Symbolic verification of privacy-type properties for security protocols with XOR (extended version)

David Baelde, Stéphanie Delaune, Ivan Gazeau, Steve Kremer

To cite this version:

HAL Id: hal-01533694
https://inria.hal.science/hal-01533694
Submitted on 6 Jun 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Symbolic verification of privacy-type properties for security protocols with XOR
(extended version)

David Baelde*, Stéphanie Delaune†, Ivan Gazeau‡, and Steve Kremer‡
*LSV, ENS Cachan & CNRS & INRIA Saclay Île-de-France, France
†CNRS & IRISA, Rennes, France
‡LORIA, INRIA Nancy - Grand-Est, France

Abstract—In symbolic verification of security protocols, process equivalences have recently been used extensively to model strong secrecy, anonymity and unlinkability properties. However, tool support for automated analysis of equivalence properties is limited compared to trace properties, e.g., modeling authentication and weak notions of secrecy. In this paper, we present a novel procedure for verifying equivalences on finite processes, i.e., without replication, for protocols that rely on various cryptographic primitives including exclusive or (xor). We have implemented our procedure in the tool AKISS, and successfully used it on several case studies that are outside the scope of existing tools, e.g., unlinkability on various RFID protocols, and resistance against guessing attacks on protocols that use xor.

I. INTRODUCTION

Protecting authenticity and confidentiality of transactions by the use of cryptography has become standard practice. Security protocols such as TLS, SSH, or Kerberos are nowadays widely deployed. However, history has shown that designing secure protocols is challenging, because of the concurrent execution of protocols in an adversarial environment. Many attacks exploit flaws in the protocol logic rather than, or sometimes combined with, weaknesses in the underlying cryptographic primitives, e.g., [13]. During the past two decades, several efficient automated verification tools have been developed to detect logical flaws, e.g., ProVerif [16], AVISPA [6], Maude-NPA [28], Tamarin [38], and they have successfully discovered many attacks in academic protocols [34], standards [10] and deployed protocols [5].

Authentication and (weak forms of) confidentiality are modelled as trace properties: they are checked by verifying that each possible trace of the system satisfies some predicate. The verification of such properties is nowadays well understood and enjoys efficient tool support, as discussed above. However, some properties such as resistance against guessing attacks, strong secrecy, anonymity and unlinkability [24], [27], [4], [17] are expressed in terms of indistinguishability, a form of process equivalence [2], [1]. Automated verification of equivalence properties is not yet as mature as for trace properties. Several tools have recently been extended with the possibility to verify a strong equivalence called diff-equivalence [15], [11], [57]. This equivalence can only relate processes that only differ in the messages that they use, but not in their control flow, which is too strong for some applications. Dedicated tools for verifying process equivalences on security protocols, in the case of a bounded number of sessions, have also been developed. The SPEC tool [39] decides a symbolic bisimulation, which implies trace equivalence, for protocols that use a fixed set of standard cryptographic primitives. It does not support protocols with else branches. The APTE tool [19] decides trace equivalence for processes including else branches, and a fixed set of standard primitives. The AKISS tool [18] is able to check trace equivalence for processes without else branches but supports various cryptographic primitives, including most standard ones as well as, e.g., blind signatures and trapdoor bit commitment.

Some protocols use cryptographic primitives that have algebraic properties [25]. Exclusive or (xor) is such a primitive, and protocols implemented on low-power devices, such as RFID tags, often rely on it because of its computational efficiency [41]. There exist many results for taking into account algebraic properties for trace based properties, in particular for the xor operator [23], [21], [33]. However, only a few procedures for equivalence properties take algebraic properties into account. Delaune et al. [26] have studied equivalence of constraint systems, showing in particular that the theory for xor is decidable in PTIME. However, they only consider the case of pure group theories, not allowing any other equations, e.g., those modelling encryption. When considering an unbounded number of sessions, tools that can verify diff-equivalence do not effectively support xor. The Tamarin tool supports a theory for Diffie-Hellman exponentiations, but not xor. The Maude-NPA tool supports xor in principle, but when verifying equivalence properties it does not terminate even on simple examples. For trace properties such as secrecy and authentication, termination is achieved in practice on several examples, e.g., a xor based variant of the NSL protocol, the...
IBM CCA crypto API: sometimes, attacks are discovered by bounding the search depth to avoid non-termination; the analysis of CCA also relies on Never Patterns, a technique that allows to prune the search space, but which requires an external, protocol-dependent \(^{[22]}\) justification.

**AKISS in a nutshell:** The procedure that we present in this paper builds on previous work by Chadha et al. \(^{[18]}\), and its implementation in the AKISS tool. This tool checks equivalences for protocols modelled as processes in a calculus similar to the applied pi-calculus \(^{[1]}\), but without else branches nor replication. It actually checks for two equivalences which over- and under-approximate the standard notion of trace equivalence for cryptographic protocols, allowing one to either prove or disprove trace equivalence. The coarser equivalence also coincides with trace equivalence on the large class of *determinate* processes.

The AKISS tool supports cryptographic primitives that can be expressed through a convergent rewrite system enjoying the finite variant property \(^{[22]}\). Termination is guaranteed for the subclass of subterm convergent rewrite theories, but also achieved in practice on several examples outside this class.

The procedure is based on a fully abstract modelling of symbolic traces of the protocols into first-order Horn clauses. Each symbolic trace is translated into a set of clauses called *seed statements*, and a dedicated *resolution procedure* is applied to this set to construct a set of statements that have a simple form, called *solved statements*. It is shown that these solved statements form a sound and complete representation of the symbolic trace under study. Therefore, only the set of solved statements is required to decide whether a given symbolic trace is included in some process. To decide trace equivalence between processes \(P\) and \(Q\), the procedure checks whether each symbolic trace of \(P\) is included in \(Q\) as described above, and vice versa.

**Contributions:** We design a new procedure for verifying trace equivalence of protocols that use the xor operator, extending the work by Chadha et al. \(^{[18]}\). Our procedure follows the general structure of the original one, modelling protocols as processes in (a variant of the) applied pi-calculus (Section \([II]\)) and translating symbolic traces into Horn clauses (Section \([III]\)). The xor operator is not supported in AKISS because it cannot be modelled by a convergent rewrite system. Our approach consists in first orienting the equations of xor into a convergent rewrite system *modulo associativity and commutativity* (AC), then generalizing the procedure of Chadha et al. to reason modulo AC (Section \([IV]\)). A direct generalisation would be sound and complete, but would not terminate even on very simple examples. We therefore completely redesign the resolution procedure, using a new strategy to forbid certain steps that would yield non-termination. Showing that these steps are indeed unnecessary requires essential changes to the completeness proof. The modified procedure yields an effective algorithm for checking trace equivalence (Section \([IV]\)). Although termination is not guaranteed, we have implemented our procedure as an extension of AKISS and demonstrated its effectiveness on several examples, including unlinkability for various RFID protocols \(^{[41]}\) and resistance against guessing attacks for password based protocols \(^{[30]}\), \(^{[31]}\). To the best of our knowledge, our tool is the first that can effectively verify equivalence properties for protocols that use xor. As with all of the above-mentioned tools, our results are restricted to the (symbolic) model under consideration: when our algorithm concludes that there is no attack, it is only meaningful in that model. A proof in the computational model would be stronger, but would still bear some limitations. A natural question is whether our symbolic model could be computationally sound. Note that the impossibility result of \(^{[40]}\) does not apply here, since we only consider bounded runs. Nevertheless, we do not claim computational soundness, and argue instead that symbolic security proofs are already useful information.

**Outline:** In Section \([I]\) we introduce our formalism for modelling protocols, a variant of the replication-free fragment of the applied pi calculus. In Section \([III]\) we provide a fully abstract representation in Horn clauses of protocols expressed in our calculus. Next, we present our saturation procedure based on Horn clause resolution in Section \([IV]\). We present the algorithm for checking trace equivalence, its implementation, and our case studies in Section \([V]\).

## II. Process Calculus

We introduce the process calculus and the notion of equivalence that we use to model protocols and indistinguishability. Our calculus has similarities with the applied pi-calculus \(^{[1]}\) which has been extensively used to specify security protocols. Participants in a protocol are modeled as processes, and the communication between them is modeled by means of message passing.

### A. Term algebra

As usual in symbolic models we model messages as terms. We consider several sets of atomic terms:

- \(N\) is a set of *names*, partitioned into the disjoint sets \(N_{\text{prv}}\) and \(N_{\text{pub}}\) of *private* and *public names*;
- \(X\) is the set of *message variables*, denoted \(x, y, \text{ etc.}\);
- \(W = \{w_1, w_2, \ldots\}\) is the set of *parameters*.

Intuitively, private names in \(N_{\text{prv}}\) represent nonces or keys generated by honest participants, while public names in \(N_{\text{pub}}\) represent identifiers available both to the attacker and to honest participants, and attacker nonces. Parameters are used by the attacker as pointers to refer to messages that were previously output by the protocol participants.

Given a signature \(\Sigma\) (i.e., a finite set of function symbols together with their arity) and a set of atoms \(\mathcal{A}\) we denote by \(T(\Sigma,\mathcal{A})\) the set of *terms*, defined as the smallest set that contains \(\mathcal{A}\) and that is closed under application of function symbols in \(\Sigma\). We denote by \(\text{vars}(t)\) the set of *variables* occurring in a term \(t\). As usual, a substitution is a function from variables to terms, that is lifted to terms homomorphically. The application of a substitution \(\sigma\) to a term \(u\) is written \(u\sigma\), and we denote \(\text{dom}(\sigma)\) its *domain*, i.e. \(\text{dom}(\sigma) = \{x \mid \sigma(x) \neq x\}\).
The identity substitution, of empty domain, is noted \( \emptyset \). The positions of a term are defined as usual.

We associate an equational theory \( E \) to the signature \( \Sigma \). It consists of a finite set of equations of the form \( M = N \) where \( M, N \in T(\Sigma, \lambda) \), and induces an equivalence relation over terms: \( \sim_E \) is the smallest congruence on terms, which contains all equations \( M = N \) in \( E \), and that is closed under substitution of terms for variables. To model protocols that only rely on the xor operator, we consider \( \Sigma_{\text{xor}} = \{ \oplus, 0 \} \), and the equational theory \( E_{\text{xor}} \) below:

\[
\begin{align*}
x \oplus x &= 0 \\
x \oplus (y \oplus z) &= (x \oplus y) \oplus z \\
x \oplus 0 &= x \\
x \oplus y &= y \oplus x
\end{align*}
\]

We denote by \( AC \) the equational theory defined by the two equations on the right. We may also want to consider additional primitives, e.g., pairs, symmetric and asymmetric encryptions, signatures, hashes, etc. This can be done by extending the signature as well as the equational theory.

**Example 1:** Let \( \Sigma_{\text{xor}}^+ = \Sigma_{\text{xor}} \cup \{ (\cdot, \cdot), \text{proj}_1, \text{proj}_2, \text{id} \} \), and consider the equational theory \( E_{\text{xor}}^+ \) extending \( E_{\text{xor}} \) with the equations \( \text{proj}_1((x, y)) = x \) and \( \text{proj}_2((x, y)) = y \). The symbol \( (\cdot, \cdot) \) models pairs; the \( \text{proj}_i \) symbols model projections; the unary symbol \( \text{id} \) models a hash function. Take \( id \in N_{\text{pub}} \) to model the identity of a participant, and \( r_1, r_2, k \in N_{\text{priv}} \) to represent two random numbers and a key, a priori unknown to the attacker. Let \( t_0 = (id \oplus r_2, h((r_1, k)) \oplus r_2) \). We have that \( (\text{proj}_1(t_0) \oplus id) \oplus \text{proj}_2(t_0) = \text{proj}_1(t_0) \oplus \text{proj}_2(t_0) = E_{\text{xor}}(h((r_1, k))) \).

In this paper we consider a signature \( \Sigma \) such that \( \Sigma_{\text{xor}} \subseteq \Sigma \), together with an equational theory generated by a set of equations of the form

\[ E = E_{\text{xor}} \cup \{ M = N \mid M, N \in T(\Sigma \setminus \Sigma_{\text{xor}}, \lambda) \} \]

Hence, \( E \) models xor in combination with any other equational theory that is disjoint from \( E_{\text{xor}} \).

**B. Finite variant property**

A rewrite system \( R \) is a set of rewrite rules of the form \( \ell \to r \) where \( \ell, r \in T(\Sigma, \lambda) \), and \( \text{vars}(r) \subseteq \text{vars}(\ell) \). A term \( t \) can be rewritten in one step (modulo AC) to \( u \), denoted \( t \to_{\text{R}, AC} u \), if there exists a position \( p \) in \( t \), a rule \( \ell \to r \) in \( R \) and a substitution \( \sigma \) such that \( tl_p =_{\text{AC}} \ell \sigma \) and \( u = tr \sigma \). i.e., the term at position \( p \) in \( t \) is equal to \( \ell \sigma \) modulo AC and \( u \) is the term obtained by replacing, in \( t \), the subterm \( tl_p \) with \( r \sigma \). The relation \( \to_{\text{R}, AC} \) denotes the transitive and reflexive closure of \( \to_{\text{R}, AC} \).

A rewrite system \( R \) is AC-convergent if the relation \( \to_{\text{R}, AC} \) is confluent and strongly terminating. We denote by \( tl_{\text{R}, AC} \) (or simply \( tl \)) the normal form of a term \( t \). In the following we only consider equational theories \( E \) that can be represented by a rewrite system \( R \) which is AC-convergent, i.e., such that

\[ u =_E v \iff u \downarrow_{\text{R}, AC} =_{\text{AC}} v \downarrow_{\text{R}, AC} \]

**Example 2:** The equational theory \( E_{\text{xor}}^+ \) of Example 1 can be represented by the following AC-convergent system:

\[
\begin{align*}
x \oplus (x \oplus y) &= y \\
x \oplus 0 &= x \\
x \oplus x &= 0
\end{align*}
\]

\[ \text{proj}_1((x, y)) \to x \]

\[ \text{proj}_2((x, y)) \to y \]

Let \( t_0 = (id \oplus r_2, h((r_1, k)) \oplus r_2) \). We have that:

\[
\begin{align*}
\text{proj}_1(t_0) \&= \text{proj}_2(t_0) \\
\to_{\text{R}_{\text{xor}}^+, AC} (id \oplus r_2) \&= (id \oplus r_2) \oplus (h((r_1, k)) \oplus r_2) \\
\to_{\text{R}_{\text{xor}}^+, AC} \&= id \oplus h((r_1, k))
\end{align*}
\]

Note that the first rule in \( R_{\text{xor}}^+ \) is essential in last rewriting step above, and more generally for \( R_{\text{xor}}^+ \) to represent \( E_{\text{xor}} \).

Given an AC-convergent rewrite system \( R \), we define complete sets of variants, first introduced in [22].

**Definition 1:** Consider a rewrite system \( R \) that is AC-convergent, and a set of terms \( T \). A set of substitutions \( \text{vars}_{R, AC}(T) \) is called a complete set of variants for the set of terms \( T \), if for any substitution \( \omega \) there exist \( \sigma \in \text{vars}_{R, AC}(T) \), and a substitution \( \tau \) such that:

\[
\begin{align*}
x\omega &\downarrow =_{\text{AC}} x\sigma \downarrow \tau \text{ for any } x \in \text{vars}(T), \\
(t\omega) &\downarrow =_{\text{AC}} (t\sigma) \downarrow \tau \text{ for any } t \in T.
\end{align*}
\]

The set of variants of \( t \) represents a pre-computation such that the normal form of any instance of \( t \) is equal (modulo AC) to an instance of \( t\sigma \downarrow \) for some \( \sigma \) in the set of variants, without the need to apply further rewrite steps. A rewrite system has the finite variant property if for any set of terms one can compute a finite complete set of variants. We will often write \( \text{vars}_{R, AC}(t_1, \ldots, t_n) \) instead of \( \text{vars}_{R, AC}(t_1, \ldots, t_n) \).

**Example 3:** Considering the equational theory \( E_{\text{xor}}^+ \) introduced in Example 1 and the rewrite system defined in Example 2. We have \( \sigma = \{ x \mapsto \{ x_1, x_2 \} \} \in \text{vars}_{R, AC}(\text{proj}_1(x)) \). Actually, \( \sigma \) together with the identity substitution form a complete set of variants for \( \text{proj}_1(x) \).

