \to \mathsf{B} : \mathsf{pair}(\mathsf{aenc}(\mathsf{sign}(\mathsf{pair}(\mathsf{pk}(ska),\mathsf{pair}(\mathsf{pk}(skb,k))),ska),\mathsf{pk}(skb)),\mathsf{enc}(y,k))$$

In this case the protocol does not respect strong secrecy of x, y as, by choosing $x_1 = y_1$ and $x_2 \neq y_2$, the attacker can distinguish the two situations by testing the equality of the encryptions of x and y. The detailed model is given in Figure 5. This attack is again found by AKISS.

Equational Theory:

$$\begin{array}{ccc} \mathsf{fst}(\mathsf{pair}(x,y)) \to x & \mathsf{adec}(\mathsf{aenc}(x,\mathsf{pk}(y)),y) \to x & \mathsf{check}(\mathsf{sign}(x,y),pk(y)) \to ok \\ \mathsf{snd}(\mathsf{pair}(x,y)) \to y & \mathsf{dec}(\mathsf{enc}(x,y),y) \to x & \mathsf{msg}(\mathsf{sign}(x,y)) \to x \end{array}$$

Processes:

$$\begin{aligned} P_i &= \mathit{Setup}; I_i \qquad (i \in \{1,2\}) \\ \mathit{Setup} &= \mathbf{out}(c,\mathsf{pk}(skA)).\mathbf{out}(c,\mathsf{pk}(ekB)).\mathbf{in}(c,x_1).\mathbf{in}(c,x_2).\mathbf{in}(c,y_1).\mathbf{in}(c,y_2) \\ I_i &= A_i \parallel B_i \\ A_i &= \mathbf{out}(c,\mathsf{pair}(\mathsf{aenc}(\mathsf{sign}(\mathsf{pair}(\mathsf{pk}(skA),\mathsf{pair}(\mathsf{pk}(ekB),k)),skA),\mathsf{pk}(ekB))),\mathsf{enc}(y1,k)) \\ B_i &= \mathbf{in}(c,z).[\mathsf{check}(\mathsf{adec}(z,ekB),\mathsf{pk}(skA)) \stackrel{?}{=} ok].[\mathsf{fst}(\mathsf{msg}(\mathsf{adec}(z,ekB))) \stackrel{?}{=} \mathsf{pk}(skA)]. \\ &\quad [\mathsf{fst}(\mathsf{snd}(\mathsf{msg}(\mathsf{adec}(z,ekB)))) \stackrel{?}{=} \mathsf{pk}(ekB)].\mathbf{out}(c,\mathsf{enc}(x_i,\mathsf{snd}(\mathsf{snd}(\mathsf{msg}(\mathsf{adec}(z,ekB)))))) \end{aligned}$$

Fig. 5: Formal description of the variant protocol by Blanchet [14]

AKISS also verifies strong secrecy of the nonce generated by the responder in the Needham-Schroeder-Lowe (NSL) [44] protocol. The NSL protocol is a two-way handshake protocol relying only on public encryption of fresh nonces and can be informally described as follows.

$$\begin{aligned} &\mathsf{A} \to \mathsf{B} : \mathsf{aenc}(\mathsf{pair}(n_a, A), \mathsf{pk}(skb)) \\ &\mathsf{B} \to \mathsf{A} : \mathsf{aenc}(\mathsf{pair}(n_b, \mathsf{pair}(n_a, B)), \mathsf{pk}(ska)) \\ &\mathsf{A} \to \mathsf{B} : \mathsf{aenc}(\mathsf{pair}(n_b), \mathsf{pk}(skb)) \end{aligned}$$

Once again, the modeling of NSL leads to determinate processes, and we used $\approx_{ct}$ for our verification. The detailed model is given in Figure 7.

Equational Theory:

$$\mathsf{fst}(\mathsf{pair}(x,y)) \to x \qquad \mathsf{snd}(\mathsf{pair}(x,y)) \to y \qquad \mathsf{adec}(\mathsf{aenc}(x,\mathsf{pk}(y),z),y) \to x$$

Processes:

$$\begin{aligned}
P_i &= \mathit{Setup}; I_i \qquad (i \in \{1,2\}) \\
\mathit{Setup} &= \mathbf{out}(c,\mathsf{pk}(skA)).\mathbf{out}(c,\mathsf{pk}(ekB)).\mathbf{out}(c,skD)\mathbf{in}(c,x_1).\mathbf{in}(c,x_2) \\
I_i &= A \parallel B_i \\
A &= \mathbf{out}(c,\mathsf{aenc}(\mathsf{pair}(n_a,a),\mathsf{pk}(skD),r_1)). \\
&\quad\ \mathbf{in}(c,y).[\mathsf{snd}(\mathsf{snd}(\mathsf{adec}(y,skA))) = d]. \\
&\quad\ \mathbf{out}(c,\mathsf{aenc}(\mathsf{fst}(\mathsf{snd}(\mathsf{adec}(y,skA))),\mathsf{pk}(skD),r_2)) \\
B_i &= \mathbf{in}(c,z).[\mathsf{snd}(\mathsf{adec}(z,skB)) = a]. \\
&\quad\ \mathbf{out}(c,\mathsf{aenc}(\mathsf{pair}(\mathsf{fst}(\mathsf{adec}(z,skB)),\mathsf{pair}(x_1,b)),pk(skA),r_3))
\end{aligned}$$

Fig. 6: Formal description of the NSL protocol [44]

This model includes a session of the initiator who is willing to engage with any participant (including the attacker to allow man-in-the-middle attacks) and a session of B who is willing to engage a session with A. Note that if B was willing to start a session with an arbitrary initiator the secrecy of n_b would be trivially broken in a session with the attacker. (In a more complex model one could of course add additional sessions for B with an arbitrary initiator.) We note that for the verification of NSL, one needs to explicitly model randomness for asymmetric encryption since the protocol is insecure if deterministic asymmetric encryption is used. Indeed, as the attacker may choose the value of n_b he could simply recompute the last message and compare it with the message sent by the initiator.

We also used AKiSs to verify the above protocols for *real-or-random* secrecy. Let P be a protocol and $n \in \mathit{names}(P)$. Then n is said to be *real-or-random secret* if

$$P; \mathbf{out}(c,n) \approx_t P; \mathbf{out}(c,n')$$

where n' is a fresh name, i.e. a name that does not appear in P. Real-or-random secrecy is particularly useful to model resistance to offline guessing attacks in password protocols [10]. Intuitively, an offline guessing attacks works in two phases. In the first, online phase the attacker interacts with the protocol P in an arbitrary way. In a second, offline phase the attacker tries all possible passwords against the data recorded in the first phase. Our property states that the attacker cannot distinguish the case where he tests the real password (n) from the case where he tests a wrong password (n'). We show that the EKE protocol [12] is resistant to offline guessing attacks. The protocol can be described informally as follows:

$$\begin{aligned}
&\mathsf{A} \to \mathsf{B} : \mathsf{enc}(\mathsf{pk}(k)), w) \\
&\mathsf{B} \to \mathsf{A} : \mathsf{enc}(\mathsf{aenc}(r,\mathsf{pk}(k)), w) \\
&\mathsf{A} \to \mathsf{B} : \mathsf{enc}(na, r) \\
&\mathsf{B} \to \mathsf{A} : \mathsf{enc}(\langle na, nb\rangle, r) \\
&\mathsf{A} \to \mathsf{B} : \mathsf{enc}(nb, r)
\end{aligned}$$

In the first step A generates a new private session key k and sends the corresponding public key $\mathsf{pk}(k)$ to B, encrypted (using symmetric encryption) with the shared password w. Then, B generates a fresh symmetric session key r, which he encrypts (using asymmetric encryption) with the previously received public key $\mathsf{pk}(k)$. Finally, he encrypts the resulting ciphertext with the password w and sends the result to A. The last three steps perform a handshake to avoid replay attacks. Using AKiSs we have shown that the protocol resists

to offline guessing attacks on the password w. As EKE also leads to determinate processes, we used the $\approx_{ct}$ relation. The detailed description of our model is given in Figure 7.

Equational Theory:

$$\mathsf{fst}(\mathsf{pair}(x,y)) \to x \qquad \mathsf{adec}(\mathsf{aenc}(x,\mathsf{pk}(y),z),y) \to x$$
$$\mathsf{snd}(\mathsf{pair}(x,y)) \to y \qquad \mathsf{dec}(\mathsf{enc}(x,y),y) \to x$$

Processes:

$$\begin{aligned}
P_1 &= (A \parallel B); \mathbf{out}(c,w)\\
P_2 &= (A \parallel B); \mathbf{out}(c,w')\\
A &= \mathbf{out}(c,\mathsf{enc}(\mathsf{pk}(k),w)).\\
&\quad\ \mathbf{in}(c,x_1).\\
&\quad\ \mathbf{out}(c,\mathsf{enc}(n_a,\mathsf{adec}(dec(x_1,w),k))).\\
&\quad\ \mathbf{in}(c,x_2).[\mathsf{fst}(\mathsf{dec}(x_2,\mathsf{adec}(\mathsf{dec}(x_1,w),k))) \stackrel{?}{=} n_a].\\
&\quad\ \mathbf{out}(c,\mathsf{snd}(\mathsf{dec}(x_2,\mathsf{adec}(\mathsf{dec}(x1,w),k)))).\\
B &= \mathbf{in}(c,y_1).\\
&\quad\ \mathbf{out}(c,\mathsf{enc}(\mathsf{aenc}(r,\mathsf{dec}(y_1,w),r_b),w)).\\
&\quad\ \mathbf{in}(c,y_2).\\
&\quad\ \mathbf{out}(c,\mathsf{enc}(\mathsf{pair}(\mathsf{dec}(y_2,r),n_b),r)).
\end{aligned}$$

Fig. 7: Formal description of the EKE protocol [12]

Anonymity for electronic voting protocol. A voting protocol must respect voter privacy: the adversary should not be able to learn how each voter voted. AKiSs can automatically verify voter privacy in the FOO electronic voting protocol [39] and the Okamoto protocol [46]. Voter privacy is naturally modeled as an equivalence property [33, 9]: it is not possible to distinguish the situation where honest voter A votes 'yes' and honest B votes 'no' from the situation that A votes 'no' and B votes 'yes'. Note that our modeling of the protocols, that we precise below, is exactly the same as in [33]. We assume that *only* voters A and B are honest while all other entities are dishonest. An arbitrary number of dishonest voters are however subsumed by the attacker and need not be modeled directly.

We now briefly describe the two protocols. The FOO protocol relies on blind signatures and a commitment function. The equational theory is specified in Figure 8. The protocol consists in 3 phases informally described as follows.

$$\begin{aligned}
&\textit{Phase } 1:\\
&\mathsf{V} \to \mathsf{A} : \mathsf{sign}(\mathsf{blind}(\mathsf{commit}(v,r),b),skV))\\
&\mathsf{A} \to \mathsf{V} : \mathsf{sign}(\mathsf{blind}(\mathsf{commit}(v,r),b),skA))\\
&\textit{Phase } 2:\\
&\mathsf{V} \to \mathsf{C} : \mathsf{sign}(\mathsf{commit}(v,r),skA)\\
&\textit{Phase } 3:\\
&\mathsf{V} \to \mathsf{C} : r
\end{aligned}$$

In the first phase the voter V commits to his vote v which he blindly signs and sends to the election administrator A. A checks eligibility of V and then signs the blinded commitment. Blinding the commitment ensures that A cannot trace the ballot. V unblinds the signature and obtains a ballot which is signed by A. In the second phase A submits the signed ballot to a collector C who publishes all the submitted ballots on a public bulletin board. Finally, in the 3rd phase, V submits the random r which allows to open the commitment to C who again publishes this value on the bulletin board. The election can now be tallied by any observer. The detailed model is given in Figure 8. Note that only two honest voters need to be modelled for showing anonymity. All remaining voters and election authorities are subsumed by the adversary. The

processes $A_{yes}B_{no}$ and $A_{no}B_{yes}$ model the situation where these two honest voters have swapped their vote. The protocols do not lead to determinate processes. Therefore, we proved the relation $A_{yes}B_{no} \approx_{ft} A_{no}B_{yes}$.

As the processes are not

Equational Theory:

$$\mathsf{open}(\mathsf{commit}(x,y),y) \to x \qquad \mathsf{check}(\mathsf{sign}(x,y),pk(y)) \to x \qquad \mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x,y),z),y) \to \mathsf{sign}(x,z)$$

Processes:

$$\begin{aligned}
P_1 &= \mathit{Setup}; A_{yes}B_{no}\\
P_2 &= \mathit{Setup}; A_{no}B_{yes}\\
\mathit{Setup} &= \mathbf{out}(c,\mathsf{pk}(sk_A)).\mathbf{out}(c,\mathsf{pk}(sk_B))\\
\mathit{V\text{-}Phase1} &= \mathbf{out}(V,\mathsf{sign}(\mathsf{blind}(\mathsf{commit}(v,r),b),sk)).\\
&\quad\ \mathbf{in}(A,x).[\mathsf{check}(x,\mathsf{pk}(skA)) \stackrel{?}{=} \mathsf{blind}(\mathsf{commit}(v,r),b)]\\
\mathit{V\text{-}Phase2} &= \mathbf{out}(C,\mathsf{unblind}(x,b))\\
\mathit{V\text{-}Phase3} &= \mathbf{out}(C,r)\\
AB &= (\mathit{V\text{-}Phase1}\{^{v_a}/_{v},^{sk_A}/_{sk},^{r_a}/_{r},^{b_a}/_{b},^{x_a}/_{x},\} \parallel \mathit{V\text{-}Phase1}\{^{v_b}/_{v},^{sk_B}/_{sk},^{r_b}/_{r},^{b_b}/_{b},^{x_b}/_{x},\});\\
&\quad\ (\mathit{V\text{-}Phase2}\{^{b_a}/_{b},^{x_a}/_{x}\} \parallel \mathit{V\text{-}Phase2}\{^{b_b}/_{b},^{x_b}/_{x}\});\\
&\quad\ (\mathit{V\text{-}Phase3}\{^{r_a}/_{r}\} \parallel \mathit{V\text{-}Phase3}\{^{r_b}/_{r}\})\\
A_{yes}B_{no} &= AB\{^{yes}/_{v_a},^{no}/_{v_b}\}\\
A_{no}B_{yes} &= AB\{^{no}/_{v_a},^{yes}/_{v_b}\}
\end{aligned}$$

Fig. 8: Formal description of the FOO protocol [39]

We will not give a detailed description of the Okamoto protocol and refer the reader to [33]. The protocol is a variant of the FOO protocol which aims at achieving receipt-freeness. To avoid vote selling a voter should not be able to provide a receipt of how he voted to a potential coercer. In the FOO protocol this is possible by sending all private names to a coercer. The main tool to avoid this problem in the Okamoto protocol is the use of trapdoor commitment functions. These functions allow to change the value of committed vote using a secret value called the trapdoor. Following [24] we model trapdoor commitment by the following equational theory:

$$\begin{aligned}
\mathsf{open}(\mathsf{tdcommit}(x,y,z),y) &\to x & \mathsf{tdcommit}(x,\mathsf{f}(x_1,y,z,x),z) &\to \mathsf{tdcommit}(x_1,y,z)\\
\mathsf{open}(\mathsf{tdcommit}(x,y,z),\mathsf{f}(x,y,z,x_1)) &\to x_1 & \mathsf{f}(x_1,\mathsf{f}(x,y,z,x_1),z,x_2) &\to \mathsf{f}(x,y,z,x_2)
\end{aligned}$$

Intuitively, a trapdoor commitment $\mathsf{tdcommit}(x,y,z)$ commits to x using the key y and trapdoor z. the commitment can be opened using key y to x. However, knowing the trapdoor z one may compute an altermate key $f(x_1,y,z,x)$ which opens the commitment $\mathsf{tdcommit}(x,y,z)$ to x_1 rather than x. This equational theory is out of the scope of most tools, even in the simpler case of a passive adversary. The only result we are aware of that can verify this equational theory is [24]. As for the FOO protocol we used the relation $\approx_{ft}$ to prove anonymity.