The following substitutions, together with the identity substitution, form a complete set of variants for \( x \oplus y \):

\[
\begin{align*}
\sigma_1 &= \{ x \mapsto y \oplus z \}, \sigma'_1 = \{ y \mapsto x \oplus z \}, \\
\sigma_2 &= \{ x \mapsto x' \oplus z, y \mapsto y' \oplus z \}, \\
\sigma_3 &= \{ y \mapsto x \}, \text{ and} \\
\sigma_4 &= \{ x \mapsto 0 \}, \sigma'_4 = \{ y \mapsto 0 \}.
\end{align*}
\]

This finite variant property is satisfied by many equational theories of interest, e.g., symmetric and asymmetric encryptions, signatures, blind signatures, zero-knowledge proofs. Moreover, such a property plays an important role regarding equational unification. It implies the existence of a complete set of unifiers, and gives us a way to compute it effectively [29].

**Definition 2:** Consider an AC-convergent \( R \) and a set of equations \( \Gamma = \{ u_1 = v_1, \ldots, u_k = v_k \} \). A set of substitutions \( S \) is a complete set of \( R \), AC-unifiers for \( \Gamma \) if:

1) for each \( \sigma \in S \) and \( i \in \{ 1, \ldots, k \} \), \( u_i \sigma \downarrow =_{\text{AC}} v_i \downarrow \sigma \downarrow ; \)
Example 1. When receiving such a message, the reader will send a nonce for the variable $x$. We assume the usual definitions of parallel composition and that we aim at verifying trace equivalence, parallel composition can be added as syntactic sugar to denote the set of all interleavings \[18\]. Therefore, in equivalence, parallel composition can be added as syntactic calculus without replication and that we aim at verifying trace equivalence, we will show that this protocol fails to achieve this unlinkability property.

To define the semantics of our calculus, we introduce the notion of deducibility. At a particular point in time, after some interaction with a protocol, an attacker may know a sequence of messages $t_1, \ldots, t_\ell \in \mathcal{T}(\Sigma, N)$. Such a sequence is organised into a frame $\varphi = \{w_1 \mapsto t_1, \ldots, w_\ell \mapsto t_\ell\}$ that is a substitution of size $|\varphi| = \ell$.

Definition 3: Let $\varphi$ be a frame, $t \in \mathcal{T}(\Sigma, N)$ and $R \in \mathcal{T}(\Sigma, N_{\text{pub}} \cup \text{dom}(\varphi))$. We say that $t$ is deducible from $\varphi$ using $R$, written $R \vdash_R t$, when $R \varphi \downarrow = \text{AC} t$.

Intuitively, an attacker is able to deduce new messages by applying function symbols in $\Sigma$ to names in $N_{\text{pub}}$ and terms stored in $\varphi$. The term $R$ is called a recipe.

Example 5: Continuing Example 2 consider the frame

$$\varphi = \{w_1 \mapsto r_1, w_2 \mapsto \langle id \oplus r_2, h(\langle r_1, k \rangle) \oplus r_2\rangle\}.$$  

We have that $h(\langle r_1, k \rangle)$ is deducible from $\varphi$ using the recipe $R = (\text{proj}_1(w_2) \oplus \text{proj}_2(w_2)) \oplus \langle id \rangle$.

We now define the semantics of our process calculus by means of a labelled transition relation on configurations. A configuration is a pair $(P, \varphi)$ where $P$ is a ground process, and $\varphi$ is a frame used to record the messages that the participants have sent previously.

The relation $\xrightarrow{\ell}$ where $\ell$ is either an input, an output, or an unobservable action test is defined as follows:

\[
\text{RECV } (\text{in}(c, x). P, \varphi) \xrightarrow{\text{in}(c, R)} (P\{x \mapsto t\}, \varphi) \text{ if } \varphi \vdash_R t
\]

\[
\text{SEND } (\text{out}(c, t). P, \varphi) \xrightarrow{\text{out}(c)} (P, \varphi \cup \{w_{|\varphi|+1} \mapsto t\})
\]

\[
\text{TEST } (s = t). P, \varphi \xrightarrow{\text{test}} (P, \varphi) \text{ if } s \downarrow = \text{AC} t
\]

Using our formalism, we can model the two roles of this protocol with the following ground processes:

$$P_{\text{tag}} = \text{in}(c, x). \text{out}(c, \langle id \oplus r_2, h(\langle r_1, k \rangle) \oplus r_2\rangle). 0$$

$$P_{\text{reader}} = \text{out}(c, r_1). \text{in}(c, y). \left(\text{proj}_1(y) \oplus \langle id \rangle \oplus \text{proj}_2(y) = h(\langle r_1, k \rangle)\right), 0$$

where $r_1, r_2, k \in N_{\text{prv}}, \langle id \rangle \in N_{\text{pub}}$, and $x, y \in \mathcal{X}$. The protocol itself corresponds to the set of ground processes obtained by interleaving these two roles.

The aim of this protocol is not only to authenticate the tag but also to ensure its unlinkability. An attacker should not be able to observe whether he has seen the same tag twice or two different tags. We will formalize this later on, relying on a notion of trace equivalence, and we will show that this protocol fails to achieve this unlinkability property.
Example 6: Consider the ground process \( P_{\text{diff}} \) that models an execution of the tag \( id \) (who shares the key \( k \) with the reader) followed by an execution of the tag \( id' \) (who shares the key \( k' \) with the reader), i.e., \( P_{\text{diff}} = P_{\text{tag}} \cdot P_{\text{tag}}' \) where:

\[
P_{\text{tag}} = \text{in}(c, x) \cdot \text{out}(c, \langle id \oplus r_2, h((r_1, k)) \oplus r_2 \rangle)
\]

\[
P_{\text{tag}}' = \text{in}(c, x') \cdot \text{out}(c, \langle id' \oplus r_2', h((r_1', k')) \oplus r_2' \rangle)
\]

We also consider the ground process \( P_{\text{same}} \) obtained from \( P_{\text{diff}} \) by replacing the occurrence of \( id' \) (resp. \( k' \)) by \( id \) (resp. \( k \)) (but keeping the nonce \( r_2' \)). This process models an execution of two instances of the protocol by the same tag \( id \) (who shares the key \( k \) with the reader). Following our semantics, we have the derivations described in Figure 2 where \( r_1 \in N_{\text{pub}}, i.e., r_1 \) is a public name known by the attacker.

When \( \ell \neq \text{test} \) we define \( \Rightarrow \) to be \( \text{test}^* \cdot \ell \cdot \text{test}^* \) and we lift \( \Rightarrow \) to \( \Rightarrow \) to sequences of actions. Given a protocol \( \mathcal{P} \), we write \( \langle \mathcal{P}, \varphi \rangle \xrightarrow{\ell_1, \ldots, \ell_n} \langle \mathcal{P}', \varphi' \rangle \) if there exists \( P \in \mathcal{P} \) such that \( \langle P, \varphi \rangle \xrightarrow{\ell_1, \ldots, \ell_n} \langle P', \varphi' \rangle \), and similarly for \( \Rightarrow \).

D. Process equivalence

In order to define our notion of equivalence, we first define what it means for a test to hold on a frame.

Definition 4: Let \( \varphi \) be a frame and \( R_1, R_2 \) be two terms in \( T(\Sigma, N_{\text{pub}} \cup \text{dom}(\varphi)) \). The test \( R_1 \cong R_2 \) holds on frame \( \varphi \), written \( (R_1 \cong R_2)\varphi \), if \( R_1 \varphi \downarrow = AC \equiv AC \).

Example 7: Consider the frames \( \varphi_{\text{diff}} = \{w_1 \mapsto t, w_2 \mapsto t'\} \) and \( \varphi_{\text{same}} = \{w_1 \mapsto t, w_2 \mapsto t''\} \) where the terms \( t, t' \), and \( t'' \) are as follows:

- \( t = \langle id \oplus r_2, h((r_1, k)) \oplus r_2 \rangle \)
- \( t' = \langle id' \oplus r_2', h((r_1', k')) \oplus r_2' \rangle \)
- \( t'' = \langle id \oplus r_2', h((r_1, k)) \oplus r_2' \rangle \)

They correspond to the frames obtained at the end of the executions considered in Example 6. The test

\[
\text{proj}_1(w_1) \oplus \text{proj}_2(w_1) \Rightarrow \text{proj}_1(w_2) \oplus \text{proj}_2(w_2)
\]

holds in \( \varphi_{\text{same}} \) but not in \( \varphi_{\text{diff}} \). An attacker can xor the two components of each message and check whether this computation yields an equality or not.

Definition 5: A protocol \( \mathcal{P} \) is trace included in a protocol \( Q \), denoted \( \mathcal{P} \subseteq Q \), if whenever \( \langle \mathcal{P}, \emptyset \rangle \xrightarrow{\ell_1, \ldots, \ell_n} \langle \mathcal{P}', \varphi \rangle \) and \( (R_1 = R_2)\varphi \), then there exists a configuration \( \langle Q', \varphi' \rangle \) such that \( \langle Q, \emptyset \rangle \xrightarrow{\ell_1, \ldots, \ell_n} \langle Q', \varphi' \rangle \) and \( (R_1 = R_2)\varphi' \).

This notion of equivalence does not coincide with the usual notion of trace equivalence as defined e.g. in [20]. It is actually coarser and is therefore sound for finding attacks. Moreover, it has been shown that the two notions coincide for the class of determinate processes [13].

Definition 6: We say that a protocol \( \mathcal{P} \) is determinate if whenever \( \langle \mathcal{P}, \emptyset \rangle \xrightarrow{\ell_1, \ldots, \ell_n} \langle \mathcal{P}', \varphi \rangle \) and \( \langle \mathcal{P}, \emptyset \rangle \xrightarrow{\ell_1, \ldots, \ell_n} \langle \mathcal{P}', \varphi' \rangle \), then for any test \( R_1 \cong R_2 \), we have that: \( (R_1 = R_2)\varphi \) if, and only if \( (R_1 = R_2)\varphi' \).

In general, checking determinacy is as difficult as checking equivalence. However, it is typically ensured easily in practice: for instance, any protocol whose roles have a deterministic behaviour can be modeled as a determinate process using a different channel for each role. In case processes are not determinate, the above relation can be used to disprove trace equivalence, i.e., find attacks. It is also possible to check a more fine-grained notion of trace equivalence which implies the usual notion of trace equivalence. This fine-grained notion can be verified straightforwardly by using the algorithm for verifying the above defined (coarse-grained) trace equivalence in a black-box manner, cf. [18] for details.

Example 8: Going back to our running example, we have that \( \mathcal{P}_{\text{same}} = \{P_{\text{same}}\} \) and \( \mathcal{P}_{\text{diff}} = \{P_{\text{diff}}\} \) are not in equivalence according to our definition (as well as the usual notion of trace equivalence since these two protocols are determinate). More precisely, we have that \( \mathcal{P}_{\text{same}} \not\subseteq \mathcal{P}_{\text{diff}} \).

Indeed, we have shown that:

\[
\langle P_{\text{same}}, \emptyset \rangle \xrightarrow{\text{in}(c, r_1), \text{out}(c), \text{in}(c, r_1), \text{out}(c)} \langle \emptyset, \varphi_{\text{same}} \rangle \text{, and}
\]
\[
\langle \text{proj}_1(w_1) \oplus \text{proj}_2(w_1) \rangle = \langle \text{proj}_1(w_2) \oplus \text{proj}_2(w_2) \rangle \text{.}
\]

However, the only extended trace \( \langle P', \varphi' \rangle \) such that

\[
\langle P_{\text{diff}}, \emptyset \rangle \xrightarrow{\text{in}(c, r_1), \text{out}(c), \text{in}(c, r_1), \text{out}(c)} \langle P', \varphi' \rangle
\]

is \( \langle \emptyset, \varphi_{\text{diff}} \rangle \) and we have seen that \( \langle \text{proj}_1(w_1) \oplus \text{proj}_2(w_1) \rangle \neq \langle \text{proj}_1(w_2) \oplus \text{proj}_2(w_2) \rangle \) does not hold in \( \varphi_{\text{diff}} \) (see Example 7).

However, we have that \( \mathcal{P}_{\text{diff}} \subseteq \mathcal{P}_{\text{same}} \). This is a non trivial inclusion that has been checked using our tool.

III. MODELLING USING HORNS CLAUSES

Our procedure is based on a fully abstract modelling of a process in first-order Horn clauses.
We have that \( P \models w \in T \) assume that such variables may only be substituted by terms in sequences of symbolic labels.

\[ \ell \in \{ \text{in}(c,t), \text{out}(c), \text{test} \mid t \in T(\Sigma, N \cup \mathcal{X}), c \in C_h \} \]

The empty sequence is denoted \( \epsilon \). Intuitively, a symbolic run stands for a set of possible runs of the protocol. We denote \( u \sqsubseteq AC \) \( v \) when \( u \) is a prefix (modulo \( AC \) of \( v \).

We assume a set \( Y \) of recipe variables disjoint from \( \mathcal{X} \), and use capital letters \( X, Y, Z \) to range over \( Y \). We assume that such variables may only be substituted by terms in \( T(\Sigma, N_{\text{pub}} \cup W \cup Y) \).

Our logic is based on four predicates, whose semantics is given in Figure 3 where \( w \) denotes a symbolic run, \( R, R' \) are terms in \( T(\Sigma, N_{\text{pub}} \cup W \cup Y) \), and \( t \) is a term in \( T(\Sigma, N \cup \mathcal{X}) \).

Intuitively:

- **reachability predicate**: \( r_w \) holds when the run represented by \( w \) is executable;
- **intruder knowledge predicate**: \( k_w(R, t) \) holds if whenever the run represented by \( w \) is executable, the message \( t \) can be constructed by the intruder using the recipe \( R \);
- **identity predicate**: \( i_w(R, R') \) holds if whenever the run \( w \) is executable, \( R \) and \( R' \) are recipes for the same term; and
- **reachable identity predicate**: \( r_i_w(R, R') \) is a shortcut for \( r_w \land i_w(R, R') \).

A (ground) atomic formula is interpreted over a pair consisting of a process \( P \) and a frame \( \varphi \), and we write \( (P, \varphi) \models f \) when the atomic formula \( f \) holds for \( (P, \varphi) \) or simply \( P \models f \) when \( \varphi \) is the empty frame.

We consider first-order formulas built from the above atomic formulas using conjunction, implication and universal quantification. The semantics is as expected, but the domain of quantified variables depends on their type: variables in \( \mathcal{X} \) may be mapped to any term in \( T(\Sigma, N) \), while recipe variables in \( Y \) are mapped to recipes, i.e. terms in \( T(\Sigma, N_{\text{pub}} \cup W) \).

**Example 9**: Continuing our running example, let \( w = \text{in}(c, r_1), \text{out}(c), \text{in}(c, r_1), \text{out}(c) \), \( t_0 = \text{id} \land h(r_1, k) \), and \( R_i = \text{proj}_1(w_i) \land \text{proj}_2(w_i) \) with \( i \in \{1, 2\} \). We have that:

\[ (P_{\text{same}}, \emptyset) \models r_w \land k_w(R_1, t_0) \land k_w(R_2, t_0) \land i_w(R_1, R_2) \]

Consider \( t = (\text{id} \land r_2, h((x, k)) \land r_2) \) and the formulas

- \( f_1 = \forall X, x, f_{\text{in}(c, x), \text{out}(c)} \iff k_v(X, x) \);
- \( f_2 = \forall X, x, k_{\text{in}(c, x), \text{out}(c)}(w_1, t) \iff k_v(X, x) \).

We have that \( P_{\text{same}} \models f_1 \) and \( P_{\text{same}} \models f_2 \).

**A. Predicates**

We define symbolic runs, denoted \( u, v, w \), as finite sequences of symbolic labels.

**B. Seed statements**

We shall be mainly concerned with particular forms of Horn clauses which we call statements.

**Definition 7**: A statement is a Horn clause of the form \( H \models k_{u_1}(X_1, t_1), \ldots, k_{u_n}(X_n, t_n) \) where:

- \( H \in \{ r_{u_0}, k_{u_0}(R, t), i_{u_0}(R, R'), r_{i_{u_0}}(R, R') \} \);
- \( u_0, u_1, \ldots, u_n \) are symbolic runs such that \( u_i \sqsubseteq AC \) \( u_0 \) for any \( i \in \{1, \ldots, n\} \);
- \( t_i, \ldots, t_n \in T(\Sigma, N \cup \mathcal{X}) \);
- \( R, R' \in T(\Sigma, N_{\text{pub}} \cup W \cup Y) \); and
- \( X_1, \ldots, X_n \) are distinct variables from \( Y \).

Lastly, \( \text{vars}(t) \subseteq \text{vars}(t_1, \ldots, t_n) \) when \( H = k_{u_0}(R, t) \).

In the above definition, we implicitly assume that all variables are universally quantified, i.e., all statements are ground.

By abuse of language we sometimes call \( \sigma \) a grounding substitution for a statement \( H \models B_1, \ldots, B_n \) when \( \sigma \) is grounding for each of the atomic formulas \( H, B_1, \ldots, B_n \). The **skeleton** of a statement \( f \), denoted \( \text{ske}(f) \), is the statement where recipes are removed.

\[ r_{\ell_1 \sigma \downarrow, \ldots, \ell_m \sigma \downarrow} \iff \{ k_{\ell_1 \sigma \downarrow, \ldots, \ell_1 \sigma \downarrow}(X_j, x_j \sigma \downarrow) \} \in R(m) \] for all \( 0 \leq m \leq n \)
for all \( \sigma \in \text{csu}_{\text{AC}}(\{ s_k = t_k \} \in R(m) \)
for all \( \tau \in \text{variants}_{\text{AC}}(\ell_1 \sigma_1, \ldots, \ell_m \sigma_m) \)

k_{\ell_1 \sigma \downarrow, \ldots, \ell_m \sigma \downarrow}(w_j(S(m))) \cdot t_m \sigma \downarrow \downarrow \iff \{ k_{\ell_1 \sigma \downarrow, \ldots, \ell_1 \sigma \downarrow}(X_j, x_j \sigma \downarrow) \} \in R(m) \] for all \( m \in S(n) \)
for all \( \sigma \in \text{csu}_{\text{AC}}(\{ s_k = t_k \} \in R(m) \)
for all \( \tau \in \text{variants}_{\text{AC}}(\ell_1 \sigma_1, \ldots, \ell_m \sigma_m, t_m \sigma) \)

\[ k_v(c, c) \iff \text{for all public names } c \in \Lambda_{\text{pub}} \]

**Fig. 4: Seed statements**

As mentioned earlier, our decision procedure is based on a fully abstract modelling of a process in first-order Horn clauses. In this section, given a ground process \( P \) we will
where \( w \) respectively denote the set of indices of the receive, send and stage. However, since we will give the attacker the abilities to the output term. The two last families of statements model one to successfully pass all the tests. Following the same idea, \( \ell \) represented by unifiers modulo the whole equational theory

\[
2) \text{ if } f \|
\]

For each \( \mathbf{S} \), we start by fixing some conventions.

\( \exists \mathbf{x} \mathbf{a} (\mathbf{P}, \mathbf{x}) \) we denote by \( \mathbf{a} \) an arbitrary \( w \).

\( \mathbf{C} \). Soundness and completeness

We show that as far as reachability and intruder knowledge predicates are concerned, the set seed\((P)\) is a complete abstraction of a process. However, we need one more definition to state this fact.

Definition 8: Given a set \( K \) of statements, \( \mathcal{H}(K) \) is the smallest set of ground facts such that:

\[
\begin{align*}
\sigma \text{ grounding for } f & \text{ with } \text{skel}(f) \in K \\
\text{CONSEQ} & \quad H\sigma \in \mathcal{H}(K) \\
\text{EXTEND} & \quad k_u(R,t) \in \mathcal{H}(K) \\
\end{align*}
\]

Theorem 1: Let \( P \) be a ground process.

\[
\begin{align*}
P & \models f \text{ for any } f \in \text{seed}(P) \cup \mathcal{H}(\text{seed}(P)) \\
& \quad \text{If } \langle Q, \varphi \rangle \models f \text{ for some } (Q, \varphi), \text{then} \\
1) & \quad r_{L_1 \ldots L_m} \models \mathcal{H}(\text{seed}(P)); \text{ and} \\
2) & \quad \varphi \models t \text{ then } k_{L_1 \ldots L_m} (R, t) \in \mathcal{H}(\text{seed}(P)).
\end{align*}
\]

We will see that the completeness of seed\((P)\) can be built upon to achieve full abstraction, i.e., also including identities of the process \( P \) and a procedure for checking equivalence.

IV. Saturation

We shall now describe how to verify equivalence given the protocol representation as Horn clauses introduced in the previous section. Given a ground process \( P \) and a protocol \( \mathcal{Q} \), we saturate the set of seed statements for \( P \) to construct a set of simple statements which we will call solved statements. The saturation procedure ensures that the set of solved statements is a complete abstraction of \( P \). Then, we use the resulting solved statements to decide whether \( P \) is trace included in \( \mathcal{Q} \). Repeating this procedure for all \( P \in \mathcal{P} \), and doing similarly for processes in \( \mathcal{Q} \), we are then able to decide whether two determinate protocols \( P \) and \( \mathcal{Q} \) are in trace equivalence.

In this section we will describe the saturation procedure. It manipulates a set of statements called a knowledge base.

Definition 9: Given a statement \( f = (H \models B_1, \ldots, B_n) \),

\[
\begin{align*}
\bullet & \quad u = \text{in}(c, x), \text{out}(c), v = \text{in}(c, x').\text{out}(c); \\
\bullet & \quad t = (id \oplus r_2, h(x, k)) \oplus r_2; \\
\bullet & \quad t' = (id \oplus r_2, h(x', k)) \oplus r_2.
\end{align*}
\]

The following statements belong to seed\((P_{\text{same}})\):

\[
\begin{align*}
r_u & \equiv k_r(x, x) \\
k_{u}(w, t) & \equiv k_r(x, x) \\
k_{u}(w_1, t) & \equiv k_r(x, x) \\
k_{u}(w_2, t') & \equiv k_r(x', x), k_r(x, x)
\end{align*}
\]

When considering the last kind of statements in Figure 4 with \( f = \text{proj}_1 \) and the empty run, we obtain:

\[
\begin{align*}
k_r(\text{proj}_1(x), \text{proj}_1(x)) & \equiv k_r(x, x) \\
k_r(\text{proj}_1(x, x_1)) & \equiv k_r(x, \{x_1, x_2\})
\end{align*}
\]

Figure 5 shows those obtained for \( f = \oplus \) and an arbitrary \( w \).
• if is said to be solved if for all 1 ≤ i ≤ n, we have that 
  \( B_i = k_{x_1,\ldots,x_{i}}(X_i, x_i) \) for some \( x_i \in \mathcal{X} \), and \( X_i \in \mathcal{Y} \).
• f is said to be well-formed if whenever it is solved and 
  \( H = k_{x_1,\ldots,x_{n}}(R, t) \), we have that \( t \notin \mathcal{X} \).

A set of well-formed statements is called a knowledge base. If \( K \) is a knowledge base, \( K_{\text{solved}} = \{ f \in K \mid f \text{ is solved} \} \).

Given an initial knowledge base \( K \), the saturation procedure is a non-deterministic process which produces another knowledge base. At each step of the saturation procedure, a new statement is generated and the knowledge base is updated with the new statement. This two-step process is repeated until a fixed point is reached. We denote by \( \text{sat}(K) \) the set of reachable fixed points starting from the initial set \( K \).

Before describing these two steps in Section IV-B, we explain in the following section why a naive adaptation of the original AKISS procedure would not be effective.

A. Difficulties

In the original procedure [18], the saturated knowledge base is obtained by applying (among others) the following resolution rule based on most general unifiers (mgu):

\[
\begin{align*}
  f &= (H \Leftarrow k_w(X, t), B_1, \ldots, B_n) \in K \\
  g &= (k_w(R, t') \Leftarrow B_{n+1}, \ldots, B_m) \in K_{\text{solved}} \\
  \sigma &= \text{mgu}(k_w(X, t), k_w(R, t')) \\
  K &= K \cup h
\end{align*}
\]

where \( h = \left( (H \Leftarrow B_1, \ldots, B_n, B_{n+1}, \ldots, B_m)\sigma \right) \).

This rule is close to a standard resolution step, between an unsolved statement \( f \) and a solved deduction statement \( g \). Note, however, that we do not impose the two symbolic runs involved in this resolution step to be unifiable but we only require that \( w \) is unifiable with a prefix of the other symbolic run (namely \( u \)). Intuitively, this comes from the fact that the knowledge of the attacker is monotone (the attacker never forgets any data), and a term \( t'\sigma \) deducible in the run \( w\sigma \) will remain deducible in any extension of this run.

For the sake of simplicity, we consider here that the update operator \( \cup \) simply adds \( h \) to \( K \). A naive approach to add the xor operator consists in replacing the condition \( \sigma = \text{mgu}(k_w(X, t), k_w(R, t')) \) by \( \sigma \in \text{mgu}_{AC}(k_w(X, t), k_w(R, t')) \), i.e., performing unification modulo AC instead of simply computing the mgu between these two terms. The obtained procedure would be correct but would rarely terminate.

Example 11: Let \( P = \text{in}(c, z_1), \text{in}(c, z_2), [z_2 = a \oplus z_1], 0 \) and \( w = \text{in}(c, z_1), \text{in}(c, a \oplus z_1) \). The set seed\((P)\) contains (among others) the following statements:

\[
\begin{align*}
  \text{i}_{w, \text{test}} &\Leftarrow k_w(Z_1, z_1), k_w(Z_2, a \oplus z_1) \\
  k_w(X \oplus Y, x \oplus y) &\Leftarrow k_w(X, x), k_w(Y, y)
\end{align*}
\]

The resolution rule can be applied on these two statements, and one of the substitutions is

\[
\sigma = \{ x \mapsto a \oplus z_1, y \mapsto z_1, z_1 \mapsto z_1 \oplus z_1 \}
\]

resulting in the following statement:

\[
\text{i}_{w, \text{test}} \Leftarrow k_w(Z_1, z_1), k_w(Z_2, a \oplus z_1), k_w(Y, z_12).
\]

This statement can again be resolved using the same statement as before, yielding an infinite loop.

B. Saturation procedure

We now explain our saturation procedure which is inspired from the one given in [13]. First, as expected, we perform resolution modulo associativity and commutativity (AC) to capture algebraic properties of xor. Second, in order to achieve termination in practice, we constrained the resolution rules in various ways while preserving completeness of our procedure.

1) Generating new statements: Given a knowledge base \( K \), new statements \( f \) are generated by applying the rules in Figure 6. Each of these rules generates a new statement \( h \).

Roughly, as already explained, the rule RESOLUTION applies the standard rule of resolution from first-order logic between an unsolved statement \( f \) and a solved deduction statement \( g \) and allows us to propagate constraints imposed from a partial execution of a trace to its possible extensions through the unification of the symbolic runs \( u \) and \( w \).

The rule RESOLUTION+ does essentially the same for the statement \( g = f_0^+ \) but applies some special treatments to avoid non-termination issues. The rule EQUATION allows us to derive new identities on recipes that may be imposed by the execution of the protocol. The rule TEST allows us to conclude which identities necessarily hold in an execution of the protocol. Once the statement \( h \) is generated, we update the knowledge base \( K \) with \( h \). This process is explained below.

More precisely, we restrict the use of the resolution rule and we only apply it on a selected atom. To formalise this, we assume a selection function sel which returns \( \perp \) when applied on a solved statement, and an atom \( k_w(X, t) \) with \( t \notin \mathcal{X} \) when applied on an unsolved statement. Resolution must be performed on this selected atom.

In order to avoid the termination problem illustrated in the previous section when considering equational theories that include xor, we introduce a marking on atomic formulas in the hypothesis of unsolved statements. The marking is used to disallow resolution against a statement in \( f_0^+ \). We denote unsolved statements with their marking as

\[
H \Leftarrow B_1, \ldots, B_n | M
\]

where \( M \subseteq \{ B_1, \ldots, B_n \} \) is the set of hypotheses of the statement which are marked. Marking will only be used for unsolved statements, and we implicitly set an empty marking on newly generated solved statements. Statements in \( f_1^+ \) and \( f_2^+ \) will be marked directly when constructing the initial knowledge base. More precisely, we mark the two hypotheses of any statement in \( f_1^+ \cup f_2^+ \) (see Definition 14). Intuitively, completeness is preserved as derivation trees in \( \mathcal{H}(K) \) can always be reorganised by pushing the use of CONSEQ rules with statement in \( f_0^+ \) below those with statements in \( f_1^+ \).
and \( f_2^+ \). Other statements will be marked dynamically in rule \textsc{Resolution}+, i.e., when performing resolution against a statement in \( f_1^+ \): to decide which of the two new hypotheses has to be marked we rely on the following notions.

\textbf{Definition 10:} Given a term \( t \), we define \( \text{factor}(t) = \{t_1, \ldots, t_n\} \) when \( \bigoplus_i t_i = t \) and none of the \( t_i \) is itself a sum. The function \( \text{rigid}(t) \) returns a term \( t_i \in \text{factor}(t) \) such that \( t_i \notin \mathcal{X} \) or \( \perp \) if no such \( t_i \) exists.

When performing \textsc{Resolution}+ with a selected atom for which a rigid factor can be found, we mark the hypothesis of the generated statements that contains the factor returned by rigid. This factor has to be rigid in the sense that it cannot be a variable. Again, we can show that we preserve completeness when marking this hypothesis. When we need to derive a term which is a sum, and we decide to split this sum in two parts, we will assume that the chosen rigid factor has to be obtained in an atomic way (it cannot be the result of a sum anymore). This amounts to favour one arrangement among all the possible ones up to associativity and commutativity of the xor operator.

\textbf{Example 12:} Going back to Example [11], we have that \textsc{Resolution}+ will be applied between these two statements, and \( \text{rigid}(a \oplus z_1) \) necessarily returns \( a \). Therefore, the resulting statement becomes:

\[ r_{wσ} \Leftarrow k_w(Z_1, z_{11} \oplus z_{12}), k_wσ(X, a \oplus z_{11}), k_wσ(Y, z_{12}) \parallel M \]

where \( M = \{k_wσ(X, a \oplus z_{11})\} \).

This forbids the use of \textsc{Resolution}+ on \( k_wσ(X, a \oplus z_{11}) \). Provided that our function \( \text{sel} \) returns \( k_wσ(X, a \oplus z_{11}) \), the saturation procedure can now only do a \textsc{Resolution} rule on the next statement and therefore the non termination issue mentioned in Example [11] is avoided.

\textbf{Example 13:} The marking of \( f_1^+ \) and \( f_2^+ \) is also important to ensure termination in practice. Indeed, otherwise, it would be possible to apply the \textsc{Resolution}+ rule between \( f_1^+ \) (and \( f_1^+ \)) on the atom \( k_w(X_1, x_1 \oplus x_2) \). However, the term \( x_1 \oplus x_2 \) has no rigid factor, and among the resulting statements, we will find the following one:

\[ k_wσ(X_1 \oplus X_2 \oplus X_3, x_{11} \oplus x_{12}) \Leftarrow k_wσ(X_1, x_{11} \oplus x_{21}), k_wσ(X_2, x_{12} \oplus x_{22}), k_wσ(X_3, x_{21} \oplus x_{22}) \parallel \emptyset \]

As no literal is marked, whatever is the selection function, the \textsc{Resolution}+ rule could be applied and the saturation procedure would enter an infinite loop.

Finally, the \textsc{Resolution} and \textsc{Resolution}+ rules induce a parent/child relationship between the unsolved statement used in the rule and the generated statement, which will eventually be added to the knowledge base after canonization (see below). This relation allows us to define the ancestor of any statement to be the parent of its parent etc. until we reach an unsolved statement in the initial knowledge base. In the following, we need to distinguish all deduction statements whose oldest ancestor belongs to \( f_1^+ \cup f_2^+ \) (with marking). We call these statements \textit{VIP statement}, and they will deserve a privileged treatment in the update.

2) Update: We will now define the update operator \( \Uparrow \) which adds statements generated by the rules of Figure 6 to the knowledge base. We first need to introduce the set of consequences of a knowledge base.

\textbf{Definition 11:} Let \( K \) be a knowledge base, the set of consequences, \( \text{conseq}(K) \), is the smallest set such that

\textbf{Axiom}

\[ k_{uv}(R, t) \Leftrightarrow k_u(R, t), B_1, \ldots, B_m \in \text{conseq}(K) \]

\[ k_u(R, t) \Leftarrow B_1, \ldots, B_n \in K \quad \sigma \text{ a substitution} \]

\textbf{Res}

\[ k_{uv}(R, t) \sigma \Leftarrow C_1, \ldots, C_m \in \text{conseq}(K) \]

We shall see that a weak form of update is sufficient when considering a deduction statement that is already a consequence of the knowledge base up to a change of recipe.

\textbf{Definition 12:} The canonical form \( f \Uparrow \) of a statement

\[ f = (H \Leftarrow B_1, \ldots, B_n \parallel M) \]

is the statement obtained by first normalizing all the recipes, then applying the rule \textsc{Remove} as many times as possible, and for solved deduction statement, applying the rule \textsc{Shift} as many times as possible.