To our knowledge, no other tool can handle this automatically. We are aware of two other attempts for verifying the FOO protocol. Using ProVerif [14], Delaune *et al.* [35], verify a transformation of the protocol. However, the soundness of this transformation has never been proven. Chothia *et al.* [22] verify a different notion of anonymity (also based on process equivalence) using the μCRL tool. However, the attacker they consider is only an observer that cannot interact with the protocol participants, yielding a finite state system.

Efficiency. On a standard modern laptop, AKiSs takes a few minutes (e.g. 3 mins for FOO) to carry out the above verification. The use of a multi-core server already reduces these timings by about 40%. We expect that

some optimizations of the saturation procedure and the use of more efficient data structures will diminish these times significantly. Most of the computational effort goes into the saturation of the traces. Interleaving individual roles of a protocol introduces an exponential blowup on the number of traces and saturations to perform. However, it would be straightforward to scale to larger protocols and more sessions by parallelizing the saturation of these traces (e.g. on clusters of machines).

7 Conclusion

In this paper we present a novel procedure for verifying equivalence properties for a bounded number of sessions of cryptographic protocols. The procedure has been implemented in a tool which is able to handle examples which are out of the scope of existing tools.

There are several directions for future work. The implementation of the tool should be optimized and we plan to analyze more examples coming from electronic voting, RFID protocols and auction protocols which all have requirements stated in terms of equivalences.

We would also like to extend the procedure to be able to take disequalities into account. On the one hand, disequalities will allow to verify processes with else branches which are important in a number of practical examples. On the other hand, characterizing disequalities in our decision procedure would allow to directly decide trace equivalence based on static equivalence (rather than static inclusion). Another direction would be to extend the procedure to allow AC operators in order to treat protocols based on exclusive or or Diffie-Hellman exponentiations.

References

1. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, 2006.
2. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th ACM Symposium on Principles of Programming Languages (POPL'01)*. ACM, 2001.
3. M. Abadi and C. Fournet. Private authentication. *Theoretical Computer Science*, 322(3):427–476, 2004.
4. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. Comput.*, 148(1):1–70, 1999.
5. M. Arapinis, T. Chothia, E. Ritter, and M. D. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Comp. Soc. Press, 2010.
6. A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In *Proc. 17th International Conference on Computer Aided Verification (CAV'05)*, LNCS, pages 281–285. Springer, 2005.
7. M. Arnaud, V. Cortier, and S. Delaune. Combining algorithms for deciding knowledge in security protocols. In *Proc. 6th International Symposium on Frontiers of Combining Systems (FroCoS'07)*, volume 4720 of *Lecture Notes in Artificial Intelligence*, pages 103–117. Springer, 2007.
8. F. Baader and W. Snyder. Unification theory. In *Handbook of Automated Reasoning, volume I, chapter 8*, pages 445–532. Elsevier Science, 2001.
9. M. Backes, C. Hritcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proc. 21st IEEE Computer Security Foundations Symposium (CSF'08)*, 2008.
10. M. Baudet. Deciding security of protocols against off-line guessing attacks. In *12th ACM Conference on Computer and Communications Security (CCS'05)*, 2005.
11. M. Baudet, V. Cortier, and S. Delaune. YAPA: A generic tool for computing intruder knowledge. In *Proc. 20th International Conference on Rewriting Techniques and Applications (RTA'09)*, volume 5595 of *LNCS*, pages 148–163. Springer, 2009.
12. S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Symposium on Security and Privacy (S&P'92)*, pages 72–84. IEEE Comp. Soc., 1992.

13. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Comp. Soc. Press, 2001.
14. B. Blanchet. Automatic proof of strong secrecy for security protocols. In *Symposium on Security and Privacy (S&P'04)*, pages 86–100, 2004.
15. B. Blanchet, M. Abadi, and C. Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *Symposium on Logic in Computer Science*, pages 331–340. IEEE Comp. Soc. Press, 2005.
16. J. Borgström. *Equivalences and Calculi for Formal Verifiation of Cryptographic Protocols*. Phd thesis, EPFL, Switzerland, 2008.
17. J. Borgström, S. Briais, and U. Nestmann. Symbolic bisimulation in the spi calculus. In *Proc. 15th Int. Conference on Concurrency Theory*, volume 3170 of *LNCS*, pages 161–176. Springer, 2004.
18. M. Bruso, K. Chatzikokolakis, and J. den Hartog. Analysing unlinkability and anonymity using the applied pi calculus. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Comp. Soc. Press, 2010.
19. V. Cheval, H. Comon-Lundh, and S. Delaune. Automating security analysis: symbolic equivalence of constraint systems. In *Proc. International Joint Conference on Automated Reasoning (IJCAR'10)*, LNAI. Springer, 2010. To appear.
20. V. Cheval, H. Comon-Lundh, and S. Delaune. Trace equivalence decision: Negative tests and non-determinism. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*, Chicago, Illinois, USA, Oct. 2011. ACM Press. To appear.
21. Y. Chevalier and M. Rusinowitch. Decidability of equivalence of symbolic derivations. *Journal of Automated Reasoning*, 2010. To appear.
22. T. Chothia, S. Orzan, J. Pang, and M. Torabi Dashti. A framework for automatically checking anonymity with *mu* crl. In *2nd Symposium on Trustworthy Global Computing (TGC'06)*, volume 4661 of *LNCS*, pages 301–318. Springer, 2007.
23. Ş. Ciobâcă. Computing finite variants for subterm convergent rewrite systems. Research Report LSV-11-06, Laboratoire Spécification et Vérification, ENS Cachan, France, Apr. 2011. 16 pages.
24. Ş. Ciobâcă, S. Delaune, and S. Kremer. Computing knowledge in security protocols under convergent equational theories. In R. Schmidt, editor, *Proc. 22nd International Conference on Automated Deduction (CADE'09)*, LNAI, pages 355–370, Montreal, Canada, Aug. 2009. Springer.
25. Ş. Ciobâcă, S. Delaune, and S. Kremer. Computing knowledge in security protocols under convergent equational theories. *Journal of Automated Reasoning*, 2011. To appear.
26. H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 2005.
27. V. Cortier and S. Delaune. Deciding knowledge in security protocols for monoidal equational theories. In *Proc. 14th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, LNAI. Springer, 2007.
28. V. Cortier and S. Delaune. A method for proving observational equivalence. In *Proc. 22nd IEEE Computer Security Foundations Symposium (CSF'09)*, pages 266–276, Port Jefferson, NY, USA, July 2009. IEEE Computer Society Press.
29. C. J. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Proc. 20th International Conference on Computer Aided Verification (CAV'08)*, volume 5123 of *LNCS*, pages 414–418. Springer, 2008.
30. M. Dahl, S. Delaune, and G. Steel. Formal analysis of privacy for vehicular mix-zones. In *Proc. 15th European Symposium on Research in Computer Security (ESORICS'10)*, volume 6345 of *LNCS*, pages 55–70. Springer, 2010.
31. M. Dahl, S. Delaune, and G. Steel. Formal analysis of privacy for anonymous location based services. In *Proc. Workshop on Theory of Security and Applications (TOSCA'11)*, 2011. To appear.
32. S. Delaune, S. Kremer, and O. Pereira. Simulation based security in the applied pi calculus. In R. Kannan and K. Narayan Kumar, editors, *Proceedings of the 29th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'09)*, volume 4 of *Leibniz International Proceedings in Informatics*, pages 169–180, Kanpur, India, Dec. 2009. Leibniz-Zentrum für Informatik.
33. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 2008. To appear.
34. S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi calculus. *Journal of Computer Security*, 2009. To appear.

35. S. Delaune, M. D. Ryan, and B. Smyth. Automatic verification of privacy properties in the applied pi-calculus. In Y. Karabulut, J. Mitchell, P. Herrmann, and C. D. Jensen, editors, *Proceedings of the 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security (IFIPTM'08)*, volume 263 of *IFIP Conference Proceedings*, pages 263–278, Trondheim, Norway, June 2008. Springer.
36. D. Dolev and A. Yao. On the security of public key protocols. In *Proc. of the 22nd Symp. on Foundations of Computer Science*, pages 350–357. IEEE Comp. Soc. Press, 1981.
37. L. Durante, R. Sisto, and A. Valenzano. Automatic testing equivalence verification of spi calculus specifications. *ACM Transactions on Software Engineering and Methodology*, 12(2):222–284, 2003.
38. S. Escobar, C. Meadows, and J. Meseguer. Maude-npa: Cryptographic protocol analysis modulo equational properties. In *Foundations of Security Analysis and Design V*, volume 5705 of *LNCS*, pages 1–50. Springer, 2009.
39. A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology — AUSCRYPT '92*, volume 718 of *LNCS*, pages 244–251. Springer Verlag, 1992.
40. J. Goubault-Larrecq. Deciding $\mathcal{H}_1$ by resolution. *Information Processing Letters*, 95(3):401–408, Aug. 2005.
41. H. Hüttel. Deciding framed bisimilarity. In *Proc. 4th International Workshop on Verification of Infinite-State Systems (INFINITY'02)*, pages 1–20, 2002.
42. S. Kremer and M. D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *14th European Symposium on Programming (ESOP'05)*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005.
43. J. Liu and H. Lin. A complete symbolic bisimulation for full applied pi calculus. In *Proc. 36th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'10)*, volume 5901 of *LNCS*, pages 552–563. Springer, 2010.
44. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, 1996.
45. P. Narendran, F. Pfenning, and R. Statman. On the unification problem for cartesian closed categories. *J. Symb. Log.*, 62(2):636–647, 1997.
46. T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. 5th Int. Security Protocols Workshop*, volume 1361 of *LNCS*. Springer, 1997.
47. A. Tiu and J. Dawson. Automating open bisimulation checking for the spi-calculus. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*, pages 307–321. IEEE Comp. Soc. Press, 2010.
48. C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *Proc. 16th International Conference on Automated Deduction (CADE'99)*, volume 1632 of *LNCS*, pages 314–328. Springer, 1999.

A Proof of Theorem 2: Soundness and completeness of the set of seed Statements

We prove soundness (see Lemma 3 and Proposition 3) and completeness (see Lemma 4) for the set of seed statements.

Lemma 3 (Soundness of the set of seed statements). *Let T be a ground trace. For any statement f in the set of seed statements* $\mathsf{seed}(T)$ *we have that $T \models f$.*

Proof. We suppose the same naming conventions for T as in the definition of the set of seed statements (see Section 4.1). We prove that for each statement $f \in \mathsf{seed}(T)$ we have that $T \models f$. There are four kinds of seed statements (see Figure 2). We handle them one-by-one.

1. Let m be such that $0 \leq m \leq n$, let σ and τ be substitutions such that $\sigma \in \mathrm{mgu}_{\mathsf{R}}(\{s_k = t_k\}_{k \in T(m)})$ and $\tau \in \mathrm{variants}(l_1\sigma, \ldots, l_m\sigma)$. We show that

$$f = \Big((\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow} \Leftarrow \{\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow)\}_{j \in R(m)}) \Big)$$

   is a statement that is true in T.
   Let ω be an arbitrary substitution grounding for f. Assume furthermore that $T \models (\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow))\omega$ for all $j \in R(m)$. We show that $T \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow})\omega$. In fact we will show a stronger statement. In particular, we to show that

$$T \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_p\sigma\tau\downarrow})\omega$$

   for all $0 \leq p \leq m$. We proceed by induction on p.
   Base case: $p = 0$. We have $(\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_p\sigma\tau\downarrow})\omega = \mathsf{r}.$ and $T \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_p\sigma\tau\downarrow})\omega$ trivially.
   Inductive case: $p > 0$. We assume that $T \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{p-1}\sigma\tau\downarrow})\omega$ and we show that $T \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_p\sigma\tau\downarrow})\omega$ by case analysis on a_p. Before, we do the case analysis, let us first fix some notations.
   Let $T_1 = T$ and $\varphi_1 = \varphi$. As $T \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{p-1}\sigma\tau\downarrow})\omega$, we have that there exist $L_1, \ldots, L_{p-1}$ such that

$$(T_i, \varphi_i) \xrightarrow{L_i} (T_{i+1}, \varphi_{i+1})$$

   and $L_i\varphi_i =_{\mathsf{R}} \ell_i\sigma\tau\downarrow\omega$ for all $1 \leq i < p$, where $T_i = (a_i. \ldots . a_n)\{x_j \mapsto x_j\sigma\tau\downarrow\omega\}_{j \in R(i-1)}$ and where φ_i extends φ_{i-1} (for all $1 < i \leq p$). We can now do the case analysis.
   (a) if $a_p = \mathbf{out}(c_p, t_p)$, then $\ell_p = \mathbf{out}(c_p)$ by definition. Let $T_{p+1} = (a_{p+1}. \ldots . a_n)\{x_j \mapsto x_j\sigma\tau\downarrow\omega\}_{j \in R(p)}$ and let $\varphi_{p+1} = \varphi_p \cup \{w_{dom(\varphi_p)+1} \mapsto t_p\sigma\tau\downarrow\omega\}$. Let $L_p = \mathbf{out}(c_p)$. By the definition, we have that

$$(T_p, \varphi_p) \xrightarrow{L_p} (T_{p+1}, \varphi_{p+1}),$$

   which is was we wanted to prove.
   (b) if $a_p = [s_p \stackrel{?}{=} t_p]$, then $\ell_p = \mathbf{test}$. Let $T_{p+1} = (a_{p+1}. \ldots . a_n)\{x_j \mapsto x_j\sigma\tau\downarrow\omega\}_{j \in R(p)}$ and let $\varphi_{p+1} = \varphi_p$. As $\sigma \in \mathrm{mgu}_{\mathsf{R}}(\{s_k = t_k\}_{k \in T(m)})$, we have that $s_p\sigma =_{\mathsf{R}} t_p\sigma$ and therefore $s_p\sigma\tau\downarrow\omega =_{\mathsf{R}} t_p\sigma\tau\downarrow\omega$. Hence,

$$(T_p, \varphi_p) \xrightarrow{\mathbf{test}} (T_{p+1}, \varphi_{p+1}),$$

   as we wanted to prove.
   (c) If $a_p = \mathbf{in}(c_p, x_p)$, we know that $p \in R(p)$. Let $T_{p+1} = (a_{p+1}. \ldots . a_n)\{x_j \mapsto x_j\sigma\tau\downarrow\omega\}_{j \in R(p)}$ and let $\varphi_{p+1} = \varphi_p$. As $p \in R(p)$, we have that $T \models (\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{p-1}\sigma\tau\downarrow}(X_p, x_p\sigma\tau\downarrow))\omega$ (this is an antecendent of f). Therefore $\varphi_p \vdash^{X_p\omega} x_p\sigma\tau\downarrow\omega$ and, by letting $L_p = \mathbf{in}(c_p, x_p\sigma\tau\downarrow\omega)$, we obtain by the definition of $\rightarrow$ that

$$(T_p, \varphi_p) \xrightarrow{L_p} (T_{p+1}, \varphi_{p+1}),$$

   which is what we wanted to prove.

We have shown that $T \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_p\sigma\tau\downarrow})\omega$.