\textbf{Remove}

\[ H \Leftarrow k_u(Y, t), B_1, \ldots, B_n \parallel M \setminus k_{uv}(X, t) \]

\textbf{Shift}

\[ k_{uv}(R, t) \Leftarrow k_u(X, x), B_1, \ldots, B_n \text{ with } x \in \text{factor}(t) \]

\textbf{Definition 13:} Let \( K \) be a knowledge base, and \( f \) a statement. The update of \( K \) by \( f \), denoted \( K \Uparrow f \), is \( K \) when \( \text{skel}(f) \) is not in normal form (the statement is dropped). Otherwise, two options are possible:

- \( K \Uparrow f = K \cup \{f \Uparrow\} \)
- \( K \Uparrow f = K \cup \{k_w(R, t) \Leftarrow B_1, \ldots, B_n\} \) provided that
  - \( f \Uparrow \) is a solved statement but not a VIP one, and
  - \( (k_w(R', t) \Leftarrow B_1, \ldots, B_n) \in \text{conseq}(K \text{ solved}) \)

Note that the update is not a function because there may be several \( R' \) for which the second option can be chosen. Even when such an \( R' \) exists, we may still update the base by choosing the first option. These choices are implementation details, and our results hold regardless.

3) Initial knowledge base: We finally define on which knowledge base we initiate the saturation procedure.

\textbf{Definition 14:} Let \( P \) be a ground process, and \( N^0_{\text{pub}} \) be a set of names. We have that

\[ \text{seed}(P, N^0_{\text{pub}}) = f^+_0 \Uparrow f^+_1 \Uparrow f^+_2 \Uparrow f^+_3 \Uparrow f^+_4 \Uparrow S \]
\[ f = \left( H \iff k_u(X, t), B_1, \ldots, B_n \mid \mathcal{M} \right) \in K \text{ such that } k_u(X, t) = \text{sel}(f) \]

**Resolution**

\[ g = \left( k_u(R, t') \iff B_{n+1}, \ldots, B_m \right) \in K_{\text{solved}} \setminus f_0^+ \]

\[ K = K \cup \{ h \sigma \mid \sigma \in \text{csu}_{\mathcal{NA}}(k_u(X, t), k_u(R, t')) \} \]

where \( h = \left( H \iff B_1, \ldots, B_n, B_{n+1}, \ldots, B_m \mid \mathcal{M} \setminus \{ k_u(X, t) \} \right) \)

**Equation**

\[ f, g \in (K_{\text{solved}} \setminus f_0^+) \]

\[ f = \left( k_u(R, t) \iff B_1, \ldots, B_n \right) \]

\[ g = \left( k_{u'}(R', t') \iff B_{n+1}, \ldots, B_m \right) \]

\[ K = K \cup \{ h \sigma \mid \sigma \in \text{csu}_{\mathcal{NA}}(u, u') \} \text{ where } h = \left( i_{u'}(R', t') \iff B_1, \ldots, B_m \right) \]

**Test**

\[ f, g \in K_{\text{tested}} \]

\[ f = \left( i_u(R, R') \iff B_1, \ldots, B_n \right) \]

\[ g = \left( r_{u'} \iff B_{n+1}, \ldots, B_m \right) \]

\[ K = K \cup \{ h \sigma \mid \sigma \in \text{csu}_{\mathcal{NA}}(u, u') \} \text{ where } h = \left( r_{i_{u'}(R', t')} \iff B_1, \ldots, B_m \right) \]

Fig. 6: Saturation rules

where \( f_i \) (with \( 0 \leq i \leq 4 \)) are defined as given in Figure 5 and \( S \) are the remaining statements. Let \( K_0 \) be the set of statements which contains:

1) deduction statements in \( f_1^+ \cap \text{seed}(P, \mathcal{N}_\text{pub}) \);
2) deduction statements in \( f_2^+ \cup f_3^+ \cap \text{seed}(P, \mathcal{N}_\text{pub}) \) with their two hypotheses marked:

\[ k_u(X_1 \oplus X_2, t) \iff B_1, B_2 \mid \{ B_1, B_2 \} \]

The initial knowledge base associated to \( \text{seed}(P, \mathcal{N}_\text{pub}) \), denoted as \( \text{Kinit}(\text{seed}(P, \mathcal{N}_\text{pub})) \), is defined to be \( K_0 \) updated by the set \( S \), i.e.,

\[ \text{Kinit}(\text{seed}(P, \mathcal{N}_\text{pub})) = \left( (K_0 \cup g_1, g_2, \ldots, g_k) \right) \]

where \( g_1, \ldots, g_k \) is an enumeration of the statements in \( S \). We sometimes write \( \text{Kinit}(P) \) for \( \text{Kinit}(\text{seed}(P, \mathcal{N}_\text{pub})) \).

When building the initial knowledge base we first add some of the variants related to \( \oplus \). Note in particular that we mark all hypotheses of the variants in \( f_1^+ \) and \( f_2^+ \) [Figure 5], and we do not add statements in \( f_3^+ \) and \( f_4^+ \). All other seed statements are simply added using the update operator.

Please observe that \( \text{Kinit}(P) \) depends on the order in which statements in \( \text{seed}(P) \) are updated. The exact order, however, is not important and our results shall hold regardless of the chosen order. The saturation procedure takes \( \text{Kinit}(P) \) as an input and produces a knowledge base \( \text{Ksat} \in \text{sat}(\text{Kinit}(P)) \). The reason for choosing \( \text{Kinit}(P) \) instead of \( \text{seed}(P) \) as the starting point of the saturation procedure is that \( \text{seed}(P) \) may not be a knowledge base (recall that a knowledge base is a set of well-formed statements). The fact that the set \( \text{Kinit}(P) \) is, however, a knowledge base follows directly from the fact that we apply our SHIFT rule (through the canonization process) before adding a deduction statement to the current set.

**C. Soundness and completeness**

We shall show that the solved statements of any \( K \in \text{sat}(\text{Kinit}(P)) \) form a complete abstraction of \( P \), with respect to some extension of \( \mathcal{H} \) that we define next.

**Definition 15**: Let \( K \) be a set of statements. We define \( \mathcal{H}_e(K) \) to be the smallest set of ground facts containing \( \mathcal{H}(K) \) and that is closed under the rules of Figure 7.

**Theorem 2**: Let \( K \in \text{sat}(\text{Kinit}(P)) \) for some ground process \( P \). We have that \( P \vdash f \) for any \( f \in K \cup \mathcal{H}_e(K) \) and, if \( (P, \emptyset) \xrightarrow{L_1, \ldots, L_n} (Q, \varphi) \), then:

1) \( r_{L_1, \varphi, \ldots, L_n, \varphi} \in \mathcal{H}_e(\text{Ksolved}) \);
2) if \( \varphi \vdash R_t \) then \( k_{L_1, \varphi, \ldots, L_n, \varphi}(R, t) \in \mathcal{H}_e(\text{Ksolved}) \);
3) if $\varphi \vdash_R t$ and $\varphi \vdash_{R'} t$, then $i_{L_1,\varphi \vdash \ldots \vdash \varphi \vdash_{L_n}(R, R') \in H_{\text{es}}(K_{\text{esolved}})$.

The proof follows the same general outline as in the original argument without xor [18]. However, some points that were irrelevant or straightforward in the original proof require special attention here.

As explained before, our marking discipline is justified by means of rearrangements of CONSEQ rules with $f_{ij}$ statements in derivation trees of $H(K)$. Due to these rearrangements, inductions on derivation trees are not straightforward anymore. We sometimes have to prove the existence of a derivation tree which is smaller only for a complex measure, where in the original proof a standard induction on the size of the derivation tree or on the size of the recipe of the head was sufficient. The existence of these other derivation trees themselves relies on invariants of the saturation procedure that we enforce by the canonization rules and the distinguished VIP statements. This is in sharp contrast with the original proof where the canonization rules only act as an optimization to terminate faster.

We finally note that our saturation procedure also brings new improvements that are not directly related to xor (e.g., removal of non normal terms, canonization for unsolved statements) but which we had to introduce (and justify) to obtain an effective procedure supporting xor. Discarding these unnecessary statements could also improve the efficiency of the original AKISS procedure.

V. Algorithm

In this section, we first discuss the effectiveness and termination of the saturation procedure, and describe our algorithm to verify trace inclusion for determinate processes.

A. Effectiveness of the procedure

The termination of the procedure has been proved for the original version of AKISS but only for subterm convergent equational theories. It is shown, among others, that the initial knowledge base only needs to be derived from finitely many seed statements, and in particular it does not need to contains all fresh nonces the attacker can generate. As stated below, this result is also true in our setting.

Lemma 1: Let $P$ be a ground process, $N^P_{\text{pub}} \subseteq N_{\text{pub}}$ be the finite set of public names occurring in $P$. We have that:

$$\text{sat}(K_{\text{int}}(P)) \supseteq \{K \cup \text{ext}(K) \mid K \in \text{sat}(K_{\text{int}}(\text{seed}(P, N^P_{\text{pub}})))\}.$$  

where $\text{ext}(K)$ is the set containing the following statements:

- $k_e(n, n) \iff \text{for any } n \in N^P_{\text{pub}} \setminus N^P_{\text{pub}}$,
- $i_e(n, n) \iff \text{for any } n \in N^P_{\text{pub}} \setminus N^P_{\text{pub}}$,
- $r_i(n, n) \iff B_1, \ldots, B_n$ for any $r_u \iff B_1, \ldots, B_n \in K$ in solved form, any $n \in N_{\text{pub}} \setminus N^P_{\text{pub}}$.

Another issue is that, when computing the update operator, we need to check whether there exists $R$ such that the statement $k_w(R, t) \iff B_1, \ldots, B_n$ is a consequence of a set of solved statements. This can be achieved using a simple backward search, similar to the one in [18].

However, with the addition of the xor, the saturation procedure may itself not terminate even if the initial knowledge base is finite and each saturation step is computable. A first reason would be the use of an unsuitable selection function. In order to avoid termination issues, we will consider a selection function that selects in priority a marked literal when it exists, a literal which is not a sum otherwise, and one that contains a rigid factor as a last resort. In case there is no other choice than selecting a literal containing a sum of variables the saturation enters an infinite loop as illustrated by the following example.

Example 14: Consider the ground process:

$$P = \text{in}(c, x) \cdot \text{in}(c, y) \cdot \text{in}(c, z) \cdot [x = y \oplus z] \cdot 0.$$  

Among others, the set of seed statements will contain:

$$r_w \iff k_{w_1}(X, y \oplus z), k_{w_2}(Y, y), k_{w_3}(Z, z) \parallel \emptyset$$

where $w = \text{in}(c, y \oplus z) \cdot \text{in}(c, y) \cdot \text{in}(c, z) \cdot \text{test}$, and for some $w_1$, $w_2$, and $w_3$ that we do not specify since they are not relevant here. The RESOLUTION+ rule will be applied on the first hypothesis, and since there is no rigid factor in $y \oplus z$, no hypothesis will be marked. We will therefore generate (among others) a new statement of the following form:

$$r \iff k(X_1, y_1 \oplus z_1), k(X_2, y_2 \oplus z_2), k(Y, y_1 \oplus y_2), k(Z, z_1 \oplus z_2) \parallel \emptyset$$

on which the same RESOLUTION+ rule can be applied again, entering an infinite loop.

Even though this example illustrates that termination is not guaranteed, we were able to verify a large range of different protocols, as illustrated in Section VI.

B. Description and correctness of the algorithm

Our procedure is described in Figure 8. Let $P$ be a protocol, i.e., a finite set of processes, and $P$ be a ground process. Let $K^0 \in \text{sat}(K_{\text{int}}(\text{seed}(P, N^P_{\text{pub}})))$ be a saturation of $P$ where $N^P_{\text{pub}}$ is the set containing all public names occurring in $P$. In our procedure the process $P$ will be represented by the set $K^0_{\text{esolved}}$ of solved statements of $K^0$.

The test $\text{REACH-IDENTITY}(K^0_{\text{esolved}}, P)$ checks whether each reachable identity $r_{i_1, \ldots, i_n}(R, R') \iff B_1, \ldots, B_n$ in $K^0_{\text{esolved}}$ holds in $P$. To perform this check for a given reachable identity, we first compute the recipes $R_i$ that allow the process $P$ to execute the trace $\ell_1\sigma, \ldots, \ell_n\sigma$. The substitution $\sigma$ replaces variables by fresh names in order to close the run. Next we check whether the corresponding trace $(M_1, \ldots, M_n)$ is executable in $P$ and whether the test $\ell_1\omega \parallel R'\omega$ holds in the resulting frame $\varphi$, i.e., the frame reached by $P$ after performing $M_1, \ldots, M_n$. If all the tests succeed, $P$ is trace included in $P$. If one test fails, the algorithm returns this test as a witness of non equivalence.
REACH-IDENTITY($K_0^\text{sol}$, $P$)
For all $r_1, \ldots, r_k (R', R') \iff k_{w_1}(X_1, x_1), \ldots, k_{w_m}(X_m, x_m) \in R_0^\text{sol}
let $c_1, \ldots, c_k$ be fresh public names such that $\sigma : \text{vars}(\ell_1, \ldots, \ell_k) \cup \{x_1, \ldots, x_m\} \rightarrow \{c_1, \ldots, c_k\}$ is a bijection
for all $i$ such that $\ell_i = \text{in}(d_i, t_i)$, let $R_i$ be such that $k_{\ell_i, \sigma} \circ \cdots \circ k_{\ell_1, \sigma}(R_i, t, \sigma) \in H(K_0^\text{sol}) \cup \{k_r(c_i, c_1) \iff 1 \leq i \leq k\}$
let $M_i = \{\ell_i \iff \ell_i \in \text{test, out}(c) \mid c \in C\}$
\begin{align*}
\text{let } M_i = \{\ell_i \iff \ell_i \in \text{test, out}(c) \mid c \in C\} & \text{ if } \ell_i \in \{\text{test, out}(c) \mid c \in C\} \\
\text{let } M_i = \{\ell_i \iff \ell_i = \text{in}(d_i, t_i)\} & \text{ if } \ell_i = \text{in}(d_i, t_i) \\
\text{check that } (P, \emptyset) \Rightarrow M_1 \Rightarrow M_2 \Rightarrow \cdots \Rightarrow M_k \Rightarrow (P, \varphi) & \text{ and } R \varphi \Downarrow =_{AC} R' \varphi \Downarrow \text{ where } \omega = \{X_i \mapsto x_i\}. \text{ Fig. 8: Test for checking } P \nsubseteq P
\end{align*}

Theorem 3: Let $P$ be a ground process, and $N^{P}_{\text{pub}} \subseteq N^{P}_{\text{pub}}$ be the finite set of public names occurring in $P$. Let $P$ be a protocol, and $K^P_0 \in \text{sat}(K_{\text{init}}(\text{seed}(P, N^{P}_{\text{pub}})))$. We have that:
- if $P \subseteq P$ then REACH-IDENTITY($K_0^\text{sol}$, $P$) holds;
- if $P$ is determinate and REACH-IDENTITY($K_0^\text{sol}$, $P$) holds then $P \nsubseteq P$.

Note that REACH-IDENTITY requires to compute, for each input of the run, a recipe $R_i$. It necessarily exists (as the run $\ell_1, \ldots, \ell_n$ is reachable) and its computation amounts to finding $R$ such that $k_w(R, t) \in H(K)$ given $w, t, K$. This can be achieved using a simple recursive backward search, as in [13].

VI. IMPLEMENTATION AND CASE STUDIES
Given that our procedure may not terminate, our main goal was to evaluate whether termination is achieved in practice. We validate our approach by integrating our procedure in the tool AKISS [3] and by testing it on various examples.

A. Integration in AKISS
Our implementation includes the marking strategy, a selection function that returns in priority an hypothesis that is marked and reasoning modulo AC. In general, it is difficult to implement unification and computation of variants for theories involving an AC operator. We leverage an existing tool, namely Maude [35], which implements such algorithms efficiently. To minimize the high cost of external calls to Maude, we also implement a naive algorithm for AC unification that handles most of the simple cases internally, and we only call Maude for solving the difficult cases (∼3% of cases in practice).

To ease the specification of the protocols our tool supports additional operators in the process calculus for parallel composition ($P \parallel Q$), non-deterministic choice ($P + + Q$), sequential composition ($P :: Q$), and a phase operator ($P \gg Q$). The latter allows at any moment during the execution of $P$ to proceed to the execution of $Q$ and will be used among others to model resistance against guessing attacks as done e.g. in the ProVerif tool [10]. All these operators are syntactic sugar and can be translated to sets of (linear) processes in a straightforward way.