2. Let $m \in S(n)$ and let $\sigma \in \text{variants}(t_m)$. We show that the statement

$$f = \Big((\mathsf{k}_{\ell_1\sigma\downarrow,\ldots,\ell_m\sigma\downarrow}(w_{|S(m)|}, (t_m\sigma)\downarrow) \Leftarrow \{\mathsf{k}_{\ell_1\sigma\downarrow,\ldots,\ell_{j-1}\sigma\downarrow}(X_j, x_j\sigma\downarrow)\}_{j\in R(m)})\Big)$$

is true in T.
Let ω be a substitution grounding for f. We assume that

$$T \models (\mathsf{k}_{\ell_1\sigma\downarrow,\ldots,\ell_{j-1}\sigma\downarrow}(X_j, x_j\sigma\downarrow))\omega$$

for all $j \in R(m)$ and we show that $T \models (\mathsf{k}_{\ell_1\sigma\downarrow,\ldots,\ell_m\sigma\downarrow}(w_{|S(m)|}, (t_m\sigma)\downarrow))\omega$.
Let $T_i = (a_i.\ldots.a_n)\{x_j \mapsto x_j\sigma\omega\}_{j\in R(i-1)}$ and $\varphi_i = \cup_{1\leq j\leq |S(i-1)|}\{w_j \mapsto t_{o(j)}\sigma\omega\}$, where $o(j) = \min\{x \mid |S(x)| = j\}$, *i.e.* $o(j)$ denotes the index of the jth send action.
We distinguish two cases:

(a) if there exist $L_1, \ldots, L_m$ such that $(T_1, \varphi_1) \xrightarrow{L_1} (T_2, \varphi_2) \xrightarrow{L_2} \ldots \xrightarrow{L_m} (T_{m+1}, \varphi_{m+1})$ such that $L_i\varphi_i =_{\mathsf{R}} l_i\sigma\downarrow\omega$ for all $1 \leq i \leq m$, we have that

$$\varphi_m(w_{|S(m)|}) = t_{o(S(m))}\sigma\omega = t_m\sigma\omega$$

and we have that $\varphi \vdash^{w_{|S(m)|}} t_m\sigma\omega$ and therefore $\varphi \vdash^{w_{|S(m)|}} (t_m\sigma)\downarrow\omega$ which implies that $T \models (\mathsf{k}_{\ell_1\sigma\downarrow,\ldots,\ell_m\sigma\downarrow}(w_{|S(m)|}, t_m\sigma\downarrow))\omega$.

(b) otherwise, we trivially have that $T \models \mathsf{k}_{\ell_1\sigma\downarrow,\ldots,\ell_m\sigma\downarrow}(w_{|S(m)|}, (t_m\sigma)\downarrow)\omega$.

We have shown that $T \models f$.

3. Let c be a public name. We show that

$$f = \Big(\mathsf{k}(c, c) \Leftarrow \Big)$$

is true in T because $\emptyset \vdash^c c$.

4. Let g be a function symbol of arity k and let $\sigma \in \text{variants}(g(x_1, \ldots, x_k))$. We show that the statement

$$f = \Big(\mathsf{k}(g(X_1, \ldots, X_k), g(x_1, \ldots, x_k)\sigma\downarrow) \Leftarrow \{\mathsf{k}(X_j, x_j\sigma\downarrow)\}_{j\in\{1,\ldots,k\}}\Big)$$

is true in T.
Let ω be an arbitrary substitution grounding for f. We assume that $T \models \mathsf{k}(X_j, x_j\sigma\downarrow)\omega$ for all $1 \leq j \leq k$ and we show that

$$T \models (\mathsf{k}(g(X_1, \ldots, X_k), g(x_1, \ldots, x_k)\sigma\downarrow))\omega.$$

We have that

$$\emptyset \vdash^{X_j\omega} x_j\sigma\downarrow\omega$$

for all $1 \leq j \leq k$ by our hypothesis. But this implies

$$\emptyset \vdash^{g(X_1\omega,\ldots,X_k\omega)} g(x_1\sigma\downarrow\omega, \ldots, x_k\sigma\downarrow\omega) =_{\mathsf{R}} g(x_1, \ldots, x_k)\sigma\downarrow\omega$$

which immediately implies that $T \models (\mathsf{k}(g(X_1, \ldots, X_k), g(x_1, \ldots, x_k)\sigma\downarrow))\omega$.
We have shown that $T \models f$.

We have shown for every statement $f \in \mathsf{seed}(T)$ that $T \models f$. □

Proposition 3 (Soundness of $\mathcal{H}(\,)$). *Let T be a ground trace and K be a set of statements such that for all $f \in K$ we have that $T \models f$. Then for all $f \in \mathcal{H}(K)$ we also have that $T \models f$.*

Proof. The proof of this proposition is a straightforward induction on the size of the smallest proof of $f \in \mathcal{H}(K)$.

Base case. The proof of $f \in \mathcal{H}(K)$ is obtained by applying the rule Simple Consequence when $n = 0$. We have that $f' = (H \Leftarrow) \in K$ and $f = f'\sigma$ where σ is a substitution grounding for f'. As $f' \in K$, by hypothesis, $T \models f'$. Hence, as all variables in f' are universally quantified, $T \models f'\sigma$.

Inductive case. We proceed by case distinction on the last rule which has been applied.

- Simple Consequence: We have that $f' = (H \Leftarrow B_1 \dots B_n) \in K$, σ is a substitution grounding for f' such that $f = H\sigma$ and $B_i\sigma \in \mathcal{H}(K)$ for $1 \leq i \leq n$. As $H \Leftarrow B_1 \dots B_n \in K$ we have by hypothesis that $T \models H \Leftarrow B_1 \dots B_n$ and hence $T \models (H \Leftarrow B_1 \dots B_n)\sigma$. By induction hypothesis we also have that $T \models B_i\sigma$. Hence, we conclude that $T \models H\sigma$.
- ExtendK: We have that $\mathsf{k}_u(R, t) \in \mathcal{H}(K)$. By induction hypothesis $T \models \mathsf{k}_u(R, t)$. It follows from the semantics of k that $T \models \mathsf{k}_{uv}(R, t)$.
- ExtendI: Similar to the ExtendK.

Lemma 4. *Let T and S be traces and let φ be a frame. If $(T, \emptyset) \xrightarrow{L_1,\dots,L_n} (S, \varphi)$ then:*

(A) $\mathsf{r}_{L_1\varphi\downarrow,\dots,L_n\varphi\downarrow} \in \mathcal{H}(\mathsf{seed}(T))$
(B) if $\varphi \vdash^R t$ *then* $\mathsf{k}_{L_1\varphi\downarrow,\dots,L_n\varphi\downarrow}(R, t\downarrow) \in \mathcal{H}(\mathsf{seed}(T))$

Proof. We prove the two statements by induction on n. We assume that the two statements hold for any index less than n and we prove them for n.

As $(T, \emptyset) \xrightarrow{L_1,\dots,L_n} (S, \varphi)$, we have that:

- there exists ω such that $(L_1\varphi\downarrow, \dots, L_n\varphi\downarrow) = (\ell_1, \dots, \ell_n)\omega$.
- $s_k\omega =_{\mathsf{R}} t_k\omega$ for all $k \in T(n)$.

We prove each of statements in turn:

(A) As $s_k\omega =_{\mathsf{R}} t_k\omega$ for all $k \in T(n)$, it follows by the definition of $\mathrm{mgu}_{\mathsf{R}}$ that there exists $\sigma \in \mathrm{mgu}_{\mathsf{R}}(\{s_k \stackrel{?}{=} t_k\}_{k\in T(n)})$ such that:
(a) $dom(\sigma) \subseteq X$,
(b) $s_k\sigma =_{\mathsf{R}} t_k\sigma$ for all $k \in T(n)$ and
(c) $\omega[X] =_{\mathsf{R}} (\sigma\pi)[X]$ for some substitution π
where $X = vars(\{s_k, t_k\}_{k\in T(n)})$.
It follows that $(\ell_1, \dots, \ell_n)\omega\downarrow = (\ell_1, \dots, \ell_n)\sigma\pi\downarrow$ for some substitution π.
By the definition of variants$((\ell_1, \dots, \ell_n)\sigma)$, we have that there exists $\tau \in$ variants$((\ell_1, \dots, \ell_n)\sigma)$ such that $(\ell_1, \dots, \ell_n)\sigma\pi\downarrow = (\ell_1, \dots, \ell_n)\sigma\tau\downarrow\tau'$ for some substitution τ'.
By the definition of the seed knowledge base $\mathsf{seed}(T)$, we have that the statement

$$f = \left(\mathsf{r}_{\ell_1\sigma\tau\downarrow,\dots,\ell_n\sigma\tau\downarrow} \Leftarrow \mathsf{k}_{\ell_1\sigma\tau\downarrow,\dots,\ell_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow)_{j\in R(n)}\right) \in \mathsf{seed}(T)$$

is in the seed knowledge base $\mathsf{seed}(T)$.
Let τ'' be the substitution that extends τ' by sending X_j to R_j for all $j \in R(n)$ (where R_j are recipes for $x_j\omega$).
We have by the induction hypothesis that each component of the right hand side of $f\tau''$ is in $\mathcal{H}(\mathsf{seed}(T))$. Therefore the head of $f\tau''$ is in $\mathcal{H}(\mathsf{seed}(T))$:

$$\mathsf{r}_{\ell_1\sigma\tau\downarrow\tau'',\dots,\ell_n\sigma\tau\downarrow\tau''} = \mathsf{r}_{\ell_1\sigma\tau\downarrow\tau',\dots,\ell_n\sigma\tau\downarrow\tau'} \in \mathcal{H}(\mathsf{seed}(T)).$$

(B) By induction on R, we show that:

$$\mathsf{k}_{L_1\varphi\downarrow,\dots,L_n\varphi\downarrow}(R, R\varphi\downarrow) \in \mathcal{H}(\mathsf{seed}(T))$$

(a) If $R = c$ is a public name, and as the statement $f = \left(\mathsf{k}(c,c) \Leftarrow \right)$ is in the set of seed statements by definition, we have that $\mathsf{k}(R, R\varphi\downarrow) = \mathsf{k}(c,c) \in \mathcal{H}(\mathsf{seed}(T))$ by definition and therefore $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, R\varphi\downarrow) \in \mathcal{H}(\mathsf{seed}(T))$ by the ExtendK rule.

(b) If $R = f(R_1, \ldots, R_k)$, let γ be the substitution of domain $dom(\gamma) \subseteq \{y_1, \ldots, y_k\}$ that maps y_j to $R_j\varphi\downarrow$ for all $1 \leq j \leq k$.
Let $\tau \in \text{variants}(f(y_1, \ldots, y_k))$ and τ' be such that $R\varphi\downarrow = (f(y_1, \ldots, y_k)\tau)\downarrow\tau'$.
By the definition of the seed knowledge base, we have that the statement

$$g = \left(\mathsf{k}_{\ell_1,\ldots,\ell_n}(f(Y_1, \ldots, Y_k), f(y_1, \ldots, y_k)\tau\downarrow) \Leftarrow \{\mathsf{k}_{\ell_1,\ldots,\ell_n}(Y_j, y_j\tau\downarrow)\}_{j\in\{1,\ldots,k\}}\right) \in \mathsf{seed}(T).$$

Let $\tau'' = \omega\downarrow \cup \tau' \cup \{Y_j \mapsto R_j\}_{j\in\{1,\ldots,k\}}$. We have that all antecedents of $g\tau''$ are in $\mathcal{H}(\mathsf{seed}(T))$ by the induction hypothesis. Therefore, the head of $g\tau''$ is also in $\mathcal{H}(\mathsf{seed}(T))$.

(c) If $R = w_j$, let m be the smallest index such that $|S(m)| = j$ (i.e. m is the index of the action a_m that output the content of w_j) and let t_m be the term such that $a_m = \mathbf{out}(c, t_m)$ for some channel c.
Let $\tau \in \text{variants}(\ell_1, \ldots, \ell_m, t_m)$ and τ' be substitutions such that $(\ell_1, \ldots, \ell_m, t_m)\omega\downarrow = (\ell_1, \ldots, \ell_m, t_m)\tau\downarrow\tau'$.
We have by the definition of the seed knowledge base that the statement

$$h = \left(\mathsf{k}_{\ell_1\tau\downarrow,\ldots,\ell_m\tau\downarrow}(w_j, t_m\tau\downarrow) \Leftarrow \{\mathsf{k}_{\ell_1\tau\downarrow,\ldots,\ell_{k-1}\tau\downarrow}(X_k, x_k\tau\downarrow)\}_{k\in R(m)}\right) \in \mathsf{seed}(T).$$

Let R_k be recipes of $x_k\tau\downarrow\tau' =_{\mathsf{R}} x_k\omega$ in the smallest possible prefix of φ. Let $\tau'' = \tau' \cup \{X_k \mapsto R_k\}_{k\in R(m)}$. We have that the antecedents of $h\tau''$ are in $\mathcal{H}(\mathsf{seed}(T))$ by the induction hypothesis. Therefore the head of $h\tau''$:

$$\mathsf{k}_{\ell_1\tau\downarrow\tau'',\ldots,\ell_m\tau\downarrow\tau''}(w_j, t_m\tau\downarrow\tau'') = \mathsf{k}_{\ell_1\tau\downarrow\tau',\ldots,\ell_m\tau\downarrow\tau'}(w_j, t_m\tau\downarrow\tau') = \mathsf{k}_{\ell_1\omega\downarrow,\ldots,\ell_m\omega\downarrow}(w_j, t_m\omega\downarrow) \in \mathcal{H}(\mathsf{seed}(T)).$$

But $\ell_1\omega\downarrow, \ldots, \ell_m\omega\downarrow$ is a prefix of w and therefore by the ExtendK rule $\mathsf{k}_w(w_j, t_m\omega\downarrow) = \mathsf{k}_w(R_j, R_j\varphi\downarrow) \in \mathcal{H}(\mathsf{seed}(T))$, which is what we had to prove.

B Soundness and completeness of saturation: Proof of Theorem 3

We start by showing soundness.

B.1 Soundness of saturation

Soundness is an immediate consequence of Lemma 5, Lemma 6, Lemma 8, Lemma 9 and Lemma 10 proved below.

Lemma 5 (Soundness of canonicalization). *If $T \models f$ then $T \models f\Downarrow$.*

Proof. We will show that each canonicalization rule is sound:

1. For the Rename rule, consider a statement

$$f = \left(H \Leftarrow \mathsf{k}_{t_1,\ldots,t_k}(X, x), \mathsf{k}_{t_1,\ldots,t_l}(Y, x), B_1, \ldots, B_n\right)$$

where $k \leq l$ and we show that if $T \models f$ then $T \models g$ where

$$g = \left((H \Leftarrow \mathsf{k}_{t_1,\ldots,t_k}(X, x), B_1, \ldots, B_n)\{Y \mapsto X\}\right)$$

Let τ be a grounding substitution for g such that $T \models \mathsf{k}_{t_1,\ldots,t_k}(X, x)\{Y \mapsto X\}\tau, B_1\{Y \mapsto X\}\tau, \ldots, B_n\{Y \mapsto X\}\tau$. We show that if $T \models f$ then $T \models H\{Y \mapsto X\}\tau$.

Let τ' be a substitution identical to τ, except for $\tau'(Y) = \tau(X)$. We will show that all the antecedents in $f\tau'$ are true in T.
Indeed, $\mathsf{k}_{t_1,\ldots,t_k}(X,x)\tau' = \mathsf{k}_{t_1,\ldots,t_k}(X,x)\{Y \mapsto X\}\tau$ holds by hypothesis. As $k \leq l$ and $T \models \mathsf{k}_{t_1,\ldots,t_k}(X,x)\tau'$, we also have that $T \models \mathsf{k}_{t_1,\ldots,t_k}(X,x)\tau' = \mathsf{k}_{t_1,\ldots,t_l}(Y,x)\tau'$. Furthermore $T \models B_1\tau' = B_1\{Y \mapsto X\}\tau, \ldots, B_n\tau' = B_n\{Y \mapsto X\}\tau$ by hypothesis. As $T \models f$, and all antecedents of $f\tau'$ are true in T, we obtain that $T \models H\tau'$. But $H\tau' = H\{Y \mapsto X\}\tau$ and therefore we have that $T \models H\{Y \mapsto X\}\tau$. As we have chosen τ arbitrarily, it follows that $T \models g$.