To mitigate the potential exponential blowup caused by this translation, partial order reduction techniques [8], [9] can be used. We integrate some of these optimisations which notably consist in automatically prioritising outputs when protocols are determinate [9].

Note that for AKISS finding attacks or proving their absence are equally difficult tasks, as for both it first completes the saturation of the traces. However, for some equivalence properties, one of the inclusions is trivially satisfied. In such a case, we only check the inclusion that is not trivially satisfied and therefore reduce by two the number of symbolic traces to explore. Note that our procedure easily allows to check inclusion instead of equivalence.

We now report on experimental results that have been obtained by running our tool on a 20 core Intel(R) Xeon(R) @ 3.10GHz with 300 GB of RAM.

B. Unlinkability of RFID protocols
We analyse an unlinkability property on various RFID protocols that rely on the xor operator. A description of these protocols can be found in [41], [36], [7], and one of them, namely the KCL protocol, is detailed in Example 4.

We model unlinkability as an interaction between the reader and either $\text{tag}_1$ or $\text{tag}_2$, assuming that the attacker has previously observed a session between the reader and $\text{tag}_1$. We use the process $P_{\text{tag}}$ (Example 4) to model the tag. The processes $P_{\text{tag}_1}$ and $P_{\text{tag}_2}$ are slightly different versions of $P_{\text{tag}}$. More precisely, they are obtained from $P_{\text{tag}}$ by replacing $id$ and $k$ by $id_1$ and $k_1$ (resp. $id_2$ and $k_2$). Then, we consider a process $P_{\text{init}}$ to model the outputs observed by an attacker during an honest interaction between $\text{tag}_1$ and the reader:

$P_{\text{init}} = \text{out}(c, r). \text{out}(c, \langle id_1 \oplus r', h(\langle r, k_1\rangle) \oplus r'\rangle), 0$

Using non-deterministic choice we model that the reader may engage a session with $\text{tag}_1$ (using $id_1$ and key $k_1$) or $\text{tag}_2$ (using $id_2$ and key $k_2$):

$P_{\text{reader}} = \text{out}(c, r_1), \text{in}(c, y), \langle \langle \text{proj}_1(y) \oplus id_1 \\rangle \oplus \text{proj}_2(y) = h(\langle r_1, k_1\rangle) \rangle, 0$

++ $\langle \langle \text{proj}_1(y) \oplus id_2 \\rangle \oplus \text{proj}_2(y) = h(\langle r_1, k_2\rangle) \rangle, 0$

Unlinkability can finally be expressed as the following process equivalence:

$P_{\text{init}} :: (P_{\text{tag}_1} \parallel P_{\text{reader}}) \Rightarrow P_{\text{init}} :: (P_{\text{tag}_2} \parallel P_{\text{reader}})$

The (known) attack explained in [Example 8] on a simplified scenario (without the readers) is again found by the tool. A possible fix would be to replace the message $h(\langle r_1, k_1\rangle) \oplus r_2$ by $h(\langle r_1, k_2\rangle)$. Our tool is then able to establish unlinkability.

In total we modelled 7 RFID protocols: KCL, LD, LAK, OTYT and YPL from [41], MW from [36], and AT from [7].
Note that one inclusion trivially holds: when considering two different tags less equalities hold than in the case of two identical tags. We can therefore avoid unnecessary computations and only check the other inclusion, i.e.:

\[ P_{\text{init}} :: (P_{\text{tag}} \parallel P_{\text{reader}}) \subseteq P_{\text{init}} :: (P_{\text{tag}} \parallel P_{\text{reader}}) \]

On 4 of the 7 protocols we find (known) attacks which violate unlinkability. The results are summarised in Figure 9 and confirm termination of the saturation procedure with our resolution strategy when analysing unlinkability on various RFID protocols. When there is no attack, we consider an additional scenario where the attacker can abort its first observation to start a new session:

\[
P_{\text{init}} :: ((P_{\text{tag}} \parallel P_{\text{reader}}) \triangleright (P_{\text{tag}} \parallel P_{\text{reader}}))
\]

\[
\approx P_{\text{init}} :: ((P_{\text{tag}} \parallel P_{\text{reader}}) \triangleright (P_{\text{tag}} \parallel P_{\text{reader}}))
\]

<table>
<thead>
<tr>
<th>RFID Protocol</th>
<th># sessions</th>
<th># traces</th>
<th>time</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCL (Ex. [4])</td>
<td>1</td>
<td>1</td>
<td>1s</td>
<td>×</td>
</tr>
<tr>
<td>KCL fixed</td>
<td>1</td>
<td>20</td>
<td>3s</td>
<td>×</td>
</tr>
<tr>
<td>KCL fixed</td>
<td>2</td>
<td>980</td>
<td>1m47</td>
<td>✓</td>
</tr>
<tr>
<td>LD</td>
<td>1</td>
<td>20</td>
<td>&lt;1s</td>
<td>×</td>
</tr>
<tr>
<td>OTYT</td>
<td>1</td>
<td>20</td>
<td>&lt;1s</td>
<td>×</td>
</tr>
<tr>
<td>YPL</td>
<td>1</td>
<td>20</td>
<td>&lt;1s</td>
<td>✓</td>
</tr>
<tr>
<td>LAK</td>
<td>2</td>
<td>980</td>
<td>30m</td>
<td>✓</td>
</tr>
<tr>
<td>MW</td>
<td>1</td>
<td>40</td>
<td>7s</td>
<td>✓</td>
</tr>
<tr>
<td>MW</td>
<td>2</td>
<td>4280</td>
<td>10h</td>
<td>✓</td>
</tr>
<tr>
<td>AT</td>
<td>1</td>
<td>20</td>
<td>1s</td>
<td>✓</td>
</tr>
<tr>
<td>AT</td>
<td>2</td>
<td>1040</td>
<td>33s</td>
<td>✓</td>
</tr>
</tbody>
</table>

We note ✓ when AKISS concludes that the property holds and × when it reports an attack.

Fig. 9: Summary for RFID protocols

C. Resistance against offline guessing attacks

We analyse resistance against offline guessing attacks on various password based protocols that rely on the exclusive-or operator from [31]. Protocols which rely on passwords are often subject to off-line guessing attacks: an attacker may observe, or even actively participate in an execution of the protocol and then, in a second, offline phase, verify if a guessed value is indeed the real password without further interaction with the protocol. During this offline verification phase, the attacker has to try all possible values for the password and perform a test to check whether a guess is the real password or not. This can be modelled relying on trace equivalence by checking whether the attacker can make a difference between a scenario where the real password is revealed at the end, and another one where a wrong password is revealed [24], [12].

We illustrate this encoding on the so-called direct authentication protocol [31] whose description is detailed below.

**Direct authentication protocol.** A and B initially share a poorly chosen secret \( pw \), and wish to establish a session key \( k \).

To achieve this goal, A generates a public key \( pub \) and sends it to B encrypted with \( pw \) together with a challenge \( ra \). B replies by computing a ciphertext that contains fresh nonces \( nb_1, nb_2 \), and another one \( cb \) called a confounder. This ciphertext also contains an encryption of the challenge \( ra \) with \( pw \) for authentication purposes. Then, A generates a fresh key \( k \), and a challenge response mechanism is used to ensure that both parties agree on the key.

\[
A \rightarrow B : ra, \{ pw \}_{pub}
\]

\[
B \rightarrow A : \{ B, A, nb_1, nb_2, cb, \{ ra \}_{pw} \}_{pub}
\]

\[
A \rightarrow B : \{ nb_1, k \oplus nb_2 \}_{pw}
\]

\[
B \rightarrow A : \{ f_1(ra), rb \}_k
\]

\[
A \rightarrow B : \{ f_2(rb) \}_k
\]

This protocol has been designed to be resistant against guessing attacks. An attacker should not be able to perform an off-line verification of whether a guess of the password is successful or not. Actually, several examples of such protocols are described in [31]. They try to achieve the same goal relying on slightly different primitives or considering a different environment (e.g. some of them rely on a trusted third party \( S \)).

We model resistance against guessing attacks by checking an equivalence property between a scenario where the real password is revealed at the end, and another where a wrong password (modeled through a fresh name) is revealed [24], [12]. For each protocol, we consider one session between two honest agents \( A \) and \( B \) (and the trusted third party \( S \) when needed). Resistance against guessing attacks is expressed through the following equivalence:

\[
(P_A \parallel P_B) \triangleright out(c, pw) \approx (P_A \parallel P_B) \triangleright out(c, pw')
\]

where \( pw \) is the real password, and \( pw' \) is the fresh name.

In total, we modelled 5 password-based protocols: the Toy, Nonece, Secret Public Key, and Direct Authentication protocols from [31], as well as a protocol by Gong [30]. For each protocol we first verify resistance against guessing attacks in the presence of a passive adversary, i.e. a pure eavesdropper. Whenever this equivalence holds, we analyse the active case for one session. Since one inclusion trivially holds, we only check the other one, i.e.:

\[
(P_A \parallel P_B) \triangleright out(c, pw) \subseteq (P_A \parallel P_B) \triangleright out(c, pw')
\]

The results are summarised in Figure 10 and illustrate that our tool can also be effectively used to analyse resistance against guessing attacks on various password-based protocols. The Direct Auth protocol has been tested with two instances of each role in parallel, the other protocols (Nonece and Sec Pub Key) require more than 50Gb of memory and have not been successfully tested.

D. Other case studies

We have also encoded some authentication properties as equivalences for both RFID protocols that guarantee unlinkability (the LAK and fixed KCL protocols) and for a xor-based variant of the NSL protocol [21]. For both LAK and NSL-xor
We note ✓ when AKISS concludes that the property holds and ✗ when it reports an attack.

Fig. 10: Summary for password-based protocols

Finally, on the NSL-xor protocol we verified strong secrecy of the nonces \( n_a \) and \( n_b \), as defined by Blanchet [14]: the adversary initially provides two values and must be unable to distinguish the situations where the first, respectively, the second value is used as the secret. For instance strong secrecy of the nonce \( n_a \) is modelled as follows.

\[
\text{in}(c, x_1).\text{in}(c, x_2) : NSL\{z_1/n_a\} \\
\approx \\
\text{in}(c, x_1).\text{in}(c, x_2) : NSL\{z_2/n_a\}
\]

We show that neither \( n_a \) nor \( n_b \) are strongly secret, even when we only consider one honest session among \( A \) and \( B \). The results are summarised in Figure 11.

Fig. 11: Summary for other protocols

VII. CONCLUSION

We presented what we believe is the first effective procedure to verify equivalence properties for protocols that use xor. The need for such verification techniques is among others motivated by the unlinkability property in RFID protocols. Our procedure builds on the theory underlying the AKISS tool and presents a novel resolution strategy which we show to be complete. Even though termination is not guaranteed the tool did terminate on all practical examples that we have tested.

Directions for future work include adding new canonization rules to extend the termination proof of AKISS [18] to theories including the xor operator. Another direction is to consider other AC operators such as Diffie-Hellman exponentiation and bilinear pairings, which are supported by the Tamarin tool [33].

REFERENCES

ROOFS REGARDING SEED STATEMENTS

The soundness part of Theorem 1 is a consequence of Lemma 2 and Lemma 3.

Lemma 2: Let $P$ be a ground process. We have that $P \models f$ for any statement $f \in seed(P)$.

Proof. We suppose the same naming conventions for $P$ as in Definition of seed statements (see Section III-B). We prove that for each statement $f \in seed(P)$ we have that $P \models f$. There are four kinds of seed statements (see Figure 4) which we consider one-by-one.

Case 1. Let $m$ be such that $0 \leq m \leq n$, $\sigma$ and $\tau$ be substitutions such that $\sigma \in \text{csu}_{\text{R,AC}}(\{s_k = t_k\}_{k \in T(m)})$ and $\tau \in \text{vars}_{\text{R,AC}}(\ell_1,\ldots,\ell_m)$. Let $f$ be the following statement:

$$r_1\sigma\downarrow_{\ldots}r_m\sigma\downarrow \models \{k_1\sigma\downarrow_{\ldots}k_j\sigma\downarrow(X_j,x_j\sigma\downarrow)\}_{j \in R(m)}$$

We have to show that $P \models f$. Let $\omega$ be an arbitrary substitution grounding for $f$. We assume furthermore that $P \models \{k_1\sigma\downarrow_{\ldots}k_j\sigma\downarrow(X_j,x_j\sigma\downarrow)\}_{\omega} \models \tau \models \text{any } j \in R(m)$. We have to show that $P \models \{r_1\sigma\downarrow_{\ldots}r_m\sigma\downarrow\}_{\omega}$. In fact we will show a stronger statement. We show that

$$P \models \{r_1\sigma\downarrow_{\ldots}r_m\sigma\downarrow\}_{\omega} \models \omega \leq p \leq m.$$  

We proceed by induction on $p$. When $p = 0$, we have that $\{r_1\sigma\downarrow_{\ldots}r_m\sigma\downarrow\}_{\omega} \models \tau \models \text{trivially.}$

Now, assume that $p > 0$, and $P \models \{r_1\sigma\downarrow_{\ldots}r_{p-1}\sigma\downarrow\}_{\omega}$. We have to show that $P \models \{r_1\sigma\downarrow_{\ldots}r_{p}\sigma\downarrow\}_{\omega}$ by case analysis on $ap$. Before we do the case analysis, let us first fix some notations.

Let $(P_1,\varphi_1) = (P,\emptyset)$. As $P \models r_1\sigma\downarrow_{\ldots}r_{p-1}\sigma\downarrow\_\models \omega \uparrow_{\ldots}i < p$ we have that there exist $L_1,\ldots,L_{p-1}$ such that for all $1 \leq i < p$

$$(P_i,\varphi_i) \rightarrow (P_{i+1},\varphi_{i+1})$$

and $\varphi_i$ extends $\varphi_{i-1}$.

We can now do the case analysis.

1) Case $ap = \text{out}(c_p,t_p)$. We have that $p = \text{out}(c_p)$. Let $L_p = \text{out}(c_p)$, $\ell_{p+1} = \{ap_{1+1},\ldots,an\}_{\downarrow_{\omega}X_j \rightarrow x_j\sigma\downarrow} \uparrow_{\omega}$ and $\varphi_{p+1} = \varphi_p \cup \{x_{\downarrow_{\omega}}\sigma \uparrow \downarrow\}$. We have:

$$\begin{array}{c}
(P_p,\varphi_p) \rightarrow (P_{p+1},\varphi_{p+1})
\end{array}$$

which is what we wanted to prove.

2) Case $ap = [s_p = t_p]$. We have that $\ell_p = \text{test}$. Let $P_{p+1} = \{ap_{1+1},\ldots,an\}_{\downarrow_{\omega}X_j \rightarrow x_j\sigma\downarrow} \uparrow_{\omega}$ and let $\varphi_{p+1} = \varphi_p$. As $\sigma \in \text{csu}_{\text{R,AC}}(\{sk = tk\}_{k \in T(m)})$, we have
Case 2. Let $p \in \text{Var}(\varphi_p)$. We have $p \in R(p)$ and $\ell_p = \text{in}(c_p, x_p)$. Let $P_{p+1} = (a_{p+1}, \ldots, a_n)\{x_j \mapsto x_j \sigma \sigma' \omega \} \in R(p)$, and $f_p + 1 = \varphi_p$. As $p \in R(p)$, we have that 
\[ P \models (k_{f_p}, \varphi_p) \xrightarrow{L_p} (P_{p+1}, \varphi_{p+1}) \]
which is what we wanted to prove.

We have shown that $P \models (\tau_{f_p}, \varphi_p)$. 

**Case 2.** Let $m \in S(n)$, $\sigma \in \text{vars}(\varphi_{\sigma_1}, \ldots, \varphi_{\sigma_m})$. Let $f$ be the following statement:
\[ k_{f_1}, \ldots, k_{f_m} (f(Y_1, \ldots, Y_k), f(y_1, \ldots, y_k) \tau) \]

We assume that $P \models k_{f_1}, \ldots, k_{f_m} (Y_j, y_j \tau) \omega$ for all $1 \leq j \leq k$. We distinguish two cases:

1) if there exist $L_1, \ldots, L_m$ such that 
\[ (P_1, \varphi_1) \xrightarrow{L_1} (P_2, \varphi_2) \xrightarrow{L_2} \ldots \xrightarrow{L_m} (P_{m+1}, \varphi_{m+1}) \]
and $L_i \varphi_i \downarrow = \text{AC} \ell_i \sigma \tau \downarrow \omega$ for all $1 \leq i \leq m$, we have that 
\[ \varphi_{m+1} = \text{AC} (w(S(m))) = t_{\sigma \omega} = t_{m \sigma \tau \omega} \]
and we have that 
\[ \varphi_{m+1} \vdash w(S(m)) \quad \text{and therefore} \quad \varphi_{m+1} \vdash t_{m \sigma \tau \omega} \]
which implies that 
\[ P \models (k_{f_1}, \varphi_1, \ldots, k_{f_m} (w(S(m)), (t_{m \sigma \tau})) \omega). \]
2) otherwise, we trivially have that 
\[ P \models (k_{f_1}, \varphi_1, \ldots, k_{f_m} (f(Y_1, \ldots, Y_k), f(y_1, \ldots, y_k) \tau), \omega). \]

Hence, in both cases, we have shown that $P \models f$.

**Lemma 3.** Let $f$ be a ground process and $K$ be a set of statements such that for all $f \in K$ we have that $P \models f$. We have that $P \models f$ for all $f \in \mathcal{H}(K)$.

**Proof.** We prove this result by structural induction on the proof-tree witnessing the fact that $f \in \mathcal{H}(K)$. Let $\pi$ be such a proof tree.

**Base case.** The proof-tree $\pi$ is reduced to a leaf meaning that it has been obtained by applying the rule CONSEQ on a statement 
\[ f' = (H \leftarrow i) \] (i.e. with $n = 0$). We have that $f = f' \sigma$ where $\sigma$ is a substitution grounding for $f'$. Moreover, by hypothesis, we have that $P \models f'$. Hence, as all variables in $f'$ are universally quantified, we have also that $P \models f' \sigma$, i.e. $P \models f$.

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

- **CONSEQ:** We have that $f' = (H \leftarrow B_1 \ldots B_n) \in K$, $\sigma$ is a substitution grounding for $f'$ such that $f = H \sigma$, $B_i \sigma \in \mathcal{H}(K)$ for $1 \leq i \leq n$, and $\text{skel}(f' \sigma)$ is in normal form. As $H \leftarrow B_1 \ldots B_n \in K$ we have by hypothesis that $P \models H \leftarrow B_1 \ldots B_n$ and hence $P \models (H \leftarrow B_1 \ldots B_n) \sigma$. By induction hypothesis we also have that $P \models B_i \sigma$ for $1 \leq i \leq n$. Hence, we conclude that $P \models H \sigma$.

- **EXTEND:** We have $f = k_{uv}(R, t)$ with $k_{uv}(R, t) \in \mathcal{H}(K)$. By induction hypothesis, we have that $P \models k_{uv}(R, t)$, and it follows from the semantics of $k$ that $P \models k_{uv}(R, t)$.

This allows us to conclude.

**B. Completeness**

The completeness part of Theorem 1 is a consequence of the following lemma.

**Lemma 4:** Let $P$ be a ground process such that $(P, \emptyset) \xrightarrow{L_1 \ldots L_n} (Q, \varphi)$ then
We assume that the two statements hold for any index less than \( n \) and we prove them for \( n \). By hypothesis, we have that 
\[
E_1 (Q, \varphi), \quad \text{and therefore there exists } \omega \text{ such that:}
\]
\[
\begin{align*}
    & (L_1 \varphi \downarrow, \ldots, L_n \varphi \downarrow) =_{\text{AC}} (\ell_1, \ldots, \ell_n) \omega \downarrow, \text{ and} \\
    & s_k \omega \downarrow =_{\text{AC}} t_k \omega \downarrow \text{ for all } k \in T(n).
\end{align*}
\]

First, we prove statement (A). We have 
\[
s_k \omega \downarrow =_{\text{AC}} t_k \omega \downarrow
\]
for all \( k \in T(n) \). Therefore, there exists \( \sigma \in \text{vars}_{R, \text{AC}}(\{s_k \mid k \in T(n)\}) \) and a substitution \( \pi \) such that:
\[
\begin{align*}
    1) & \quad s_k \sigma \downarrow =_{\text{AC}} t_k \sigma \downarrow \text{ for all } k \in T(n) \text{; and} \\
    2) & \quad x \omega \downarrow =_{\text{AC}} x \sigma \pi \downarrow \text{ for any } x \in \text{vars}(\{s_k, t_k \mid k \in T(n)\}).
\end{align*}
\]

It follows that 
\[
(\ell_1, \ldots, \ell_n) \omega \downarrow =_{\text{AC}} (\ell_1, \ldots, \ell_n) \sigma \pi \downarrow.
\]
By definition of \( \text{vars}_{R, \text{AC}}((\ell_1, \ldots, \ell_n) \sigma) \), we know that there exists \( \tau \in \text{vars}_{R, \text{AC}}((\ell_1, \ldots, \ell_n) \sigma) \) such that
\[
(\ell_1, \ldots, \ell_n) \sigma \pi \downarrow =_{\text{AC}} (\ell_1, \ldots, \ell_n) \sigma \tau \downarrow \tau'
\]
for some substitution \( \tau' \). Let \( f \) be the following statement.

\[
r_{\ell_1 \sigma \tau \downarrow, \ldots, \ell_n \sigma \tau \downarrow} \leftarrow k_{\ell_1 \sigma \tau \downarrow, \ldots, \ell_n \sigma \tau \downarrow}((X_j, x_j \sigma \tau \downarrow)_{j \in J(n)}).
\]

By the definition of \( \text{seed}(P) \), we have that \( f \in \text{seed}(P) \). Let \( \tau'' \) be the substitution that extends \( \tau' \) by \( \{X_j \mapsto R_j\}_{j \in J(n)} \) where \( R_j \) are recipes for \( x_j \omega \). Applying our induction hypothesis, we have that
\[
k_{\ell_1 \sigma \tau \downarrow, \ldots, \ell_n \sigma \tau \downarrow}((X_j, x_j \sigma \tau \downarrow)_\omega) \in \mathcal{H}(\text{seed}(P)) \text{ for } j \in R(n).
\]

Therefore,
\[
r_{\ell_1 \sigma \tau \downarrow, \ldots, \ell_n \sigma \tau \downarrow} \in \mathcal{H}(\text{seed}(P)).
\]

We now prove statement (B). By structural induction on the recipe \( R \), we show that:
\[
k_{L_1 \varphi \downarrow, \ldots, L_n \varphi \downarrow}(R, R \varphi \downarrow) \in \mathcal{H}(\text{seed}(P))
\]
1) If \( R = c \) is a public name, and as \( f = \left(k_c(c, c) \leftarrow \right) \) is in the set of seed statements by definition, we have that \( k_c(R, R \varphi \downarrow) = k_c(c, c) \in \mathcal{H}(\text{seed}(T)) \), and therefore \( k_{L_1 \varphi \downarrow, \ldots, L_n \varphi \downarrow}(R, R \varphi \downarrow) \in \mathcal{H}(\text{seed}(T)) \) by the EXTEND rule.

2) If \( R = w_j \), let \( m \) be the smallest index such that
\[
|S(m)| = j \text{ (i.e. } m \text{ is the index of the action } a_m \text{ that outputs the content of } w_j) \text{ and let } t_m \text{ be the term such that } a_m = \text{out}(c, t_m) \text{ for some channel } c.
\]

Let \( \sigma \in \text{vars}_{R, \text{AC}}(\{s_k \mid k \in T(m)\}) \) be such that
\[
(\ell_1, \ldots, \ell_n) \omega \downarrow =_{\text{AC}} (\ell_1, \ldots, \ell_n) \sigma \downarrow
\]
for some \( \pi \). Let \( \tau \in \text{vars}_{R, \text{AC}}((\ell_1, \ldots, \ell_m, t_m) \sigma) \) and \( \tau' \) be a substitution such that
\[
(\ell_1, \ldots, \ell_m, t_m) \omega \downarrow =_{\text{AC}} (\ell_1, \ldots, \ell_m, t_m) \sigma \tau \downarrow \tau'.
\]
Let \( h \) be the following statement:

\[
k_{\ell_1 \sigma \tau \downarrow, \ldots, \ell_m \sigma \tau \downarrow}(w_j, t_m \sigma \tau \downarrow) \leftarrow \{k_{\ell_1 \sigma \tau \downarrow, \ldots, \ell_{k-1} \sigma \tau \downarrow}(X_k, x_k \sigma \tau \downarrow)\}_{k \in R(m)}.
\]

By definition of \( \text{seed}(P) \), we have that \( h \in \text{seed}(P) \). For \( k \in R(m) \), let \( R_k \) be recipes of \( x_k \sigma \tau \downarrow =_{\text{AC}} x_k \omega \downarrow \) in the smallest possible prefix of \( \varphi \). Let
\[
\tau'' = \tau' \cup \{X_k \mapsto R_k\}_{k \in R(m)}.
\]

We have that the antecedents of \( h \tau'' \) are in \( \mathcal{H}(\text{seed}(T)) \) by the induction hypothesis. Therefore
\[
k_{\ell_1 \sigma \tau \downarrow, \ldots, \ell_m \sigma \tau \downarrow}(w_j, t_m \sigma \tau \downarrow \tau'')
\]
\[
=_{\text{AC}} k_{\ell_1 \sigma \tau \downarrow, \ldots, \ell_m \sigma \tau \downarrow}(w_j, t_m \sigma \tau \downarrow \tau')
\]
\[
=_{\text{AC}} k_{\ell_1 \omega \downarrow, \ldots, \ell_m \omega \downarrow}(w_j, t_m \omega \downarrow) \in \mathcal{H}(\text{seed}(P)).
\]

But \( (\ell_1, \ldots, \ell_m) \omega \downarrow \) is a prefix of \( (\ell_1, \ldots, \ell_n) \omega \downarrow \) and by the EXTEND rule, \( k_{\ell_1 \omega \downarrow, \ldots, \ell_m \omega \downarrow}(w_j, t_m \omega \downarrow) \in \mathcal{H}(\text{seed}(T)) \), i.e. \( k_{L_1 \varphi \downarrow, \ldots, L_n \varphi \downarrow}(R, R \varphi \downarrow) \in \mathcal{H}(\text{seed}(T)) \), which is what we had to prove.

3) If \( R = f(R_1, \ldots, R_k) \), let \( \tau \in \text{vars}_{R, \text{AC}}(f(y_1, \ldots, y_k)) \) and \( \tau' \) be such that \( R \varphi \downarrow =_{\text{AC}} f(y_1, \ldots, y_k) \tau \downarrow \tau' \) and \( R_i \varphi \downarrow =_{\text{AC}} y_i \tau \downarrow \tau' \) for \( 1 \leq i \leq k \). Let \( g \) be the following statement:

\[
k_{\ell_1, \ldots, \ell_k}(f(Y_1, \ldots, Y_k), f(y_1, \ldots, y_k) \tau \downarrow) \leftarrow \{k_{\ell_1, \ldots, \ell_n}(Y_j, y_j \tau \downarrow)\}_{j \in \{1, \ldots, k\}}.
\]

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

This allows us to conclude the proof. □

APPENDIX

PROOFS REGARDING SATURATION

We show that the update preserved knowledge bases, and prove Theorem 2.

Proposition 1: If \( K \) is a knowledge base and \( f \) is a statement then \( K \cup f \) is a knowledge base.

Proof. A knowledge base can contain any statement except solved deduction statements \( h = \left(k(R, x) \leftarrow B_1, \ldots, B_n\right) \)

Such kind of statement is not in a canonical form due to the SHIFT rule, therefore such statement is never added to the knowledge base. □
A. Soundness

In this section, we prove the soundness part of Theorem 2 (i.e. item 1). The saturation procedure is a restriction of the original procedure (as described in [18]), therefore the proof is essentially the same as in [18], and is actually an immediate consequence of the lemmas stated and proved below.

Lemma 5: Let $P$ be a ground process and $f$ be a statement. If $P \models f$ then $P \models f \downarrow$. 

Proof. Consider a statement $f$.

1) First, we have to normalize all the recipes that occur in $f$.

This will have only an effect on $H$, and the resulting statement $f'$ is such that $P \models f'$ since $\varphi \vdash_{T} t$ implies $\varphi \vdash_{T'} t$ when $T' = T \downarrow$.

2) Second, we apply the REMOVE rule as many times as possible. Let us show that applying one instance of such a rule leads to a statement $f''$ such that $P \models f''$. In such a case, we have that:

- $f = H \models k_{uv}(X,t), B_{1}, \ldots, B_{n} \parallel M$;
- $f'' = H \models k_{uv}(Y,t), B_{1}, \ldots, B_{n} \parallel (M \setminus k_{uv}(X,t))$.

Moreover, we know that $X \notin \text{vars}(H)$. Let $\tau$ be an arbitrary substitution such that $T = k_{uv}(Y,t)\tau, P = B_{1}\tau, \ldots, P = B_{n}\tau$. We consider the substitution $\tau'$ to be identical to $\tau$ except that $\tau'(X) = \tau(Y)$. We have that $P \models k_{uv}(X,t)\tau$ as well. Therefore, we have that $P \models H\tau$, and this allows us to conclude that $P \models f''$.

3) Third, we apply the SHIFT rule as many times as possible when the statement $f$ is solved. Let us show that applying one instance of such a rule leads to a statement $f'''$ such that $P \models f'''$. In such a case, we have that:

- $f = k_{uv}(R,t) \models k_{uv}(X,x), B_{1}, \ldots, B_{n}$;
- $f''' = k_{uv}(R \oplus X, t \oplus x)^{T} \models k_{uv}(X,x), B_{1}, \ldots, B_{n}$.

Moreover, we know that $x \in \text{factor}(t)$. Let $\tau$ be an arbitrary substitution such that $P \models k_{uv}(X,x)\tau, P \models B_{1}\tau, \ldots, P \models B_{n}\tau$.

Therefore, we have $P \models k_{uv}(R,t)\tau$, and thus we have $P \models k_{uv}(R \oplus X, t \oplus x)\tau$, and $P \models k_{uv}(R \oplus X, t \oplus x)\tau$. This allows us to conclude that $P \models f'''$.

We have shown that all the rules needed for computing the canonical form are sound and therefore $T \models f \downarrow$ whenever $T \models f$.

Lemma 6: Let $P$ be a ground process and $K$ be a knowledge base. If $P \models f$ for all $f \in K_{\text{conseq}}$ then we have that $P \models f$ for all $f \in \text{conseq}(K_{\text{conseq}})$.

Proof. We show that both inference rules are sound.

Case: AXIOM. Immediate from the semantics of $k$.

Case: RES. We consider $f = k_{u}(R,t) \models B_{1}, \ldots, B_{n}$ and $g_{i} = B_{i}\sigma \models C_{1}, \ldots, C_{m}$ for $1 \leq i \leq n$ be statements such that $P \models f$ and $P \models g_{i}$ with $1 \leq i \leq n$. We have to show that

$$P \models (k_{u}(R,t)\sigma \models C_{1}, \ldots, C_{m})$$

Let $\tau$ be a substitution such that $P \models C_{1}\tau, \ldots, C_{m}\tau$. We have that $P \models C_{1}\tau, \ldots, C_{m}\tau$ and $P \models g_{i}$ with $1 \leq i \leq n$. Therefore, we have that $P \models B_{i}\tau$ with $1 \leq i \leq n$. Since $P \models f$, we have that $P \models k_{u}(R,t)\sigma \tau$ as well. This allows us to conclude that $P \models k_{uv}(R,t)\sigma\tau$.

Now, we establish the soundness of each saturation rule as defined in Figure 6.

Lemma 7: Let $P$ be a ground process, $f$, $g$, and $h$ be three statements, and $\sigma$ be a substitution defined as in the RESOLUTION rule. If $P \models f$ and $P \models g$ then $P \models h\sigma$.

Proof. We consider the following statements:

$$f = H \models k_{uv}(X,t), B_{1}, \ldots, B_{n} \parallel M$$
$$g = k_{uv}(R,t') \models B_{n+1}, \ldots, B_{m}$$
$$h = H \models B_{1}, \ldots, B_{m} \parallel M \setminus \{k_{uv}(X,t)\}$$

Moreover, we have that $k_{uv}(X,t) \in \text{sel}(f)$, $g$ is a solved statement but not in $f_{\sigma}'$, and $\sigma \in \text{csuAC}(k_{uv}(X,t), k_{uv}(R,t'))$.

Let $\tau$ be a substitution grounding for $h\sigma$ and assume that $P \models B_{1}\tau, \ldots, B_{m}\tau$. We will show that $P \models H\sigma\tau$.

As $P \models B_{n+1}\tau, \ldots, B_{m}\tau$ and because $P \models g$, we have that $P \models k_{uv}(R,t')\sigma\tau$. But $k_{uv}(R,t')\sigma\tau = k_{uv}(X,t)\sigma\tau$, and therefore we have that $P \models k_{uv}(X,t)\sigma\tau$ as well. As all antecedents of $f\sigma\tau$ are true in $P$ and because $P \models f$, we have that $P \models H\sigma\tau$. As $\tau$ was chosen arbitrarily, it follows that $P \models h\sigma$.