2. For the REMOVE rule, consider a solved statement

$$f = \Big(H \Leftarrow \mathsf{k}_{t_1,\ldots,t_k}(X,x), B_1, \ldots, B_n\Big)$$

such that the rule RENAME does not apply to f and such that $x \notin \mathit{vars}(H)$. We show that if $T \models f$ then $T \models g$ where

$$g = \Big(H \Leftarrow B_1, \ldots, B_n\Big)$$

Let τ be an arbitrary substitution such that $T \models B_1\tau, \ldots, B_n\tau$. We will show that $T \models H\tau$ and hence $T \models g$.
Let $(T_1, \varphi_1) = (T, \emptyset)$. We distinguish between two cases:

(a) If

$$(T_1, \varphi_1) \xrightarrow{L_1} (T_2, \varphi_2) \xrightarrow{L_2} \ldots \xrightarrow{L_k} (T_{k+1}, \varphi_{k+1}) \text{ such that } L_i\varphi_i = t_i\tau \text{ for all } 1 \leq i \leq k$$

, we consider the substitution τ' to be identical to τ except for $\tau'(x) = (X\tau)\varphi_{k+1}$.
As $x \notin \mathit{vars}(H)$ and because f is solved and the rule RENAME does not apply, we have that $x \notin \mathit{vars}(B_1, \ldots, B_n)$ and therefore $T \models B_1\tau' = B_1\tau, \ldots, B_n\tau' = B_n\tau$.
Furthermore, we have that $T \models \mathsf{k}_{t_1,\ldots,t_k}(X,x)\tau'$ by the definition of k.
As all antecedents of $f\tau'$ are true in T and $T \models f$, it follows that $T \models H\tau'$. But $H\tau = H\tau'$ since $x \notin \mathit{vars}(H)$ and therefore $T \models H\tau$.

(b) Otherwise, we trivially have that $T \models \mathsf{k}_{t_1,\ldots,t_k}(X,x)\tau$. We have that all antecedents of $f\tau$ are true in T and therefore, as $T \models f$, it follows that $T \models H\tau$.

We have shown that $T \models g$, therefore the rule REMOVE is sound.

We have shown that both rules for computing the canonical form are sound and therefore $T \models f{\Downarrow}$ whenever $T \models f$. □

Lemma 6 (Soundness of the consequence). *If for all $f \in K$ we have that $T \models f$, then for all $f \in$ **conseq**(K) we have that $T \models f$.*

Proof. We show that both inference rules are sound.
For the AXIOM rule, soundness follows immediately from the semantics of k.
For the RES rule, let

$$f = \Big(H \Leftarrow B_1, \ldots, B_n\Big)$$

and

$$g_i = \Big(B_i\sigma \Leftarrow C_1, \ldots, C_m\Big)$$

for $1 \leq i \leq n$ be statements such that $T \models f$ and $T \models g_i$ $(1 \leq i \leq n)$.
We will show that

$$T \models \Big(H\sigma \Leftarrow C_1, \ldots, C_m)$$

by letting τ be a substitution such that $T \models C_1\tau, \ldots, C_m\tau$ and proving that $T \models H\sigma\tau$. Indeed, as $T \models C_1\tau, \ldots, C_m\tau$ and as $T \models g_i$ $(1 \leq i \leq n)$, we have that $T \models B_i\sigma\tau$ $(1 \leq i \leq n)$. But $T \models f$ and therefore $T \models H\sigma\tau$ as well, which is what we had to show.

Lemma 7 (Monotonicity of k). *If $T \models \mathsf{k}_u(R,t)$ then $T \models \mathsf{k}_{uv}(R,t)$.*

Proof. Immediate by the semantics of k.

Lemma 8 (Soundness of the Resolution saturation rule). *Let f, g and h be defined as in the* Resolution *rule. If $T \models f$ and $T \models g$ then $T \models h$.*

Proof. We consider the following statements:

$$\begin{aligned} f &= \Big(H \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_i}(X,t), B_1, \ldots, B_n\Big) \\ g &= \Big(\mathsf{k}_{\ell'_1,\ldots,\ell'_j}(R,t') \Leftarrow B_{n+1}, \ldots, B_m\Big) \\ h &= \Big((H \Leftarrow B_1, \ldots, B_m)\sigma\Big) \end{aligned}$$

with $j \leq i$ and where $\sigma = \mathrm{mgu}(\mathsf{k}_{\ell'_1,\ldots,\ell'_j}(R,t'), \mathsf{k}_{\ell_1,\ldots,\ell_j}(X,t))$. We will show that if $T \models f$ and $T \models g$ then $T \models h$.

Indeed, let τ be an arbitrary substitution grounding for h and assume that $T \models B_1\sigma\tau, \ldots, B_m\sigma\tau$. We will show that $T \models H\sigma\tau$.

As $T \models B_{n+1}\sigma\tau, \ldots, B_m\sigma\tau$ and because $T \models g$, we have that $T \models \mathrm{mgu}(\mathsf{k}_{\ell'_1,\ldots,\ell'_j}(R,t')\sigma\tau$. But $\mathrm{mgu}(\mathsf{k}_{\ell'_1,\ldots,\ell'_j}(R,t')\sigma\tau = \mathsf{k}_{\ell_1,\ldots,\ell_j}(X,t)\sigma\tau$ by choice of $\sigma = \mathrm{mgu}(\mathrm{mgu}(\mathsf{k}_{\ell'_1,\ldots,\ell'_j}(R,t'), \mathsf{k}_{\ell_1,\ldots,\ell_j}(X,t)$.

As $j \leq i$, it follows by Lemma 7 that $T \models \mathsf{k}_{\ell_1,\ldots,\ell_i}(X,t)\sigma\tau$ as well. As all antecedents of $f\sigma\tau$ are true in T and because $T \models f$, we have that $T \models H\sigma\tau$.

As τ was chosen arbitrarily, it follows that $T \models h$. □

Lemma 9 (Soundness of the Equation saturation rule). *Let f, g and h be defined as in the* Equation *rule. If $T \models f$ and $T \models g$ then $T \models h$.*

Proof. We consider the following statements:

$$\begin{aligned} f &= \Big(\mathsf{k}_u(R,t) \Leftarrow B_1, \ldots, B_n\Big) \\ g &= \Big(\mathsf{k}_{u'v'}(R',t') \Leftarrow B_{n+1}, \ldots, B_m\Big) \\ h &= \Big((\mathsf{i}_{u'v'}(R,R') \Leftarrow B_1, \ldots, B_m)\sigma\Big) \end{aligned}$$

where $\sigma = \mathrm{mgu}(\mathsf{k}_u(R,t), \mathsf{k}_{u'}(R,t'))$.

We will show that if $T \models f$ and $T \models g$ then $T \models h$. Let τ be an arbitrary substitution grounding for h. We assume that $T \models B_1\sigma\tau, \ldots, B_m\sigma\tau$ and we show that $T \models \mathsf{i}_{u'v'}(R,R')\sigma\tau$.

As $T \models B_1\sigma\tau, \ldots, B_n\sigma\tau$ and because $T \models f$ we have that $T \models \mathsf{k}_u(R,t)\sigma\tau$. But $\mathsf{k}_u(R,t)\sigma\tau = \mathsf{k}_{u'}(R,t)\sigma\tau$ by choice of $\sigma = \mathrm{mgu}(\mathsf{k}_u(R,t), \mathsf{k}_{u'}(R,t))$ and therefore $T \models \mathsf{k}_{u'}(R,t)\sigma\tau$. By monotonicity of k (Lemma 7) we also have that $T \models \mathsf{k}_{u'v'}(R,t)\sigma\tau$.

As $T \models B_{n+1}\sigma\tau \ldots, B_m\sigma\tau$ and because $T \models g$ we also obtain that $T \models \mathsf{k}_{u'v'}(R',t')\sigma\tau$. As $T \models \mathsf{k}_{u'v'}(R,t)\sigma\tau$ and $T \models \mathsf{k}_{u'v'}(R',t')\sigma\tau$, we have by definition that $T \models \mathsf{i}_{u'v'}(R,R')\sigma\tau$.

We have shown that the head of $h\tau$ is true in T. As τ was chosen arbitrarily, it follows that h holds in T. □

Lemma 10 (Soundness of the Test saturation rule). *Let f, g, h be statements as in the* Test *saturation rule. If $T \models f$ and $T \models g$ then $T \models h$.*

Proof. We consider the following statements:

$$\begin{aligned} f &= \Big(\mathsf{i}_u(R,R') \Leftarrow B_1, \ldots, B_n\Big) \\ g &= \Big(\mathsf{r}_{u'v'} \Leftarrow B_{n+1}, \ldots, B_m\Big) \\ h &= \Big((\mathsf{ri}_{u'v'}(R,R') \Leftarrow B_1, \ldots, B_m)\sigma\Big) \end{aligned}$$

where $\sigma = \mathrm{mgu}(u, u')$).

Let τ be an arbitrary substitution grounding for h. We assume that $T \models B_1\sigma\tau, \ldots, B_m\sigma\tau$ and we show that $T \models \mathsf{ri}_u(R, R')\tau$. Indeed, as $T \models B_1\sigma\tau, \ldots, B_n\sigma\tau$ and as $T \models f$, we have that

$$T \models \mathsf{i}_u(R, R')\sigma\tau. \tag{1}$$

As $T \models B_{n+1}\sigma\tau, \ldots, B_m\sigma\tau$ and as $T \models g$, we have that

$$T \models \mathsf{r}_{u'v'}\sigma\tau. \tag{2}$$

But $\sigma = \mathrm{mgu}(u, u')$ and therefore $u\sigma\tau = u'\sigma\tau$. Therefore, by Equations (1) and (2), we immediately obtain $T \models \mathsf{ri}_{u'v'}(R, R')\sigma\tau$, which is what we wanted. As τ was chosen arbitrarily, it follows that $T \models h$.

Lemma 11 (Soundness of the update). *If for all $f \in K$ we have that $T \models f$ and if $T \models g$, then for any $f \in (K \oplus g)$ we have that $T \models f$.*

Proof. If $K \oplus g = K \cup \{g\Downarrow\}$, we immediately conclude by Lemma 6. Otherwise, it must be that

$$g\Downarrow = \Big(\mathsf{k}_{\ell_1,\ldots,\ell_k}(R, t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n)\Big)$$

for some $R, t, \ell_1, \ldots, \ell_k, i_1, \ldots, i_n, X_1, \ldots, X_n, x_1, \ldots, x_n$ and $K \oplus g = K \cup \{h\}$, where

$$h = \Big(\mathsf{i}_{\ell_1,\ldots,\ell_k}(R, R') \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n)\Big)$$

and where

$$g' = \Big(\mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n)\Big) \in \mathbf{conseq}(K_{\mathsf{solved}}).$$

It is sufficient to show that $T \models h$. As $K_{\mathsf{solved}} \subseteq K$, it immediately follows that $g' \in \mathbf{conseq}(K)$ and, by Lemma 6, $T \models g'$.

We now show that $T \models h$. Let τ be an arbitrary substitution grounding for h such that the antecedents of $h\tau$ are true in T. As the antecedents of $h\tau$ are the same as the antecedents of $g\Downarrow\tau$ and those of $g'\tau$, and as $T \models g$ and $T \models g'$ we have that $T \models \mathsf{k}_{\ell_1,\ldots,\ell_k}(R, t)\tau$ and $T \models \mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t)\tau$.

But this immediately implies that $T \models \mathsf{i}_{\ell_1,\ldots,\ell_k}(R, R')\tau$ (the head of $h\tau$). As τ was chosen arbitrarily, it follows that $T \models h$. □

Proof. Immediate by Lemma 3 to Lemma 11.

B.2 Completeness of saturation

The first two items of the completeness are immediate consequences of Lemma 4 and of Lemma 21 proved below.

The third item follows immediately from the second item, Proposition 5 and Lemma 16, also proved below.

Proposition 4. *Let K be a knowledge base, let $f = \Big(H' \Leftarrow C_1, \ldots, C_m\Big)$ be a statement such that $f \in \mathbf{conseq}(K)$ and let τ be a substitution grounding for f such that $C_i\tau \in \mathcal{H}(K)$ for all $1 \leq i \leq n$. Then $H'\tau \in \mathcal{H}(K)$.*

Proof. Induction on the proof tree of $f \in \mathbf{conseq}(K)$.

If the Axiom rule was used, then we have $H\sigma \in \mathcal{H}(K)$ by the ExtendK rule.

If the Res rule was used, we have that there exists $\Big(H \Leftarrow B_1, \ldots, B_n\Big) \in K$ and a substitution σ such that $H' = H\sigma$ and $B_i\sigma \Leftarrow C_1, \ldots, C_m \in \mathbf{conseq}(K)$ $(1 \leq i \leq n)$.

By the induction hypothesis, we have that $B_i\sigma\tau \in \mathcal{H}(K)$. As $\Big(H \Leftarrow B_1, \ldots, B_n\Big) \in K$, it follows that $H\sigma\tau = H'\tau \in \mathcal{H}(K)$, which is what we had to show.

Definition 17. *Let $\mathcal{S}(H, K)$ be the* size *of the smallest proof tree of $H \in \mathcal{H}(K)$ (defined only when $H \in \mathcal{H}(K)$).*

Proposition 5. *Let K be a knowledge base. If $\mathsf{k}_w(R, t) \in \mathcal{H}_\mathsf{e}(K)$ and $\mathsf{i}_w(R, R') \in \mathcal{H}_\mathsf{e}(K)$, then*

$$\mathsf{k}_w(R', t) \in \mathcal{H}_\mathsf{e}(K).$$

Proof. As $\mathsf{k}_w(R, t) \in \mathcal{H}_\mathsf{e}(K)$, it follows that there exist R'' such that

$$\mathsf{k}_w(R'', t) \in \mathcal{H}(K) \tag{3}$$

and such that $\mathsf{i}_w(R, R'') \in \mathcal{H}_\mathsf{e}(K)$. But $\mathsf{i}_w(R, R') \in \mathcal{H}_\mathsf{e}(K)$ and therefore, by the symmetry and transitivity of $\mathsf{i}_w(\_, \_)$, we have that

$$\mathsf{i}_w(R'', R') \in \mathcal{H}_\mathsf{e}(K). \tag{4}$$

Using Equations 3 and 4 we immediately obtain by the definition of $\mathcal{H}_\mathsf{e}$ that $\mathsf{k}_w(R', T) \in \mathcal{H}_\mathsf{e}(K)$.

Definition 18. *We write $w \sqsubseteq w'$ whenever w is a prefix of w': i.e. there exists $\ell_1, \ldots, \ell_n$ such that $w' = \ell_1, \ldots, \ell_n$ and $w = \ell_1, \ldots, \ell_m$ for some $0 \leq m \leq n$.*

Proposition 6. *If $\mathsf{k}_w(R, t) \in \mathcal{H}(K)$ (resp. $\mathsf{i}_w(R, S) \in \mathcal{H}(K)$) then there exist a statement $f = \Big(\mathsf{k}_{w'}(R', t') \Leftarrow B_1, \ldots, B_m\Big) \in K$ (resp. $f = \Big(\mathsf{i}_{w'}(R', S') \Leftarrow B_1, \ldots, B_m\Big) \in K$) and a substitution σ such that $R'\sigma = R$, $t'\sigma = t$ (resp. $S'\sigma = S$), $w'\sigma \sqsubseteq w$, $B_i\sigma \in \mathcal{H}(K)$ for all $1 \leq i \leq m$ and $\sum_{1 \leq i \leq m} \mathcal{S}(B_i\sigma, K) < \mathcal{S}(\mathsf{k}_w(R, t), K)$.*

Proof. We prove the proposition by induction on the smallest proof tree of $H = \mathsf{k}_w(R, t) \in \mathcal{H}(K)$ (resp. $H = \mathsf{i}_w(R, S) \in \mathcal{H}(K)$). We proceed by case distinction on the last proof rule that has been applied.