Lemma 8: Let $P$ be a ground process, $f$, $g$, and $h$ be three statements, and $\sigma$ be a substitution defined as in the RESOLUTION+ rule. If $P \models f$ and $P \models g$ then $P \models h\sigma$.

Proof. The proof is similar to the former one.

Lemma 9: Let $P$ be a ground process, $f$, $g$, and $h$ be three statements, and $\sigma$ be a substitution defined as in the EQUATION rule. If $P \models f$ and $P \models g$ then $P \models h\sigma$.

Proof. We consider the following statements:

$$f = k_{u}(R,t) \models B_{1}, \ldots, B_{n}$$
$$g = k_{u'}(R',t') \models B_{n+1}, \ldots, B_{m}$$
$$h = i_{u'}(R',R') \models B_{1}, \ldots, B_{m}$$

where $\sigma \in \text{csuAC}((u,t),(u',t'))$.

Let $\tau$ be a substitution grounding for $h\sigma$. We assume that $P \models B_{1}\sigma\tau, \ldots, B_{m}\sigma\tau$. We will show that $P \models i_{u'}(R,R')\sigma\tau$.

As $P \models B_{1}\sigma\tau, \ldots, B_{n}\sigma\tau$ and because $P \models f$ we have that $P \models k_{u}(R,t)\sigma\tau$. But $k_{u}(R,t)\sigma\tau = k_{u'}(R',t')\sigma\tau$ by choice of $\sigma$, and therefore $P \models k_{u'}(R',t')\sigma\tau$, and we also have that $P \models k_{u'}(R',t')\sigma\tau$. As $P \models B_{n+1}\sigma\tau, \ldots, B_{m}\sigma\tau$ and because $P \models g$ we also obtain that $P \models k_{u'}(R',t')\sigma\tau$.

As $P \models k_{u'}(R',t')\sigma\tau$ and $P \models k_{u'}(R',t')\sigma\tau$, we have by definition that $P \models i_{u'}(R,R')\sigma\tau$. We have shown that the head of $h\sigma\tau$ is true in $P$. As $\tau$ was chosen arbitrarily, it follows that $h\sigma$ holds in $P$.

Lemma 10: Let $P$ be a ground process, $f,g$, and $h$ be three statements, and $\sigma$ be a substitution defined as in the TEST rule. If $P \models f$ and $P \models g$ then $P \models h\sigma$.
Proof. We consider the following statements:
\[
\begin{align*}
    f &= i_u(R, R') \iff B_1, \ldots, B_n \\
    g &= r_{u'}(u) \iff B_{n+1}, \ldots, B_m \\
    h &= r_{u''}(R, R') \iff B_1, \ldots, B_m
\end{align*}
\]
where \( \sigma \in \text{CSILAC}(u, u') \).

Let \( \sigma \) be a substitution grounding for \( h \sigma \). We assume that
\( P \models B_1 \sigma, \ldots, B_m \sigma \) and we show that
\( P \models i_{u''}(R, R') \sigma \). Indeed, as
\( P \models B_1 \sigma, \ldots, B_m \sigma \) and as \( P \models f \), we have
that \( P \models i_u(R, R') \sigma \). As \( P \models B_{n+1} \sigma, \ldots, B_m \sigma \) and
\( P \models g \), we have that \( P \models r_{u'} \sigma \).

But \( \sigma \in \text{CSILAC}(u, u') \) and therefore \( u' \sigma = \sigma \). Hence,
we immediately obtain \( P \models r_{u''}(R, R') \sigma \), which is what
we wanted. It follows that \( P \models h \sigma \). \( \square \)

Now, we establish the soundness of our update operation.

Lemma 11: Let \( P \) be a ground process, \( K \) be a knowledge base,
and \( g \) be a statement. If \( P \models g \), and \( P \models f \) for all
\( f \in K \), then \( P \models g \) for any \( f \in (K \cup g) \).

Proof. If \( K \cup g \subseteq K \cup \{g\} \), we immediately conclude by
Lemma 5. Otherwise, we have that \( K \cup g = K \cup \{h\} \) where:
\[
\begin{align*}
    g &= k_u(R, t) \iff k_u(X_1, x_1) \ldots k_u(X_n, x_n) \\
    h &= i_u(R, R') \iff i_u(X_1, x_1) \ldots i_u(X_n, x_n)
\end{align*}
\]
Moreover, we have that \( g' \in \text{conseq}(K_{\text{solved}}) \) where
\[
g' = k_u(R', t) \iff k_u(X_1, x_1) \ldots k_u(X_n, x_n).
\]

In order to conclude, we have to show that \( P \models h \). Let \( \tau \)
be a substitution grounding for \( h \) such that the antecedents of
\( h \tau \) are true in \( P \). As the antecedents of \( h \tau \) are the same as the
antecedents of \( g' \tau \) and those of \( g' \), and as \( P \models g \)
and \( P \models f' \) (thanks to Lemma 6), we have that \( P \models k_u(R, t) \tau \), \( P \models
k_u(R', t) \tau \), and \( P \models i_u(R, R') \tau \) as well. But this immediately
implies that \( P \models i_u(R, R') \tau \) (the head of \( h \tau \)). As \( \tau \)
was chosen arbitrarily, it follows that \( P \models h \). \( \square \)

Lemma 12: Let \( P \) be a ground process and \( K \) be a knowledge base
such that \( P \models f \) for all \( f \in K \). We have that
\( P \models H \) for all \( H \in \mathcal{H}_e(K) \).

Proof. This result is proved by structural induction on the
derivation tree witnessing the fact that \( H \in \mathcal{H}_e(K) \). \( \square \)

B. Completeness

In this section we prove the completeness part of Theorem 2.

The two first items are immediate consequences of Theorem 1
and Lemma 27 proved below. The third item follows from
the second item of Theorem 2 and Lemma 25 proved below.

1) Preliminaries: We start by introducing a variant of \( \mathcal{H} \),
called \( \mathcal{H}' \), where uses of \( f_0^+ \) clauses is constrained. Derivations
establishing \( H \in \mathcal{H}'(K) \) (for some ground fact \( H \))
are essentially derivations of \( H \in \mathcal{H}(K) \), decorated with \( + \) and
\( - \) annotations. In addition to it, we impose that EXTEND rules
are only applied in \( \mathcal{H}' \) derivations, and never repeatedly; this
is done through \( s \) and \( e \) annotations.

Definition 16: Given a solved knowledge base \( K \), we define
\( \mathcal{H}'(K) = \mathcal{H}'_L(K) \cup \mathcal{H}'^e(K) \) and \( \mathcal{H}'(K) = \mathcal{H}'_L(K) \cup \mathcal{H}'^e(K) \),
and
\[
\begin{align*}
    f &= (H \iff B_1, \ldots, B_n) \in K \setminus f_0^+
\\
    H \sigma &\in \mathcal{H}'(K)
\\
    f &= (H \iff B_1, B_2) \in f_0^+
\\
    B_1 \sigma, B_2 \sigma &\in \mathcal{H}'(K) \text{ with skel}(f \sigma) \text{ in normal form}
\\
    H \sigma &\in \mathcal{H}'^e(K)
\\
    \text{EXTEND} &\in \mathcal{H}'(K)
\end{align*}
\]

The sets of derivations associated to \( \mathcal{H}(K) \), \( \mathcal{H}'(K) \), \( \mathcal{H}'_L(K) \),
\( \mathcal{H}'^e(K) \) and \( \mathcal{H}'(K) \) are respectively denoted by
\( \Pi(K) \), \( \Pi'(K) \), \( \Pi'_L(K) \), \( \Pi^e(K) \) and \( \Pi'(K) \).

We consider a solved knowledge base \( K \), and we introduce the
following notation:

1) the extraction \( \text{extract}(\pi) \) of a derivation tree \( \pi \in \Pi'(K) \)
is the set of derivation trees \( \{\pi_1\} \) in case \( \pi \in \Pi'_L(K) \); and
\( \text{extract}(\pi_1) \cup \text{extract}(\pi_2) \) where \( \pi_1 \) and \( \pi_2 \) are
the immediate sub-derivation trees of \( \pi \) in case \( \pi \in \Pi^e(K) \).

2) the size \( S(\pi) \) of a derivation tree \( \pi \in \Pi'(K) \)
is the number of nodes occurring in \( \pi \) excluding EXTEND nodes.

3) the real world \( \mathcal{W}(\pi) \) of a derivation tree \( \pi \in \Pi'(K) \)
whose root is labeled with \( k_u(R, t) \) is \( w \) in case \( \pi \notin \Pi^e(K) \), and
\( \mathcal{W}(\pi) = \mathcal{W}(\pi') \) where \( \pi' \) is the immediate
sub-derivation tree of \( \pi \) otherwise.

4) the \( l \)-restricted size \( S_l(\pi) \) of a derivation tree \( \pi \in \Pi'(K) \)
whose immediate sub-derivation tree are \( \pi_1, \ldots, \pi_n \) is
\[
\sum_{1 \leq i \leq n} \{S(\pi_i) \mid \mathcal{W}(\pi_i) \geq l \}
\]

Such a measure allows us to define the set \( \mathcal{H}^{l,\kappa}(K) \)
with \( l, \kappa \in \mathbb{N} \) as the smallest set of facts that:
\[
\begin{align*}
    &\text{contains any } H \in \mathcal{H}(K) \text{ which has a derivation tree } \\
    \pi \in \Pi'(K) \text{ such that } S(\pi) \leq \kappa; \text{ and } \\
    &\text{is closed under the same rules as } \mathcal{H}_e.
\end{align*}
\]

Remark 1: The \( l \)-restricted size information in the following
proofs is not used to prove the completeness of the saturation
but to prove the correction of the algorithm (Theorem 3).

So for a first reading of the proof, it is safe to skip these
information and to read \( \mathcal{H}_e(K) \) instead of \( \mathcal{H}^{l,\kappa}(K) \).

We start the proofs with a technical lemma which will allow
us to exploit the flexibility in how a sum can be obtained.

Lemma 13: Let \( K \) be a solved knowledge base, and \( \pi \in \Pi'_L(K) \)
with its root labeled with \( k_u(R, \bigoplus_{i=1}^n t_i) \) (and no \( t_i \)
is a sum). Let \( k \in \{1, \ldots, n\} \). There exists a partition of
\{1, \ldots, n\} in A, B, and a derivation tree $\pi' \in \Pi_\nu(K)$ with its root labeled with $k_0(R', \bigoplus_{i=1}^n t_i)$ such that:

- $k \in A$;
- $R = aB R'$;
- $S(\pi') = S(\pi)$;
- the immediate sub-derivation trees of $\pi'$ are:
  - $\pi'_1 \in \Pi_\nu(K)$ with root $h'_1 = k_0(R_A, \bigoplus_{i \in A} t_i)$; and
  - $\pi'_2 \in \Pi_\nu(K)$ with root $h'_2 = k(R_B, \bigoplus_{i \in B} t_i)$.

**Proof.** We proceed by induction on $S(\pi)$. Let $\pi_1$ and $\pi_2$ be the immediate sub-derivations of $\pi$, respectively labeled with $h_1$ and $h_2$. We may assume w.l.o.g. that $t_k$ occurs in $h_1$. If $\pi_1 \in \Pi_\nu(K)$, we conclude immediately. We now consider that $\pi_1 \in \Pi_\nu(K)$. We apply our induction hypothesis on $\pi_1$ to get $\pi_a \in \Pi_\nu(K)$ and $\pi_b \in \Pi_\nu(K)$ labeled respectively with $h_a$ and $h_b$, and such that $S(\pi_a) + S(\pi_b) + 1 = S(\pi_1)$. We apply $\mathit{SUM}$ on $\pi_a$ and $\pi_b$ to obtain $\pi_3$. We can conclude with $\pi'_1$ being $\pi_a$, $\pi'_2$ being $\pi_3$, and $\pi'$ the result of applying $\mathit{SUM}$ on $\pi'_1$ and $\pi'_2$. We can check that $S(\pi') = S(\pi'_1) + S(\pi'_2) + 1 = S(\pi)$.

Now, after a simple observation on $\mathit{EXTEND}$ (Proposition 2), we establish that $\mathcal{H}$ and $\mathcal{H}'$ coincide (Lemma 14).

**Lemma 14:** Let $K$ be a solved knowledge base such that $f_0^+ \subseteq K$. If $k_u(R, t) \in \mathcal{H}(K)$, then $k_u(R, t) \in \mathcal{H'}(K)$ for any $v$.

**Proof.** We proceed by induction on the derivation $\pi$ establishing that $k_u(R, t) \in \mathcal{H'}(K)$. We distinguish three cases:

1. Case $\pi \in \Pi_\nu(K)$. In such a case, we conclude by applying rule $\mathit{EXTEND}'$ to extend the world with $v$.
2. Case $\pi \in \Pi^\mathit{skel}(K)$. In such a case, its last rule is already $\mathit{EXTEND}'$, extending some world $u_1$ to $u_1 u_2 = u$; we conclude by modifying this last rule to directly extend $u_1$ with $u_2 v$.
3. Case $\pi \in \Pi^\mathit{e}(K)$. In such a case, its two premises are of the form $k_u(R_i, t_i)$ with $i \in \{1, 2\}$. By induction hypothesis, we have that $k_u(R_i, t_i) \in \mathcal{H}(K)$ for $i \in \{1, 2\}$, which allows us to conclude using rule $\mathit{SUM}$.

**Proposition 2:** Let $K$ be a solved knowledge base such that $f_0^+ \subseteq K$. We have that $\mathcal{H}(K) = \mathcal{H'}(K)$.

**Proof.** The fact that $\mathcal{H}'(K) \subseteq \mathcal{H}(K)$ is immediate. For the other direction, we have to show that any derivation tree $\pi \in \Pi(K)$ can be turned into a derivation tree in $\Pi'(K)$ with the same conclusion. Let $\pi$ be a derivation tree in $\mathcal{H}(K)$ with root labeled with $H$. We proceed by case analysis on the last rule of $\pi$.

1. Case $\pi$ ends with $\mathit{EXTEND}$. In such a case, we apply the induction hypothesis to its immediate sub-derivation, and conclude by $\mathit{Proposition 2}$.
2. Case $\pi$ ends with $\mathit{CONSEQ}$. In such a case, let $f$ be the statement and $B_1, \ldots, B_n$ the premisses involved in this last step. Applying our induction hypothesis, we know that there are derivation tree $\pi'_1, \ldots, \pi'_n \in \Pi'(K)$ labeled respectively with $B_1, \ldots, B_n$. We then distinguish two cases:

- If $f \notin f_0^+$ we can use $\mathit{STD}$ on $\pi'_1, \ldots, \pi'_n$ to obtain $H \in \mathcal{H}'(K)$.
- If $f \in f_0^+$, we proceed similarly but using rule $\mathit{SUM}$ instead of $\mathit{STD}$.

**Proposition 3:** Let $K$ be a solved knowledge base, and let $f = (k_w(R, t) \equiv c_1, \ldots, c_n)$ a statement such that $f \in \mathit{conseq}(K)$ and $\tau$ be a substitution grounding for $f$ such that $\mathit{skel}(f \tau)$ is in normal form, and for all $1 \leq i \leq n$ we have that $C_i \tau$ has a derivation tree $\pi_i \in \mathcal{H}(K)$. Then $k_w(R, t) \tau$ has a derivation tree $\pi \in \mathcal{H}(K)$.

**Proof.** By induction on the proof tree of $f \in \mathit{conseq}(K)$.

- If the $\mathit{AXIOM}$ rule was used, we have that $C_i = k_w(R, t)$ for some $i$ and, by hypothesis, $C_i \tau \in \mathcal{H}(K)$. Using the $\mathit{EXTEND}$ rule we conclude.
- If the $\mathit{RES}$ rule was used, we have that there exists a solved statement $g = (k_w(R', t') \equiv B_1, \ldots, B_n) \in K$ and a substitution $\sigma$ such that $k_w(R_i, t) = k_w(R_i', t') \sigma$ and $B_i \sigma \equiv C_i, \ldots, C_m \in \mathit{conseq}(K)$ ($1 \leq i \leq n$). We assume w.l.o.g. that $\sigma$ only instantiates first-order variables that occur in $u'$ and $t'$. Therefore we have that $\mathit{skel}(B_i \sigma)$ is in normal form as only subterm of $u' \sigma(= u)$ and $t' \sigma(= t)$ may occur in it, $u, t$ are in normal form. Since $\mathit{skel}(B_i \sigma)$ is in normal form, we can apply our induction hypothesis, and conclude that $B_i \sigma \in \mathcal{H}(K)$.