- Simple Consequence: In this case we have that there exist a statement $f = \Big(H' \Leftarrow B_1, \ldots, B_m\Big) \in K$ and a substitution σ such that $H'\sigma = H$, $B_i\sigma \in \mathcal{H}(K)$ for all $1 \leq i \leq m$ and $\sum_{1 \leq i \leq m} \mathcal{S}(B_i\sigma, K) + 1 = \mathcal{S}(\mathsf{k}_{w'}(R', t')\sigma, K)$. Hence we directly conclude.
- ExtendK: In this case $H = \mathsf{k}_w(R, t)$ and we have that $w = uv$ for some u, v and $\mathsf{k}_u(R, t) \in \mathcal{H}(K)$. By induction hypothesis, we have that there exists $f = \mathsf{k}_{u'}(R', t') \Leftarrow B_1, \ldots, B_m \in K$ and σ such that $R'\sigma = R$, $t'\sigma = t$, $u'\sigma \sqsubseteq u$, $B_i\sigma \in \mathcal{H}(K)$ for all $1 \leq i \leq m$ and $\sum_{1 \leq i \leq m} \mathcal{S}(B_i\sigma, K) < \mathcal{S}(\mathsf{k}_w(R, t), K)$. As $u \sqsubseteq w$, we also have that $u'\sigma \sqsubseteq w$. Moreover, $\mathcal{S}(\mathsf{k}_w(R, t) \in \mathcal{H}(K)) = \mathcal{S}(\mathsf{k}_u(R, t) \in \mathcal{H}(K)) + 1 > \sum_{1 \leq i \leq m} \mathcal{S}(B_i\sigma, K)$ which allows us to conclude.
- ExtendI: In this case $H = \mathsf{i}_w(R, S)$ and we have that $w = uv$ for some u, v and $\mathsf{i}_u(R, S) \in \mathcal{H}(K)$. By induction hypothesis, we have that there exists $f = \mathsf{i}_{u'}(R', S') \Leftarrow B_1, \ldots, B_m \in K$ and σ such that $R'\sigma = R$, $S'\sigma = S$, $u'\sigma \sqsubseteq u$, $B_i\sigma \in \mathcal{H}(K)$ for all $1 \leq i \leq m$ and $\sum_{1 \leq i \leq m} \mathcal{S}(B_i\sigma, K) < \mathcal{S}(\mathsf{k}_w(R, t), K)$. As $u \sqsubseteq w$, we also have that $u'\sigma \sqsubseteq w$. Moreover, $\mathcal{S}(\mathsf{i}_w(R, S) \in \mathcal{H}(K)) = \mathcal{S}(\mathsf{i}_u(R, S) \in \mathcal{H}(K)) + 1 > \sum_{1 \leq i \leq m} \mathcal{S}(B_i\sigma, K)$ which allows us to conclude.

Lemma 12. *Let K be a saturated knowledge base, let $f \in K$ be a statement*

$$f = \Big(\mathsf{i}_w(R, R') \Leftarrow B_1, \ldots, B_n\Big)$$

and let σ be a substitution grounding for f such that $B_i\sigma \in \mathcal{H}(K_\mathsf{solved})$ for all $1 \leq i \leq n$. Then we have that

$$(\mathsf{i}_w(R, R'))\sigma \in \mathcal{H}(K_\mathsf{solved}).$$

Proof. Let $\mathcal{G} = \sum_{i \in \{1, \ldots, n\}} \mathcal{S}(B_i\sigma, K_\mathsf{solved})$. We prove the lemma by induction on $\mathcal{G}$. If f is a solved statement, the conclusion is immediate by the definition of $\mathcal{H}$.

Otherwise, if f is not a solved statement, there exists some B_j $(1 \leq j \leq n)$ such that $B_j = \mathsf{k}_{w_j}(X_j, t_j)$ and $t_j \notin \mathcal{X}$.

As $B_j\sigma \in \mathcal{H}(K_{\mathsf{solved}})$, it follows by Proposition 6 that $w_j = u_j v_j$ for some u_j, v_j and that there exists

$$g = \Big(\mathsf{k}_{u'_j}(R'_j, t'_j) \Leftarrow B_{n+1}, \ldots, B_m\Big) \in K_{\mathsf{solved}}$$

and a substitution σ' grounding for g such that $B_{n+1}\sigma', \ldots, B_m\sigma' \in \mathcal{H}(K_{\mathsf{solved}})$, $R'_j\sigma' = X_j\sigma$, $t'_j\sigma' = t_j\sigma$, $u'_j\sigma' = u_j\sigma$ and $\mathcal{S}(B_j\sigma) = 1 + \sum_{i\in\{n+1,\ldots,m\}} \mathcal{S}(B_i\sigma')$.

As $\omega = \sigma \cup \sigma'$ is a unifier of $H = \mathsf{k}_{u'_j}(R'_j, t'_j)$ and $\mathsf{k}_{u_j}(X_j, t_j)$, it follows that the two terms are unifiable. Let $\tau = \mathrm{mgu}(H, \mathsf{k}_{u_j}(X_j, t_j))$ denote their most general unifier. As K is saturated, it follows that the Resolution saturation rule was applied to f and g and therefore the resulting equational statement

$$h = \Big(\mathsf{i}_w(R, R') \Leftarrow B_1, \ldots, B_{j-1}, B_{j+1}, \ldots, B_m\Big)\tau \in K$$

must be in K (by the update function, equational statements are added to the knowledge base).

As ω is a unifier of H and B_j and as $\tau = \mathrm{mgu}(H, \mathsf{k}_{u_j}(X_j, t_j))$, it follows that there exists ω' such that $\omega = \tau\omega'$. We have that ω' is a substitution grounding for h, that

$$B_i\tau\omega' \in \mathcal{H}(K_{\mathsf{solved}})$$

for $i \in \{1, \ldots, j-1, j+1, \ldots, m\}$ and that $\sum_{i\in\{1,\ldots,j-1,j+1,\ldots,m\}} \mathcal{S}(B_i\tau\omega') = \mathcal{G} - 1$.

Therefore we can apply the induction hypothesis to h and ω' and conclude.

Lemma 13. *Let K be a saturated knowledge base, let $f \in K$ be a statement*

$$f = \Big(\mathsf{ri}_w(R, R') \Leftarrow B_1, \ldots, B_n\Big)$$

and let σ be a substitution grounding for f such that $B_i\sigma \in \mathcal{H}(K_{\mathsf{solved}})$ for all $1 \leq i \leq n$.

Then we have that

$$(\mathsf{ri}_w(R, R'))\sigma \in \mathcal{H}(K_{\mathsf{solved}}).$$

Proof. Identical to Lemma 12.

Lemma 14. *Let K be a saturated knowledge base, let $f \in K$ be a statement*

$$f = \Big(\mathsf{r}_w \Leftarrow B_1, \ldots, B_n\Big)$$

and let σ be a substitution grounding for f such that $B_i\sigma \in \mathcal{H}(K_{\mathsf{solved}})$ for all $1 \leq i \leq n$.

Then we have that

$$\mathsf{r}_w\sigma \in \mathcal{H}(K_{\mathsf{solved}}).$$

Proof. Identical to Lemma 12 and Lemma 13.

Lemma 15. *Let K be a saturated knowledge base. If $\mathsf{r}_u \in \mathcal{H}(K_{\mathsf{solved}})$ and $\mathsf{i}_u(R, R') \in \mathcal{H}(K_{\mathsf{solved}})$, then $\mathsf{ri}_u(R, R') \in \mathcal{H}(K_{\mathsf{solved}})$.*

Proof. As $\mathsf{r}_u \in \mathcal{H}(K_{\mathsf{solved}})$, there exists a solved statement $f = \Big(\mathsf{r}_v \Leftarrow B_1, \ldots, B_n\Big) \in K_{\mathsf{solved}}$ and a substitution σ grounding for f such that $B_i\sigma \in \mathcal{H}(K_{\mathsf{solved}})$ for all $1 \leq i \leq n$ and such that $u = v\sigma$.

As $\mathsf{i}_u(R, R') \in \mathcal{H}(K_{\mathsf{solved}})$, there exists by Proposition 6 a solved statement $g = \Big(\mathsf{i}_w(T, T') \Leftarrow B_{n+1}, \ldots, B_m\Big)$ and a substitution τ grounding for g such that $B_i\tau \in \mathcal{H}(K_{\mathsf{solved}})$ for all $n+1 \leq i \leq m$ and such that $u \sqsupseteq w\tau$, $R = T\tau$ and $R' = T'\tau$.

As $v\sigma = u \sqsupseteq w\tau$, it follows that $v = v_0 v_1$ such that v_0 and w are unifiable ($\sigma \cup \tau$ is such a unifier). Let $\omega = \mathrm{mgu}(v_0, w)$ and let π be such that $\sigma \cup \tau = \omega \circ \pi$.

As the knowledge base is saturated, the TEST saturation rule must have fired for f and g and therefore K must have been updated by h where

$$h = \Big((\mathsf{ri}_v(T, T') \Leftarrow B_1, \ldots, B_m)\omega\Big).$$

But as h is not a deduction fact, the update must have simply added h to K and therefore $h \in K$.

We have that $B_i\omega\pi = B_i\sigma \in \mathcal{H}(K_{\mathsf{solved}})$ for all $1 \leq i \leq n$ and that $B_i\omega\pi = B_i\tau \in \mathcal{H}(K_{\mathsf{solved}})$ for all $n+1 \leq i \leq m$. By applying Lemma 13 to the statement h and the substitution π, we obtain that $\mathsf{ri}_v(T, T')\omega\pi = \mathsf{ri}_u(R, R') \in \mathcal{H}(K_{\mathsf{solved}})$. □

Lemma 16. *Let K be a saturated knowledge base such that $\mathsf{k}_u(R, t) \in \mathcal{H}(K_{\mathsf{solved}})$ and $\mathsf{k}_{uv}(R', t) \in \mathcal{H}(K_{\mathsf{solved}})$. Then we have that $\mathsf{i}_{uv}(R, R') \in \mathcal{H}(K_{\mathsf{solved}})$.*

Proof. Let $u = \ell_1, \ldots, \ell_k$ and $v = \ell_{k+1}, \ldots, \ell_l$. As $\mathsf{k}_u(R, t) \in \mathcal{H}(K_{\mathsf{solved}})$, it follows by Proposition 6 that there exist

$$f = \Big(\mathsf{k}_w(S, s) \Leftarrow B_1, \ldots, B_n\Big) \in K_{\mathsf{solved}}$$

and a substitution σ grounding for f such that $B_i\sigma \in \mathcal{H}(K_{\mathsf{solved}})$ $(1 \leq i \leq n)$ and $\mathsf{k}_w(S, s)\sigma = \mathsf{k}_{u'}(R, t)$ for some $u' \sqsubseteq u$ a prefix of u.

Similarly, as $\mathsf{k}_{uv}(R', t) \in \mathcal{H}(K_{\mathsf{solved}})$, it follows that there exist

$$f' = \Big(\mathsf{k}_{w'}(S', s') \Leftarrow B'_1, \ldots, B'_m\Big) \in K_{\mathsf{solved}}$$

and a substitution σ' grounding for f' such that $B'_i\sigma \in \mathcal{H}(K_{\mathsf{solved}})$ $(1 \leq i \leq m)$ and $\mathsf{k}_{w'}(S', s')\sigma' = \mathsf{k}_{u''}(R', t)$ for $u'' \sqsubseteq uv$ a prefix of uv.

We have that $w\sigma \sqsubseteq u$, which trivially implies $w\sigma \sqsubseteq uv$. We also have $w'\sigma' \sqsubseteq uv$. Let $w = \ell'_1, \ldots, \ell'_p$ and $w' = \ell''_1, \ldots, \ell''_q$ and let $r = \max\{p, q\}$. We have that $(\ell'_1, \ldots, \ell'_r)\sigma = (\ell''_1, \ldots, \ell''_r)\sigma$.

We have that $\sigma \cup \sigma'$ is a unifier of $\mathsf{k}_{\ell'_1,\ldots,\ell'_r}(\_, s)$ and $\mathsf{k}_{\ell''_1,\ldots,\ell''_r}(\_, s')$, it follows that $\tau = \mathrm{mgu}(\mathsf{k}_{\ell'_1,\ldots,\ell'_r}(\_, s), \mathsf{k}_{\ell''_1,\ldots,\ell''_r}(\_, s'))$ exists. As K is saturated, it follows that the equational fact

$$h = \Big(\mathsf{i}_{\ell'_1,\ldots,\ell'_r}(S, S') \Leftarrow B_1, \ldots, B_n, B'_1, \ldots, B'_m\Big)\tau \in K$$

resulting from applying the EQUATION saturation rule to f and f' is in K.

As $\sigma \cup \sigma'$ is a unifier of $\mathsf{k}_{\ell'_1,\ldots,\ell'_r}(\_, s)$ and $\mathsf{k}_{\ell''_1,\ldots,\ell''_r}(\_, s')$ and as $\tau = \mathrm{mgu}(\mathsf{k}_{\ell'_1,\ldots,\ell'_r}(\_, s), \mathsf{k}_{\ell''_1,\ldots,\ell''_r}(\_, s'))$, it follows that there exists ω such that $\sigma \cup \sigma' = \tau\omega$.

We have that ω is grounding for h and that $B_1\tau\omega, \ldots, B_n\tau\omega, B'_1\tau\omega, \ldots, B'_m\tau\omega \in \mathcal{H}(K_{\mathsf{solved}})$. Therefore, we have by Lemma 12 that

$$\mathsf{i}_{\ell'_1,\ldots,\ell'_r}(S, S')\tau\omega = \mathsf{i}_{\ell'_1\sigma,\ldots,\ell'_r\sigma}(R, R') \in \mathcal{H}(K_{\mathsf{solved}}).$$

But $(\ell'_1, \ldots, \ell'_r)\sigma$ is a prefix of uv and therefore

$$\mathsf{i}_{uv}(R, R') \in \mathcal{H}(K_{\mathsf{solved}})$$

by the EXTENDI rule, which is what we wanted to prove.

Lemma 17. *Let K be a saturated knowledge base, let*

$$f = \Big(\mathsf{k}_w(R, t) \Leftarrow B_1, \ldots, B_n\Big)$$

be a statement such that $f{\Downarrow} \in K_{\mathsf{solved}}$ and let σ be a substitution grounding for f such that $B_i\sigma \in \mathcal{H}(K_{\mathsf{solved}})$ for all $1 \leq i \leq n$. Then we have that

$$(\mathsf{k}_w(R, t))\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

Proof. We prove this by induction on the number of canonicalization steps.

If f is already in canonical form, then the conclusion is immediately true by definition of $\mathcal{H}$.

Otherwise, there must be a canonicalization rule which can be applied to f. We distinguish between two cases:

1. If the RENAME canonicalization rule can be applied, then f must be of the form:

$$f = \Big(\mathsf{k}_w(R,t) \Leftarrow \mathsf{k}_u(X,x), \mathsf{k}_{uv}(Y,x), B_3, \ldots, B_n\Big).$$

Let us consider the statement f' obtained by applying RENAME to f:

$$f' = \Big(\mathsf{k}_w(R,t) \Leftarrow \mathsf{k}_u(X,x), B_3, \ldots B_n\Big)\{Y \mapsto X\}.$$

By the definition of a statement, Y has at most one occurence in $B_1, \ldots, B_n$ and therefore we have that $(B_1, B_3, \ldots B_n)\{Y \mapsto X\} = (B_1, B_3, \ldots, B_n)$. Therefore $(B_1, B_3, \ldots B_n)\{Y \mapsto X\}\sigma = (B_1, B_3, \ldots, B_n)\sigma$. We can therefore apply the induction hypothesis on f' and σ to obtain that

$$\mathsf{k}_w(R,t)\{Y \mapsto X\}\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}). \tag{5}$$

But $\mathsf{k}_u(X,x)\sigma \in \mathcal{H}(K_{\mathsf{solved}})$ and $\mathsf{k}_{uv}(Y,x)\sigma \in \mathcal{H}(K_{\mathsf{solved}})$. By Lemma 16, we have that

$$\mathsf{i}_{uv}(X,Y)\sigma \in \mathcal{H}(K_{\mathsf{solved}}). \tag{6}$$

From Equation 5 and Equation 6 and as uv is a prefix of w by the definition of a statement, we conclude by Proposition 5 that

$$\mathsf{k}_w(R,t)\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

2. If the REMOVE canonicalization rule can be applied, then f must be of the form:

$$f = \Big(\mathsf{k}_w(R,t) \Leftarrow \mathsf{k}_u(X,x), B_2, \ldots, B_n\Big).$$

Let f' be the statement obtained from f by applying REMOVE . We have that

$$f' = \Big(\mathsf{k}_w(R,t) \Leftarrow B_2, \ldots, B_n\Big).$$

By applying the induction hypothesis on f' and σ, we immediately obtain our conclusion:

$$\mathsf{k}_w(R,t)\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

Lemma 18. *Let K be a saturated knowledge base, let*

$$f = \Big(\mathsf{k}_w(R,t) \Leftarrow B_1, \ldots, B_n\Big)$$

be a statement such that $f\Downarrow = \Big(\mathsf{k}_w(R',t) \Leftarrow C_1, \ldots, C_m\Big)$ for some $R', C_1, \ldots, C_m$ and let R'' be a recipe such that

$$g = \Big(\mathsf{k}_w(R'',t) \Leftarrow C_1, \ldots, C_m\Big) \in \mathbf{conseq}(K_{\mathsf{solved}})$$

and such that

$$h = \Big(\mathsf{i}_w(R'',R') \Leftarrow C_1, \ldots, C_m\Big) \in K_{\mathsf{solved}}.$$

Let σ be a substitution grounding for f such that $B_i\sigma \in \mathcal{H}(K_{\mathsf{solved}})$ for all $1 \leq i \leq n$. Then we have that

$$(\mathsf{k}_w(R,t))\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

Proof. We prove the lemma by induction on the number of steps to reach the canonical form.

If f is already in canonical form we have that $B_1, \ldots, B_n = C_1, \ldots, C_m$ and, by applying Proposition 4 to g and σ, we have that

$$\mathsf{k}_w(R', t)\sigma \in \mathcal{H}(K_{\mathsf{solved}}).$$

Furthermore, as $h \in K_{\mathsf{solved}}$ and as all antecedents $B_1\sigma, \ldots, B_n\sigma = C_1\sigma, \ldots, C_m\sigma$ of $h\sigma$ are in $\mathcal{H}(K_{\mathsf{solved}})$, we have that

$$\mathsf{i}_w(R'', R')\sigma \in \mathcal{H}(K_{\mathsf{solved}}).$$

It immediately follows that

$$\mathsf{k}_w(R'', t)\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}),$$

which is what we had the prove.

Otherwise, there must be a canonicalization rule which can be applied to f. We distinguish between two cases:

1. If the RENAME canonicalization rule can be applied, then f must be of the form:

   $$f = \Big(\mathsf{k}_w(R, t) \Leftarrow \mathsf{k}_u(X, x), \mathsf{k}_{uv}(Y, x), B_3, \ldots, B_n\Big).$$

   Let us consider the statement f' obtained by applying RENAME to f:

   $$f' = \Big(\mathsf{k}_w(R, t) \Leftarrow \mathsf{k}_u(X, x), B_3, \ldots B_n\Big)\{Y \mapsto X\}.$$

   By the definition of a statement, Y has at most one occurence in $B_1, \ldots, B_n$ and therefore we have that $(B_1, B_3, \ldots B_n)\{Y \mapsto X\} = (B_1, B_3, \ldots, B_n)$. Therefore $(B_1, B_3, \ldots B_n)\{Y \mapsto X\}\sigma = (B_1, B_3, \ldots, B_n)\sigma$. We can therefore apply the induction hypothesis on f' and σ to obtain that

   $$\mathsf{k}_w(R, t)\{Y \mapsto X\}\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}). \tag{7}$$

   But $\mathsf{k}_u(X, x)\sigma \in \mathcal{H}(K_{\mathsf{solved}})$ and $\mathsf{k}_{uv}(Y, x)\sigma \in \mathcal{H}(K_{\mathsf{solved}})$. By Lemma 16, we have that

   $$\mathsf{i}_{uv}(X, Y)\sigma \in \mathcal{H}(K_{\mathsf{solved}}). \tag{8}$$

   From Equation 7 and Equation 8 and as uv is a prefix of w by the definition of a statement, we conclude by Proposition 5 that

   $$\mathsf{k}_w(R, t)\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

2. If the REMOVE canonicalization rule can be applied, then f must be of the form:

   $$f = \Big(\mathsf{k}_w(R, t) \Leftarrow \mathsf{k}_u(X, x), B_2, \ldots, B_n\Big).$$

   Let f' be the statement obtained from f by applying REMOVE . We have that

   $$f' = \Big(\mathsf{k}_w(R, t) \Leftarrow B_2, \ldots, B_n\Big).$$

   By applying the induction hypothesis on f' and σ, we immediately obtain our conclusion:

   $$\mathsf{k}_w(R, t)\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

Lemma 19. *Let K be a saturated knowledge base, let $f \in K$ be a statement*

$$f = \Big(\mathsf{k}_w(R, t) \Leftarrow B_1, \ldots, B_n\Big)$$

and let σ be a substitution grounding for f such that $B_i\sigma \in \mathcal{H}(K_{\mathsf{solved}})$ for all $1 \leq i \leq n$. Then we have that

$$(\mathsf{k}_w(R, t))\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

Proof. Let $\mathcal{G} = \sum_{i\in\{1,\dots,n\}} \mathcal{S}(B_i\sigma, K_{\mathsf{solved}})$. We prove the lemma by induction on $\mathcal{G}$.

If f is a solved statement, the conclusion is trivial by the definitions of $\mathcal{H}$, $\mathcal{H}_{\mathsf{e}}$.

Otherwise, there exists some $B_j = \mathsf{k}_{w_j}(X_j, t_j)$ (with $1 \leq j \leq n$) such that $t_j \notin \mathcal{X}$.

As $B_j\sigma \in \mathcal{H}(K_{\mathsf{solved}})$, we have by Proposition 6 that there exist

$$g = \Big(\mathsf{k}_{u'}(R', t') \Leftarrow B'_1, \dots, B'_m\Big) \in K_{\mathsf{solved}},$$

a substitution σ' grounding for g such that $B'_1\sigma', \dots, B'_m\sigma' \in \mathcal{H}(K_{\mathsf{solved}})$, $\mathsf{k}_{u'}(R', t')\sigma' = \mathsf{k}_u(X_j, t_j)\sigma$ for some prefix $u \sqsubseteq w_j$ of w_j and $\mathcal{S}(B_j\sigma, K_{\mathsf{solved}}) > \sum_{i\in\{1,\dots,m\}} \mathcal{S}(B'_i\sigma', K_{\mathsf{solved}})$.

As $\sigma \cup \sigma'$ is a unifier of $\mathsf{k}_u(X_j, t_j)$ and $\mathsf{k}_{u'}(R', t')$, it follows that $\tau = \mathrm{mgu}(\mathsf{k}_u(X_j, t_j), \mathsf{k}_{u'}(R', t'))$ exists. Let $\sigma \cup \sigma'$ must be an instance of the most general unifier, let ω be a substitution such that $\sigma \cup \sigma' = \tau\omega$.

As K is saturated, it follows that the RESOLUTION saturation rule was applied to f and g. Let h be the resulting statement:

$$h = \Big(\mathsf{k}_w(R, t) \Leftarrow B_1, \dots, B_{j-1}, B_{j+1}, \dots, B_n, B'_1, \dots, B'_m\Big)\tau.$$

We distinguish two cases:

1. if h is not solved we have that $h \in K$ by the update function (as K is saturated).
   We can therefore apply the induction hypothesis on h and on the substitution ω to immediately conclude.
2. if h is solved, we distinguish two cases:
   (a) either $h\Downarrow \in K$, in which case we conclude by applying Lemma 17 to h and ω.
   (b) or $h\Downarrow = \Big(\mathsf{k}_w(R'', t) \Leftarrow C_1, \dots, C_k\Big)$ and

$$h' = \Big(\mathsf{k}_w(R''', t) \Leftarrow C_1, \dots, C_k\Big) \in \mathbf{conseq}(K_{\mathsf{solved}})$$

   and

$$h'' = \Big(\mathsf{i}_w(R''', R'') \Leftarrow C_1, \dots, C_k\Big) \in K_{\mathsf{solved}}$$

   for some R''', in which case we conclude by applying Lemma 18.

Proposition 7. *If* $\mathsf{k}_u(R, t) \in \mathcal{H}_{\mathsf{e}}(K)$*, then* $\mathsf{k}_{uv} \in \mathcal{H}_{\mathsf{e}}(K)$.

Proof. As $\mathsf{k}_u(R, t) \in \mathcal{H}_{\mathsf{e}}(K)$, it follows that $\mathsf{k}_u(R', t) \in \mathcal{H}(K)$ and $\mathsf{i}_u(R', R) \in \mathcal{H}_{\mathsf{e}}(K)$ for some R'. By the EXTENDK rule, we have that $\mathsf{k}_{uv}(R', t) \in \mathcal{H}(K)$ and by the EXTEND rule, we have that $\mathsf{i}_{uv}(R', R) \in \mathcal{H}_{\mathsf{e}}(K)$. We conclude by rule EQUATIONAL CONSEQUENCE that $\mathsf{k}_{uv}(R, t) \in \mathcal{H}_{\mathsf{e}}(K)$, which is what we had to show.

Lemma 20. *Let K be a saturated knowledge base such that $\mathsf{k}_w(R, t) \in \mathcal{H}(K)$. Then*

$$\mathsf{k}_w(R, t) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

Proof. By induction on $\mathcal{S}(\mathsf{k}_w(R, t), K)$.

We distinguish two cases:

1. either $\mathsf{k}_u(R, t) \in \mathcal{H}(K)$ for some u prefix of w, in which case we obtain by the induction hypothesis that $\mathsf{k}_u(R, t) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. We conclude by Proposition 7.
2. or there exist
$$f = \Big(\mathsf{k}_u(S, s) \Leftarrow B_1, \dots, B_n\Big) \in K$$
(with $B_i = \mathsf{k}_{w_i}(X_i, t_i)$) and a substitution σ grounding for f such that $B_i\sigma \in \mathcal{H}(K)$ for all $1 \leq i \leq k$, $s\sigma = t$, $S\sigma = R$, $u\sigma = w$ and
$$\mathcal{S}(\mathsf{k}_w(R, t), K) = 1 + \sum_{i\in\{1,\dots,n\}} \mathcal{S}(B_i\sigma, K).$$

We apply the induction hypothesis on $B_i\sigma$ for all $1 \leq i \leq n$ and we have that $B_i\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. By the definition of $\mathcal{H}_{\mathsf{e}}$ it follows that there exist $\{R'_i\}_{1\leq i\leq n}$ such that

$$\mathsf{k}_{w_i\sigma}(R'_i, t_i\sigma) \in \mathcal{H}(K_{\mathsf{solved}})$$

and such that

$$\mathsf{i}_{w_i\sigma}(R'_i, X_i\sigma) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}) \text{ for all } 1 \leq i \leq n.$$

As $w_i\sigma$ is a prefix of $u\sigma = w$, we have by rule Extend that

$$\mathsf{i}_{w}(R'_i, X_i\sigma) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}) \text{ for all } 1 \leq i \leq n. \tag{9}$$

Let σ' be a substitution defined to be σ except on $\{X_1, \ldots, X_n\}$, where σ' maps X_i to R'_i for all $1 \leq i \leq n$. We apply Lemma 19 on f and σ'. We obtain that

$$\mathsf{k}_u(S, s)\sigma' = \mathsf{k}_w(S\sigma', t) = \mathsf{k}_w(S\{X_i \mapsto R'_i\}\sigma, t) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}). \tag{10}$$

Therefore, combining Equation 10 with Equation 9 (i.e. replacing R'_i by $X_i\sigma$), we immediately obtain our conclusion:

$$\mathsf{k}_w(S\{X_i \mapsto X_i\sigma\}\sigma, t) = \mathsf{k}_w(S\sigma, t) = \mathsf{k}_w(R, t) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

Lemma 21. *Let K be a set of statements and let K' be the saturation of the knowledge base obtained by updating the empty knowledge base by each statement in K.*

Then $\mathcal{H}(K) \subseteq \mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}})$

Proof. Let $H \in \mathcal{H}(K)$. We will prove by induction on the proof tree of $H \in \mathcal{H}(K)$ that each node of the tree is in $\mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}})$. We distinguish two cases:

1. if $H = \mathsf{k}_w(R, t)$ and $\mathsf{k}_u(R, t) \in \mathcal{H}(K)$ for some prefix u of w, in which case by the induction hypothesis we have that $\mathsf{k}_u(R, t) \in \mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}})$ and we conclude by Proposition 7.
2. if $H = \mathsf{i}_w(R, R')$ and $\mathsf{i}_u(R, R') \in \mathcal{H}(K)$ for some prefix u of w, we have that $\mathsf{i}_u(R, R') \in \mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}})$ by the induction hypothesis and therefore $\mathsf{i}_w(R, R') \in \mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}})$ by rule Extend.
3. otherwise there is a statement
$$f = \left(H' \Leftarrow B'_1, \ldots, B'_n\right) \in K$$
and a substitution σ grounding for f such that $H = H'\sigma$ and $B'_i\sigma \in \mathcal{H}(K)$.
By the induction hypothesis, we have that $B'_i\sigma \in \mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}})$. W.l.o.g. assume that $B'_i = \mathsf{k}_{w'_i}(X_i, t'_i)$. As $B'_i\sigma \in \mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}})$, we have by definition of $\mathcal{H}_{\mathsf{e}}$ that there exist R'_i such that
$$\mathsf{k}_{w'_i\sigma}(R'_i, t_i\sigma) \in \mathcal{H}(K'_{\mathsf{solved}}), \tag{11}$$
$$\mathsf{i}_{w'_i\sigma}(R'_i, X_i\sigma) \in \mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}}) \tag{12}$$
for all $1 \leq i \leq n$.
But $w'_i\sigma$ is a prefix of w, where w is such that $H = \mathit{predicate}_w(\ldots)$ with $\mathit{predicate} \in \{\mathsf{r}, \mathsf{i}, \mathsf{ri}, \mathsf{k}\}$. Therefore, by applying the Extend rule to Equation (12), we obtain
$$\mathsf{i}_{w}(R'_i, X_i\sigma) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}). \tag{13}$$
Let σ' be the substitution defined to be σ except that it maps X_i to R'_i for all $1 \leq i \leq n$.
We will show that $H'\sigma' \in \mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}})$. As K' was updated by f, there are three cases:
   (a) if $f \in K'$, we conclude by Lemma 19, Lemma 12, Lemma 13 or Lemma 14.
   (b) or $f\Downarrow \in K'$ and $f \notin K'$, in which case f must be a solved deduction statement. In this case, by Lemma 17, we obtain that $H'\sigma' \in \mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}})$.

(c) or $f\Downarrow = \Big(\mathsf{k}_w(R,t) \Leftarrow C_1,\ldots,C_m\Big)$ and there exists R' such that

$$\Big(\mathsf{k}_w(R',t) \Leftarrow C_1,\ldots,C_m\Big) \in \mathbf{conseq}(K'_{\mathsf{solved}})$$

and such that

$$\Big(\mathsf{i}_w(R,R') \Leftarrow C_1,\ldots,C_m\Big) \in K'_{\mathsf{solved}}.$$

In this case, we have that $H'\sigma' \in \mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}})$ by Lemma 18.

We have shown that $H'\sigma' \in \mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}})$. But $H'\sigma' = H'\{X_i \mapsto R'_i\}\sigma$ and therefore $H'\{X_i \mapsto R_i\}\sigma \in \mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}})$. By Equation (13), we obtain that $H'\{X_i \mapsto X_i\sigma\}\sigma = H'\sigma \in \mathcal{H}_{\mathsf{e}}(K'_{\mathsf{solved}})$, which is what we had to show.

C Proof of Lemma 1

We let $\Rightarrow$ denote the saturation relation. We let $\Rightarrow^=$ denote the reflexive closure of $\Rightarrow$.

Let $\mathcal{M}_0 \subseteq \mathcal{M}$ be public names and let

$$K_{names} = \{\{\mathsf{k}(m,m) \Leftarrow\}_{m\in\mathcal{M}_0} \cup \{\mathsf{i}(m,m) \Leftarrow\}_{m\in\mathcal{M}_0} \cup \{\mathsf{ri}(m,m) \Leftarrow\}_{m\in\mathcal{M}_0}\}$$

be a set of statements involving names in $\mathcal{M}_0$.

Lemma 22. *Let K be a knowledge base such that $names(K)\cap\mathcal{M}_0 = \emptyset$. Let $K_1 \subseteq K_{names}$. If h is a statement such that $names(h)\cap\mathcal{M}_0 = \emptyset$, then*

$$(K \uplus K_1) \oplus h = (K \oplus h) \uplus K_1.$$

Proof. If h is not solved or if it is not a deduction statement, we have that $(K\uplus K_1)\oplus h = (K\uplus K_1)\cup\{h\} = (K\cup\{h\})\uplus K_1 = (K\oplus h)\uplus K_1$. If h is a solved deduction statement, let

$$h\Downarrow = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R,t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n).$$

We distinguish two cases:

1. either $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \notin \mathbf{conseq}((K\uplus K_1)_{\mathsf{solved}})$ for any R', in which case
$$(K\uplus K_1)\oplus h = (K\uplus K_1)\cup\{h\Downarrow\} = (K\cup\{h\Downarrow\})\uplus K_1.$$
It follows that $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \notin \mathbf{conseq}(K_{\mathsf{solved}})$ for any R' either (since $K \subseteq K\uplus K_1$). Therefore $K\oplus h = K\cup\{h\Downarrow\}$ and we immediately conclude by replacing $K\cup\{h\Downarrow\}$ by $K\oplus h$ in the equation above.
2. or $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \in \mathbf{conseq}((K\uplus K_1)_{\mathsf{solved}})$ for some R'. In this case, $(K\uplus K_1)\oplus h = (K\uplus K_1)\cup\{f\}$ where
$$f = \Big(i_{\ell_1,\ldots,\ell_k}(R,R') \Leftarrow \{\mathsf{k}_{\ell_1,\ldots,\ell_{i_j}}(X_j,x_j)\}_{j\in\{1,\ldots,n\}}\Big)\}.$$
To conclude we show the following claim.

If $names(t)\cap\mathcal{M}_0 = \emptyset$ and

$$\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \in \mathbf{conseq}((K\uplus K_1)_{\mathsf{solved}})$$

then

$$\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \in \mathbf{conseq}(K_{\mathsf{solved}})$$

To proof this claim we proceed by induction on the size of the proof tree of

$$\mathsf{k}_{\ell_1,\ldots,\ell_k}(R',t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1,x_1),\ldots,\mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n,x_n) \in \mathbf{conseq}((K\uplus K_1)_{\mathsf{solved}}).$$

Base case: we need to consider two cases according to which rule has been applied.
- AXIOM: the rule does not depend on the knowledge base and we trivially conclude.
- RES: we have that $n = 0$, i.e., $H \Leftarrow \in (K \cup K_1)_{\mathsf{solved}}$ and $H\sigma = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t)$. As $names(t) \cap \mathcal{M}_0 = \emptyset$ we have that $H \Leftarrow \in K_{\mathsf{solved}}$. Hence, $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t) \in \mathbf{conseq}(K_{\mathsf{solved}})$.

Inductive case: We suppose that the proof ends with an application of the RES rule. We have that $H \Leftarrow B_1, \ldots, B_m \ \in (K \cup K_1)_{\mathsf{solved}}$, $B_i\sigma \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n) \in \mathbf{conseq}((K \uplus K_1)_{\mathsf{solved}})$ and $H\sigma = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t)$. Let $H = \mathsf{k}_u(S, t')$ and $B_i = \mathsf{k}_{u_i}(Y_i, y_i)$. As $H\sigma = \mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t)$ and $names(t) \cap \mathcal{M}_0 = \emptyset$, by inspection of the statements in K_1, it must be that $H \Leftarrow B_1, \ldots, B_m \ \in K_{\mathsf{solved}}$. Moreover, as $t'\sigma = t$ we have by hypothesis that $t'\sigma \cap M_0 = \emptyset$ and hence $t' \cap M_0 = \emptyset$. As $y_i \in vars(t')$ we have that $y_i\sigma \cap M_0 = \emptyset$ and we can apply our induction hypothesis to conclude that $B_i\sigma \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n) \in \mathbf{conseq}(K_{\mathsf{solved}})$ for $1 \leq i \leq n$. Hence, as

$$H \Leftarrow B_1, \ldots, B_m \ \in K_{\mathsf{solved}}$$

and

$$B_i\sigma \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n) \in \mathbf{conseq}(K_{\mathsf{solved}})$$

for $1 \leq i \leq n$ we conclude that $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t) \in \mathbf{conseq}(K)$.

Lemma 23. *If K is a knowledge base such that $names(K) \cap \mathcal{M}_0 = \emptyset$, $K_1 \subseteq K_{names}$ and*

$$K \uplus K_1 \Rightarrow K''$$

then $K'' = K' \uplus K_2$ with $K \Rightarrow^{=} K'$, $K_2 \subseteq K_{names}$ and $names(K') \cap \mathcal{M}_0 = \emptyset$.

Proof. We perform a case distinction depending on which saturation rule triggered:

1. if rule RESOLUTION triggered, we will show that $f, g \in K$.
   Indeed, no statement $\left(\mathsf{k}(m, m) \Leftarrow\right) \in K_1$ can play the role of g in the RESOLUTION saturation rule since $t' = m$ must unify with $t \notin X$. Therefore t must be m, but $m \notin names(K)$ by hypothesis and therefore t cannot be m.
   No statement in K_1 can play the role of f in the RESOLUTION saturation rule since they have no antecedents.
   Therefore $f, g \in K$ and $names(h) \notin \mathcal{M}_0$. We choose $K' = K \oplus h$, $K_2 = K_1$ and we conclude by Lemma 22.
2. if rule EQUATION triggered, we distinguish three cases:
   (a) if a statement $\left(\mathsf{k}(m, m) \Leftarrow\right) \in K_1$ plays the role of f in the EQUATION saturation rule, we have that $t = m$. As t' unifies with m, we have that either $t' = m$ or that t' is a variable. The second case is not possible since g must be well-formed. Therefore $t' = m$. As $m \notin names(K)$ by hypothesis it follows that $g \in K_1$ and therefore $g = \mathsf{k}(m, m)$. Therefore the resulting statement is $\mathsf{i}(m, m)$. We choose $K_2 = K_1 \cup \{\mathsf{i}(m, m)\}$, $K' = K$ to conclude.
   (b) if a statement $\left(\mathsf{k}(m, m) \Leftarrow\right) \in K_1$ plays the role of g, the reasoning is analogous to the case above
   (c) otherwise $f, g \in K$. Therefore $names(h) \cap \mathcal{M}_0 = \emptyset$. We choose $K' = K \oplus h$ and $K_2 = K_1$ to conclude.
3. if rule TEST triggered, we distinguish two cases:
   (a) if $\left(\mathsf{i}(m, m) \Leftarrow\right) \in K_1$ plays the role of f, we choose $K' = K$ and $K_2 = K_1 \cup \{\mathsf{ri}(m, m)\}$ to conclude.
   (b) otherwise $f \in K$. The statement g must also be in K since g is a reachability statement and K_1 does not contain reachability statements. We choose $K' = K \oplus h$ and $K_2 = K_1$ to conclude.

From the above lemma we can immediately conclude that if

$$K \cup \{\mathsf{k}(m, m)\}_{m \in \mathcal{M} \setminus names(K)} \Rightarrow^* K'$$

and K' is saturated, then

$$K \Rightarrow^* K''$$

with K'' saturated and $K' = K'' \cup K_{names}$. This means that there is no need to keep track of all (an infinite number of) names during the saturation process.

D Proof of Lemma 2

The definition of $\mathbf{conseq}(K)$ yields a direct recursive algorithm which moreover computes R:

- (Axiom) Check whether $t = x_j$ for $1 \leq j \leq n$. If this is the case return (yes, X_j).
- (Res) Otherwise, guess a (solved) statement $\mathsf{k}_u(R', t') \Leftarrow \mathsf{k}_{u_1}(Y_1, y_1), \ldots \mathsf{k}_{u_k}(Y_k, y_k) \in K$ and compute substitution σ such that $\mathsf{k}_{\ell_1,\ldots,\ell_k}(R', t) = \mathsf{k}_u(R', t')\sigma$. Check recursively whether

$$\exists R_i.\mathsf{k}_{u_i}(R_i, y_i)\sigma \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n) \in \mathbf{conseq}(K)$$

  for $1 \leq i \leq k$. In that case return (yes, $R'[Y_i \mapsto R_i]_{1 \leq i \leq n}$). Otherwise return no.

Termination is ensured because the size of t when checking whether

$$\exists R.\mathsf{k}_{\ell_1,\ldots,\ell_k}(R, t) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_{i_1}}(X_1, x_1), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_{i_n}}(X_n, x_n) \in \mathbf{conseq}(K)$$

strictly decreases in each recursive call. Indeed, when $\mathsf{k}_u(R', t') \Leftarrow \mathsf{k}_{u_1}(Y_1, y_1), \ldots \mathsf{k}_{u_k}(Y_k, y_k) \in K$ we have that $t' \notin X$ because it is well-formed and $y_i \in vars(t')$ by definition of a statement. Hence, $|y_i\sigma| < |t'\sigma| = |t|$.

E Proof of the algorithm

In order to prove Theorem 4 we need the following technical lemmas.

Lemma 24. *Let T be a trace and let K be a saturated knowledge base associated to T. Then for any statement $f \in K$, we have that:*

1. *if $f = \left(\mathsf{r}_{l_1,\ldots,l_n} \Leftarrow \{\mathsf{k}_{w_i}(X_i, t_i)\}_{i\in\{1,\ldots,m\}}\right)$ and $x \in vars(l_k)$ then there exists $w_j = l_1, \ldots, l_{k'}$ with $k' < k$ such that $x \in vars(t_j)$.*
2. *if $f = \left(\mathsf{k}_{l_1,\ldots,l_n}(R, t) \Leftarrow \{\mathsf{k}_{w_i}(X_i, t_i)\}_{i\in\{1,\ldots,m\}}\right)$ and $x \in vars(t)$ then $x \in vars(t_1, \ldots, t_m)$.*

Proof. The seed knowledge base satisfies the above properties and they are preserved by canonicalization, update and saturation.

Lemma 25. *If $dom(\varphi) \subseteq dom(\varphi')$ and $\varphi \vdash^r t$, then $\varphi' \vdash^r t$.*

Proof. The same rewrite steps to obtain t from $r\varphi$ are used to obtain t from $r\varphi'$.

Lemma 26. *Let T_0 be a trace, $\varphi_0 = \emptyset$ the empty frame, and $\{c_1, \ldots, c_k\}$ names such that $c_i \notin names(T_0)$ for all $1 \leq i \leq k$.*
If

$$(T_0, \varphi_0) \xRightarrow{L_1} (T_1, \varphi_1) \xRightarrow{L_2} \ldots \xRightarrow{L_n} (T_n, \varphi_n)$$

and $\forall 1 \leq i \leq k$

- *either $c_i \notin names(L_1, \ldots, L_n)$*
- *or $\varphi_{idx(c_i)-1} \vdash^{R_i} t_i$ for some t_i where $idx(c_i) = \min\{j \mid c_i \in names(L_j)\}$*

then

$$(T_0, \varphi_0) \xRightarrow{L_1\pi'} (T_1\pi, \varphi_1\pi) \xRightarrow{L_2\pi'} \ldots \xRightarrow{L_n\pi'} (T_n\pi, \varphi_n\pi),$$

where $\pi' = \{c_i \mapsto R_i\}_{i\in\{1,\ldots,k\}}$ and $\pi = \{c_i \mapsto t_i\}_{i\in\{1,\ldots,k\}}$.

Proof. By induction on n, the same operational steps will take place with the new labels.

Lemma 27. *Let T be a trace, let $\{c_1, \ldots, c_k\}$ be public names not appearing in T and let $\pi : \{c_1, \ldots, c_k\} \to$ Messages and $\pi' : \{c_1, \ldots, c_k\} \to$ Recipes be mappings from names to terms. If $T \models \mathsf{k}_{l_1,\ldots,l_i}(R_{i+1}, t_{i+1})$ and $T \models \mathsf{k}_{l_1\pi,\ldots,l_i\pi}(c_i\pi', c_i\pi)$ then $T \models \mathsf{k}_{l_1\pi,\ldots,l_i\pi}(R_{i+1}\pi', t_{i+1}\pi)$.*

Proof. By induction on R.

Lemma 28. *Let T be a trace and φ a frame such that $(T, \varphi) \xRightarrow{L} (T', \varphi')$ and such that*

1. *either $M = L$,*
2. *or $L = \boldsymbol{in}(d, R)$ and $M = \boldsymbol{in}(d, R')$ such that $(R = R')\varphi$.*

Then we have that $(T, \varphi) \xRightarrow{M} (T', \varphi')$.

Proof. If $M = L$ then the result is obvious. Otherwise, R and R' are recipes for the same term in φ and therefore the transition still holds. □

Theorem 4. *Let T be a trace and let P be a determinate process. Let K be the set of solved statements from a saturated knowledge base associated to T. Then $T \sqsubseteq_w P$ iff the following tests hold:*

$$\textsc{Reachability}\ \frac{\begin{array}{c}\left(\mathsf{r}_{l_1,\ldots,l_n} \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{i\in\{1,\ldots,m\}}\right) \in K \\ c_1,\ldots,c_k \textit{ fresh constants} \qquad \sigma : vars(l_1,\ldots,l_n) \to \{c_1,\ldots,c_k\} \textit{ is a bijection} \\ \mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(K) \textit{ for all } i \textit{ such that } l_i = \boldsymbol{in}(t_i) \\ M_i = l_i \textit{ if } l_i \in \{\boldsymbol{test}, \boldsymbol{out}(\_)\} \qquad M_i = \boldsymbol{in}(d_i, R_i) \textit{ if } l_i = \boldsymbol{in}(d_i, t_i)\end{array}}{(P,\emptyset) \xrightarrow{M_1,\ldots,M_n} (T',\varphi)}$$

$$\textsc{Identity}\ \frac{\begin{array}{c}\left(\mathsf{ri}_{l_1,\ldots,l_n}(R,R') \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{i\in\{1,\ldots,m\}}\right) \in K \\ c_1,\ldots,c_k \textit{ fresh constants} \qquad \sigma : vars(l_1,\ldots,l_n) \to \{c_1,\ldots,c_k\} \textit{ is a bijection} \\ \mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(K) \textit{ for all } i \textit{ such that } l_i = \boldsymbol{in}(t_i) \\ M_i = l_i \textit{ if } l_i \in \{\boldsymbol{test}, \boldsymbol{out}(\_)\} \qquad M_i = \boldsymbol{in}(d_i, R_i) \textit{ if } l_i = \boldsymbol{in}(d_i, t_i)\end{array}}{(P,\emptyset) \xrightarrow{M_1,\ldots,M_n} (T',\varphi) \textit{ such that } (R\omega = R'\omega)\varphi \textit{ where } \omega = \{X_i \mapsto x_i\sigma\}}$$

Proof. We first prove that if any of the tests fail, $T \not\sqsubseteq P$. Indeed, if the Reachability test fails, we have that $P \not\xrightarrow{M_1,\ldots,M_n} (\_,\_)$, but, by the soundness of K, we have that $(T,\emptyset) \xrightarrow{M_1,\ldots,M_n} (\_,\_)$ and therefore $T \not\sqsubseteq P$.

If the Identity test fails, we have that:

1. either $(P,\emptyset) \not\xrightarrow{M_1,\ldots,M_n} (\_,\_)$, in which case, by the soundness of K, we have that $(T,\emptyset) \xrightarrow{M_1,\ldots,M_n} (\_,\_)$ and therefore $T \not\sqsubseteq P$.
2. or for any φ' such that $(P,\emptyset) \xrightarrow{M_1,\ldots,M_n} (\_,\varphi')$ we have $(R\pi \neq R'\pi)\varphi'$. By the soundness of K, we have however that $(T,\emptyset) \xrightarrow{M_1,\ldots,M_n} (\_,\varphi)$ and $(R\pi = R'\pi)\varphi$. Therefore $T \not\sqsubseteq P$.

Next, we prove that if $T \not\sqsubseteq P$, then at least one test fails. We assume by contradiction that $T \not\sqsubseteq P$, that all tests pass and we derive a contradiction.

As $T \not\sqsubseteq P$, it follows that there exist $L_1,\ldots,L_n,\varphi$ such that either:

$$\begin{array}{ll}(T,\emptyset) \xrightarrow{L_1,\ldots,L_n} (T',\varphi) & \text{and} \\ \forall S \in P : (S,\emptyset) \not\xrightarrow{L_1,\ldots,L_n} (S',\psi) & \end{array}$$

or:

$$\begin{array}{ll}(T,\emptyset) \xrightarrow{L_1,\ldots,L_n} (T',\varphi) \text{ and } (R = R')\varphi & \text{and} \\ \forall S \in P, (S,\emptyset) \xrightarrow{L_1,\ldots,L_n} (S',\psi) \text{ implies } (R \neq R')\psi & \end{array}$$

Let n be the smallest index such that one of the above holds. We then have that:

$$(T,\emptyset) \xrightarrow{L_1} (T_1,\varphi_1) \xrightarrow{L_2} \ldots \xrightarrow{L_{n-1}} (T_{n-1},\varphi_{n-1}) \xrightarrow{L_n} (T_n,\varphi_n)$$

and there exists $S \in P$ such that:

$$(S,\emptyset) \xRightarrow{L_1} (S_1,\psi_1) \xRightarrow{L_2} \ldots \xRightarrow{L_{n-1}} (S_{n-1},\psi_{n-1}),$$

such that for all R, R' we have $(R = R')\varphi_i \implies (R = R')\psi_i$ $(1 \leq i \leq n-1)$ and:

1. either for all $U \in P$ we have $(U,\emptyset) \not\xRightarrow{L_1,\ldots,L_n} (\_,\psi)$
2. or there exist recipes R, R' such that for any $U' \in P$ such that $(U',\emptyset) \xRightarrow{L_1,\ldots,L_n} (\_,\psi')$ we have $(R \neq R')\psi'$.

We consider each of the cases separately:

1. If we are in the case of Item 1, as $(T,\emptyset) \xrightarrow{L_1,\ldots,L_n} (T_n,\varphi_n)$, we have by Theorem 3 that $\mathsf{r}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow} \in \mathcal{H}_{\mathsf{e}}(K)$. By the definition of $\mathcal{H}_{\mathsf{e}}$, we have that it contains no reachability statements in addition to those in $\mathcal{H}$: therefore $\mathsf{r}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow} \in \mathcal{H}(K)$.
Therefore there exist a statement $f = \left(\mathsf{r}_{l_1,\ldots,l_n} \Leftarrow \mathsf{k}_{w_i}(X_i,x_i)_{i\in\{1,\ldots,m\}}\right) \in K$ and a substitution τ grounding for f such that $l_i\tau = L_i\varphi_n\downarrow$ (for all $1 \leq i \leq n$) and such that $\mathsf{k}_{w_i\tau}(X_i\tau,x_i\tau) \in \mathcal{H}(K)$.
Let $c_1,\ldots,c_k$ be fresh public names and let $\sigma : vars(l_1,\ldots,l_n) \to \{c_1,\ldots,c_k\}$ be a bijection. As $\left(\mathsf{k}_{\ell_1,\ldots,\ell_{|w_i|}}(c_j,c_j) \Leftarrow\right) \in K(T)$ for all $1 \leq i \leq m$ and all $1 \leq j \leq k$ and $w_i\sigma$ is an instance of $\ell_1,\ldots,\ell_{|w_i|}$, it follows by Theorem 3 that there exists a substitution σ' such that $\mathsf{k}_{w_i\sigma}(X_i\sigma',x_i\sigma) \in \mathcal{H}_{\mathsf{e}}(K)$ for all $1 \leq i \leq m$, which implies by the definition of $\mathcal{H}_{\mathsf{e}}$ that there exists a substitution σ'' such that $\mathsf{k}_{w_i\sigma}(X_i\sigma'',x_i\sigma) \in \mathcal{H}(K)$ for all $1 \leq i \leq m$.
By instantianting f with $\sigma \cup (\sigma''[\{X_1,\ldots,X_m\}])$, we obtain that $\mathsf{r}_{l_1\sigma,\ldots,l_n\sigma} \in \mathcal{H}(K)$. By the soundness of K and of $\mathcal{H}$, it follows that $T \models \mathsf{r}_{l_1\sigma,\ldots,l_n\sigma}$. Therefore, there exist recipes R'_i (for all $1 \leq i \leq n$ such that $l_i = \mathbf{in}(\_,t_i)$) such that $T \models \mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R'_i,t_i\sigma)$. By the completeness of K, it follows that there exist recipes R_i such that $\mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i,t_i\sigma) \in \mathcal{H}(K)$.
Let $M_i = l_i$ if $l_i \in \{\mathbf{test},\mathbf{out}(\_)\}$ and let $M_i = \mathbf{in}(d_i,R_i)$ if $l_i = \mathbf{in}(d_i,t_i)$ for all $1 \leq i \leq n$. Because the Reachability test worked, it follows that there exists $S'_0 \in P$ such that, if we let $\psi'_0 = \emptyset$, we have

$$(S'_0,\psi'_0) \xRightarrow{M_1} (S'_1,\psi'_1) \xRightarrow{M_2} \ldots \xRightarrow{M_n} (S'_n,\psi'_n).$$

We already have that $\mathsf{k}_{w_j\tau}(X_j\tau,x_j\tau) \in \mathcal{H}(K)$ by choice of f and of τ. By the soundness of $\mathcal{H}$ and K, we obtain $T \models \mathsf{k}_{w_j\tau}(X_j\tau,x_j\tau)$. By Lemma 24, we have that $|w_j| < i$, and as w_j is a prefix of $l_1,\ldots,l_{i-1}$, we have that $T \models \mathsf{k}_{l_1\tau,\ldots,l_{i-1}\tau}(X_j\tau,x_j\tau)$.
Let $\pi : \{c_1,\ldots,c_k\} \to \mathcal{T}(\mathcal{F},\mathcal{N},\mathcal{M})$ and $\pi' : \{c_1,\ldots,c_k\} \to \mathcal{T}(\mathcal{F},\mathcal{M},\mathcal{W})$ be defined such that $\pi(c_l) = x_j\tau$ and $\pi'(c_l) = X_j\tau$ when $\sigma(x_j) = c_l$. As $X_j\tau = c_l\pi'$, $x_j\tau = c_l\pi$ and $l_1\tau,\ldots,l_{i-1}\tau = l_1\sigma\pi,\ldots,l_{i-1}\sigma\pi$ therefore we have that $T \models \mathsf{k}_{l_1\sigma\pi,\ldots,l_{i-1}\sigma\pi}(c_l\pi',c_l\pi)$.
We also have that $\mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i,t_i\sigma) \in \mathcal{H}(K)$. By the soundness of $\mathcal{H}$ and K, we have that $T \models \mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i,t_i\sigma)$. We apply Lemma 27 to obtain that $T \models \mathsf{k}_{l_1\sigma\pi,\ldots,l_{i-1}\sigma\pi}(R_i\pi',t_i\sigma\pi)$. But $t_i\sigma\pi = t_i\tau$ and $l_1\sigma\pi,\ldots,l_{i-1}\sigma\pi = l_1\tau,\ldots,l_{i-1}\tau$ and therefore we have that $T \models \mathsf{k}_{l_1\tau,\ldots,l_{i-1}\tau}(R_i\pi',t_i\tau)$.
Let R''_i be such that $L_i = \mathbf{in}(d_i,R''_i)$ for some channel d_i for all i such that the ith action of T is a receive. By the definition of $\models$, we have therefore that $T \models \mathsf{k}_{l_1\tau,\ldots,l_{i-1}\tau}(R''_i,t_i\tau)$.
We conclude that $T \models \mathsf{i}_{l_1\tau,\ldots,l_{i-1}\tau}(R_i\pi',R''_i)$, or, equivalently, $(R_i\pi' = R''_i)\varphi_{i-1}$. By the hypothesis, we have that $(R_i\pi' = R''_i)\psi_{i-1}$ as well.
From Lemma 26, we obtain that

$$(S'_0,\psi'_0) = (S'_0\pi,\psi_0\pi) \xRightarrow{M_1\pi'} (S'_1\pi,\psi'_1\pi) \xRightarrow{M_2\pi'} \ldots \xRightarrow{M_n\pi'} (S'_n\pi,\psi'_n\pi).$$

We will show by induction on n that

$$(S'_0\pi,\psi_0\pi) \xRightarrow{L_1} (S'_1\pi,\psi'_1\pi) \xRightarrow{L_2} \ldots \xRightarrow{L_n} (S'_n\pi,\psi'_n\pi).$$

We assume by the induction hypothesis that

$$(S'_0\pi,\psi_0\pi) \xRightarrow{L_1} (S'_1\pi,\psi'_1\pi) \xRightarrow{L_2} \ldots \xRightarrow{L_i} (S'_i\pi,\psi'_i\pi)$$

and we show that

$$(S'_i\pi,\psi'_i\pi) \xRightarrow{L_{i+1}} (S'_{i+1}\pi,\psi'_{i+1}\pi).$$

We will show that L_{i+1} and $M_{i+1}\pi'$ satisfy the conditions of Lemma 28 and then we can easily conclude by it. Indeed, either $L_{i+1} = M_{i+1}\pi'$ (in the case of a **test** or **out** action), or $L_{i+1} = \mathbf{in}(d_{i+1},R''_{i+1})$ and $M_{i+1}\pi' = \mathbf{in}(d_{i+1},R_{i+1}\pi')$ (in the case of a **in** action). In the second case, as we have shown

$(R_{i+1}\pi' = R''_{i+1})\psi_i$, by determinacy of P it follows that $\psi_i\pi \approx_s \psi'_i\pi$ and therefore $(R_{i+1}\pi' = R''_{i+1})\psi'_i$ as well. As the hypothesis of Lemma 28 are satisfied, we can conclude.

Let $U = S'_0$. We have shown that $(U, \emptyset) \xRightarrow{L_1,\dots,L_n} (\_, \_)$, therefore obtaining a contradiction. Therefore Item 1 cannot hold.

2. In the case of Item 2, we assume that for all $U \in P$ such that $(U, \emptyset) \xRightarrow{L_1,\dots,L_n} (\_, \psi)$ we have $(R \neq R')\psi$ to obtain a contradiction.
As $(R = R')\varphi_n$, it follows by Theorem 3 that $\mathsf{i}_{L_1\varphi_n\downarrow,\dots,L_n\varphi_n\downarrow}(R, R') \in \mathcal{H}_{\mathsf{e}}(K)$.
As $(R \neq R')\psi$ it follows that there exist recipes Q, Q' such that $\mathsf{i}_{L_1\varphi_n\downarrow,\dots,L_n\varphi_n\downarrow}(Q, Q') \in \mathcal{H}(K)$ but $(Q \neq Q')\psi$.
As $\mathsf{r}_{L_1\varphi_n\downarrow,\dots,L_n\varphi_n\downarrow} \in \mathcal{H}(K)$, we have by Lemma 15 that $\mathsf{ri}_{L_1\varphi\downarrow,\dots,L_n\varphi\downarrow}(Q, Q') \in \mathcal{H}(K)$. Therefore there exists a statement

$$f = \left(\mathsf{ri}_{l_1,\dots,l_n}(P, P') \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{i\in\{1,\dots,m\}}\right) \in K$$

and a substitution τ grounding for f such that $\mathsf{k}_{w_i\tau}(X_i\tau, x_i\tau) \in \mathcal{H}(K)$ (for all $1 \leq i \leq m$), $l_1\tau, \dots, l_n\tau = L_1\varphi_n\downarrow, \dots, L_n\varphi_n\downarrow$, $P\tau = Q$ and $P'\tau = Q'$.
Let $c_1, \dots, c_k$ be fresh public names and let $\sigma : vars(l_1, \dots, l_n) \to \{c_1, \dots, c_k\}$ be a bijection. As $\left(\mathsf{k}_{\ell_1,\dots,\ell_{|w_i|}}(c_j, c_j) \Leftarrow\right) \in K(T)$ for all $1 \leq i \leq m$ and all $1 \leq j \leq k$ and $w_i\sigma$ is an instance of $\ell_1, \dots, \ell_{|w_i|}$, it follows by Theorem 3 that there exists a substitution σ' such that $\mathsf{k}_{w_i\sigma}(X_i\sigma', x_i\sigma) \in \mathcal{H}_{\mathsf{e}}(K)$ for all $1 \leq i \leq m$, which implies by the definition of $\mathcal{H}_{\mathsf{e}}$ that there exists a substitution σ'' such that $\mathsf{k}_{w_i\sigma}(X_i\sigma'', x_i\sigma) \in \mathcal{H}(K)$ for all $1 \leq i \leq m$.
By instantianting f with $\sigma \cup (\sigma''[\{X_1, \dots, X_m\}])$, we obtain that $\mathsf{r}_{l_1\sigma,\dots,l_n\sigma} \in \mathcal{H}(K)$. By the soundness of K and of $\mathcal{H}$, it follows that $T \models \mathsf{r}_{l_1\sigma,\dots,l_n\sigma}$. Therefore, there exist recipes R'_i (for all $1 \leq i \leq n$ such that $l_i = \mathbf{in}(\_, t_i)$) such that $T \models \mathsf{k}_{l_1\sigma,\dots,l_{i-1}\sigma}(R'_i, t_i\sigma)$. By the completeness of K, it follows that there exist recipes R_i such that $\mathsf{k}_{l_1\sigma,\dots,l_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(K)$.
Let $M_i = l_i$ if $l_i \in \{\mathbf{test}, \mathbf{out}(\_)\}$ and let $M_i = \mathbf{in}(d_i, R_i)$ if $l_i = \mathbf{in}(d_i, t_i)$ for all $1 \leq i \leq n$. Because the Reachability test worked, it follows that there exists $S'_0 \in P$ such that, if we let $\psi'_0 = \emptyset$, we have

$$(S'_0, \psi'_0) \xRightarrow{M_1} (S'_1, \psi'_1) \xRightarrow{M_2} \dots \xRightarrow{M_n} (S'_n, \psi'_n).$$

Similarly to Item 1, we can show that there exists $S'_0 \in P$ such that

$$(S'_0, \psi'_0) = (S'_0\pi, \psi_0\pi) \xRightarrow{L_1} (S'_1\pi, \psi'_1\pi) \xRightarrow{L_2} \dots \xRightarrow{L_n} (S'_n\pi, \psi'_n\pi)$$

where π and π' are defined as in Item 1.
Furthermore, as the Identity test worked, it follows that $(P\omega = P'\omega)\psi'_n$, where $\omega = \{X_i \mapsto x_i\sigma\}$. As the equational theory is stable by substitution of terms for names, we have that $(P\omega\pi' = P'\omega\pi')\psi'_n\pi$. But $P\omega\pi' = P\tau = Q$ and $P'\omega\pi' = P'\tau = Q'$, which means that $(Q = Q')\psi'_n\pi$.
We have shown that there exists $S'_0 \in P$ such that $(S'_0, \emptyset) \xRightarrow{L_1,\dots,L_n} (\_, \psi'_n\pi)$ and $(Q = Q')\psi'_n\pi$. By determinacy of P, it follows that $(Q = Q')\psi$ for any ψ and any $U \in P$ such that $(U, \emptyset) \xRightarrow{L_1,\dots,L_n} (\_, \psi)$, therefore we obtain a contradiction.

As both cases yield a contradiction, it follows that if $T \not\sqsubseteq P$, there exists at least one test that fails.