As $(k_w(R', t') \equiv B_1, \ldots, B_n) \in K$ and $\mathit{skel}(B_i \sigma)$ are all in normal form, it follows that $k_w(R', t') \sigma = k_w(R, t) \tau \in \mathcal{H}(K)$ with derivation tree $\pi$. Using the $\mathit{EXTEND}$ rule we conclude.

We continue showing several useful properties on identity formulas in $\mathcal{H}_e(K)$.

**Lemma 15:** Let $K$ be a solved knowledge base.

1. If $i_w(R_1, R_2 \oplus R_3) \in \mathcal{H}_e^{(1, e)}(K)$ then $i_w(R_1 \oplus R_2, R_3) \in \mathcal{H}_e^{(1, e)}(K)$.
2. If $i_w(R_1, R_2) \in \mathcal{H}_e^{(1, e)}(K)$ and $i_w(R_2, R_3) \in \mathcal{H}_e^{(1, e)}(K)$ then $i_w(R_1, R_3) \in \mathcal{H}_e^{(1, e)}(K)$.
3. If $i_w(R_1, R_2) \in \mathcal{H}_e^{(1, e)}(K)$ then $i_w(R_2, R_1) \in \mathcal{H}_e^{(1, e)}(K)$.

**Proof.** We prove the three items separately.

1. We apply $\mathit{CONG}$ using $i_w(R_2, R_2) \in \mathcal{H}_e^{(1, e)}(K)$ (obtained with $\mathit{REFL}$) to get $i_w(R_1 \oplus R_2, R_2 \oplus R_3) \in \mathcal{H}_e^{(1, e)}(K)$, and conclude using $\mathit{MOD-I}$.
2. Using $\mathit{CONG}$, we get $i_w(R_1 \oplus R_2, R_2 \oplus R_3) \in \mathcal{H}_e^{(1, e)}(K)$.

We apply $\mathit{CONG}$ again using $i_w(R_2, R_2) \in \mathcal{H}_e^{(1, e)}(K)$ (obtained with Refl), and conclude using $\mathit{MOD-I}$.
3. We apply twice $\mathit{CONG}$ using $i_w(R_1, R_1) \in \mathcal{H}_e^{(1, e)}(K)$ and $i_w(R_2, R_2) \in \mathcal{H}_e^{(1, e)}(K)$ (obtained with $\mathit{REFL}$) to get $i_w(R_1 \oplus R_1 \oplus R_2, R_2 \oplus R_1 \oplus R_2) \in \mathcal{H}_e^{(1, e)}(K)$, and conclude using $\mathit{MOD-I}$.

**Proposition 4:** Let $K$ be a knowledge base. If $k_w(R[T]_{\rho}, t) \in \mathcal{H}_e(K)$ and $i_w(T, T') \in \mathcal{H}_e(K)$, then $k_w(R[T']_{\rho}, t) \in \mathcal{H}_e(K)$.
Proof. By induction on the context $R$.

Base case. As $k_w(R, t) \in \mathcal{H}_e(K)$, it follows that there exist $R''$ such that
\begin{equation}
k_w(R'', t) \in \mathcal{H}(K) \tag{1}\end{equation}
and such that $i_w(R, R'') \in \mathcal{H}_e(K)$. But $i_w(R, R'') \in \mathcal{H}_e(K)$ and therefore, by Lemma 15 we have that
\begin{equation}i_w(R'', R') \in \mathcal{H}_e(K). \tag{2}\end{equation}

Using Equations 1 and 2 we immediately obtain by the definition of $\mathcal{H}_e$ that $k_w(R', t) \in \mathcal{H}_e(K)$.

Induction case. By application of the CONG rule on the top symbol.

Definition 17: We write $w \subseteq w'$ whenever $w$ is a prefix of $w'$: i.e. there exists $\ell_1, \ldots, \ell_n$ such that $w' = w_1 \cdots w_n$ and $w = w_1' \cdots w_n'$ for some $0 \leq m \leq n$.

2) Dealing with saturation: The following Lemma is a technical one which will be used in the proof of Lemma 17 and of Lemma 21.

Lemma 16: Let $K$ be a saturated knowledge base, and $f$ be an unsolved statement in $K$ such that:
$$f = \left(H \Leftarrow B_1, \ldots, B_n \parallel M \right)$$
and $B_i = k_w(X_i, t_i)$. Let $\sigma$ be a substitution such that
- $\sigma$ is grounding for $f$,
- $\text{skel}(f\sigma)$ is in normal form, and
- for all $1 \leq i \leq n$ there exist $\pi_i \in \Pi(K_{\text{Solved}})$ to derive $B_i\sigma$ and if $B_i \in M$ then $\pi_i \in \Pi_{\text{Marked}}(K_{\text{Solved}})$.

Then we have that, during the saturation of $K$, there was an update with a statement $h = \left(H' \Leftarrow B'_1, \ldots, B'_n \parallel M' \right)$ where $B'_i = k_w(X'_i, t'_i)$ and there is a substitution $\omega$ grounding for $h$ such that:
1) $\text{skel}(h\omega)$ is in normal form;
2) $H' \omega = H\sigma$;
3) for all $1 \leq i \leq n'$, there exist $\pi'_i \in \Pi(K_{\text{Solved}})$ to derive $B'_i\omega$, and if $B'_i \in \text{Marked'}$ then $\pi'_i \in \Pi_{\text{Marked'}}(K_{\text{Solved}})$;
4) $\sum_{i \in \{1, \ldots, n\}} S(\pi_i) > \sum_{i \in \{1, \ldots, n'\}} S(\pi'_i)$;
5) $\exists \pi \in \Pi(K_{\text{Solved}})$ be the derivation tree of $H\sigma \in \mathcal{H}(K)$ with sub-derivation trees $\pi_1, \ldots, \pi_n$ and $\pi' \in \Pi(K_{\text{Solved}})$ be the derivation tree of $H'\omega \in \mathcal{H}(K)$ whose sub-derivation trees are $\pi'_1, \ldots, \pi'_n$. For all $i$ we have that $\tilde{S}(\tilde{\pi}) \leq \tilde{S}(\tilde{\pi}')$.

Proof. Let $B_j = k_w(X_j, t_j) = \text{skel}(f)$.

Case: $\pi_j \in \Pi_{\text{Marked'}}(K_{\text{Solved}})$. We consider the last node of $\pi_j$ or the previous node if the last node is EXTEND' with some word $w'$. Let
$$g = \left(k_w'(R', t') \Leftarrow B_{n+1}, \ldots, B_m \right) \in K_{\text{Solved}}.$$ 
$g \neq f'_0 \sigma' \sigma''$ and the statement and substitution of this node (we have $w'w''\sigma' = w_j\sigma$). Note that by definition of a derivation tree and $S(\pi)$, we have that $S(\pi_j) = 1 + \sum_{i \in \{n+1, \ldots, m\}} S(\pi_i)$ where $\pi_i$ are the immediate sub-derivation tree of $\pi_j$ and also proofs of $(B_i\sigma) \in \Pi(K_{\text{Solved}})$.

As $\sigma' \sigma''$ is a unifier of $k_{w_j}(X_j, t_j)$ and $k_{w'(R', t')}$, it follows that $\tau \in \text{csu}_{AC}(k_{w_j}(X_j, t_j), k_{w'(R', t')})$ exists. Therefore $\sigma \sigma' \sigma''$ must be an instance of the one element of the complete set of unifier, let $\omega$ be a substitution such that $\sigma \sigma' \omega = \tau \omega$.

As $K$ has been saturated by selecting $B_j$ and $g \neq f'_0 \sigma'$, it follows that the RESOLUTION saturation rule was applied to $f$ and $g$. We consider $h$ the resulting statement from which $K$ has been updated:
$$h = \left(H \Leftarrow B_1, \ldots, B_{j-1}, B_{j+1}, \ldots, B_m \parallel M \right).$$

We have that $\omega$ is a substitution grounding for $h$, that $\text{skel}(h\omega)$ is in normal form, that $\pi'_1 \in \Pi(K_{\text{Solved}})$ is a derivation tree for $B_i\tau\omega \in \mathcal{H}(K_{\text{Solved}})$ for $i \in \{1, \ldots, j-1, j+1, \ldots, m\}$, and that
$$\sum_{i \in \{n+1, \ldots, m\}} S(\pi_i) < \sum_{i \in \{1, \ldots, n\}} S(\pi'_i).$$

Therefore, Condition 4 is satisfied. Similarly, Condition 5 is also satisfied: when $\mathcal{W}(\pi_i) \geq 1$ we have a strict inequality otherwise we have an equality as for $i \in \{n+1, \ldots, m\}$, $\mathcal{W}(\pi_i) < 1$.

Finally, the atoms $B_i$ for $i \in \{n+1, \ldots, m\}$ are not marked ($g$ is a solved statement), while, by hypothesis, the others ($B_i$ for $i \in \{1, \ldots, j-1, j+1, \ldots, n\}$) are marked only if they have a derivation tree in $\Pi_{\text{Marked'}}(K_{\text{Solved}})$. Therefore, Condition 3 is also satisfied.

Case $\pi_j \in \Pi_{\text{Marked'}}(K_{\text{Solved}})$. The case where rigid$(t_j) = \perp$ is similar to the previous one (since there is no marking). Otherwise, we apply Lemma 13 on $\pi_j$ and the factor rigid$(t_j)$ (the one chosen during the saturation) to get $k_w(R_x, t_x)$ of derivation tree $\pi_x \in \Pi_{\text{Marked'}}(K_{\text{Solved}})$ and $k_w(R_y, t_y)$ of derivation tree $\pi_y \in \Pi(\text{Solved})$ such that $k_w(R_x \oplus R_y, t_x \oplus t_y) = k_w(R_j, t_j)\sigma$ and rigid$(t_j)$ is part of $t_x\sigma$. We also have $S(\pi_j) = 1 + \pi_j + \pi_y$.

Let $\sigma^+ \in \text{csu}_{AC}(k_w(R_j, t_j), k_w(X \oplus Y, x \oplus y))$ such that $t_x$ is an instance of $x\sigma^+, t_y$ is an instance of $y\sigma^+, R_x$ an instance of $X\sigma^+$ and $R_y$ an instance of $Y\sigma^+$. We denote by $\omega$ the substitution such that $\sigma^+ \omega = \sigma$.

As $K$ has been saturated by selecting $B_j$, it follows that the RESOLUTION+ saturation rule was applied to $f$ and $f'_0 \sigma^+$ with $\sigma^+$ as unifier.

We consider $h'$ the resulting statement from which $K$ has been updated:
$$\left(H \Leftarrow B_x, B_y, B_1, \ldots, B_{j-1}, B_{j+1}, \ldots, B_m \parallel M \cup \{B_x\}\right) \sigma^+$$
where $B_x$ is marked, $B_y$ is not, and the other keep their status.

We have that $\omega$ is a substitution grounding for $h$ such that skel$(h\omega)$ is in normal form, that $B_0\sigma^+\omega$ has a derivation tree $\pi_x \in \Pi(\text{Solved})$ for $i \in \{1, \ldots, j-1, j+1, \ldots, m\}$ and that $S(\pi_x) + S(\pi_y) + \sum_{i \in \{1, \ldots, j-1, j+1, \ldots, m\}} S(\pi_i) < \sum_{i \in \{1, \ldots, n\}} S(\pi_i)$. Therefore, our Condition 4 is satisfied.

Finally, the atoms $B_i$ for $i \in \{1, \ldots, j-1, j+1, \ldots, n\}$ are marked only if they have a derivation tree in $\Pi_{\text{Marked'}}(K_{\text{Solved}})$. Therefore our Condition 3 is also satisfied. Similarly, Condition 5 is also satisfied: when $\mathcal{W}(\pi_i) \geq 1$ we have a
strict inequality otherwise we have an equality as for \( i \in \{ n + 1, \ldots, m \} \), \( \mathcal{W} ( \pi_i ) < l \).

**Remark 2:** Note that the derivation trees of the premisses of \( h \) are the ones of the premisses of \( f \) except that the derivation tree of \( B_j \) has been replaced by all its immediate sub-derivation trees.

**Lemma 17:** Let \( K \) be a saturated knowledge base and \( f, g \in K \) be a statement such that
\[
\begin{align*}
f &= \left( H \iff B_1, \ldots, B_n \parallel M \right)
\end{align*}
\]
where \( H \) is either \( i_w (R, R'), R_i (R, R') \) or \( r_w \). Let \( \sigma \) be a substitution grounding for \( f \) such that
- \( \text{ske} (f \sigma) \) is in normal form,
- for all \( 1 \leq i \leq n \) there exists a derivation tree \( \pi_i \in i' \) of \( B_i \sigma \) and if \( B_i \) is marked then \( \pi_i \in i'' \). (\( K_{\text{solved}} \)).

For all \( l \) we have that
\[
H \sigma \in H_{e}^{(l, \kappa)} (K_{\text{solved}})
\]
where \( \kappa = \hat{S} (\pi) \) where \( \pi \) is the derivation tree whose immediate sub-derivation trees are the \( \pi_i \)’s.

**Proof.** Let \( G = \sum_{i \in \{ 1, \ldots, n \}} S (\pi_i) \). We prove the lemma by induction on \( G \).

If \( f \) is a solved statement, the conclusion is immediate by the definition of \( H_{e}^{(l, \kappa)} \).

Otherwise, we apply **Lemma 16** on \( f \) and \( \sigma \). We get the existence of \( \omega \) and \( h = (H' \iff B_1', \ldots, B_m' \parallel M') \) such that \( K \) has been updated by \( h \) and which have the six properties stated in **Lemma 16**. Let \( h_\parallel = (H'' = B_1'', \ldots, B_m'' \parallel M'') \) such that \( \{ B_1', \ldots, B_m' \} \subseteq \{ B_1'', \ldots, B_m'' \} \). Since Property 1 grants that \( \text{ske} (H'') \) is in normal form and that canonicalization rules can only replace the term of the head by a subterm, the skeleton of the considered predicates \( h_\parallel \) is still in normal form and then not: the considered skeleton in the knowledge base update by the update. Then due to Property 2 \( M'' \) corresponds to the premisses which has a derivation tree in \( i'' \) of \( K_{\text{solved}} \) and due to the Property 3 the measure is strictly smaller so we can apply our induction hypothesis on \( h_\parallel \) to get \( H'' \sigma \in H_{e}^{(l, \kappa)} (K_{\text{solved}}) \) with \( \kappa' \leq \kappa \) due to the Property 4 and the set inclusion. Finally, according to the Property 2, we have \( H' \omega = H \sigma \) and by definition of \( |_{\parallel} \), we have \( H' \omega = H'' \omega \), therefore, by definition of \( H_{e}^{(l, \kappa)} \), we have \( H \sigma \in H_{e}^{(l, \kappa)} \). \( \square \)

**Definition 18:** We say that a term \( t \) has a root variable if there exists a term \( t' \) and a variable \( x \) such that \( t = t' \times x \) or \( t = x \).

**Lemma 18:** Let \( K \) be a saturated knowledge base. If \( h = k_w (R, t) \iff B_1, \ldots, B_n \in K_{\text{solved}} \) has been obtained by applying \textsc{Resolution} or \textsc{Resolution+} between a \textsc{VIP} statement \( f \) and \( g \in K_{\text{solved}} \), then \( t \) has no root variable.

**Proof.** Recall that a \textsc{VIP} statement always stems from a resolution against a solved statement of an unsolved \textsc{VIP} statement, except for the initial \( f_1^+ \) and \( f_2^+ \) \textsc{VIP} statements.

Let \( f = k_w (X + Y, x) \iff k_w (X, x \oplus u) \iff k_w (Y, u) \parallel M \in f_2^+ \). The resolution can only be done on \( k_w (X, x \oplus u) \) against some \( g = k_w (R', t_1 \oplus t_2) \iff B_1', \ldots, B_m' \) where neither \( t_1 \) nor \( t_2 \) has a root variable (the SHIF rule prevents such solved statement to be added in the database). Hence, \( x \) is substituted by a term which does not have a root variable. Then no substitution can lead to a term with a root variable: the property holds for any solved \textsc{VIP} statement whose ancestor is in \( f_1^+ \).

Let \( f \) be a solved \textsc{VIP} statement whose ancestor is \( k_w (X + Y, x + y) \iff k_w (X, x + u) \iff k_w (Y, y + u) \parallel M \in f_2^+ \). By symmetry, assume that resolution has been performed on \( k_w (X, x + u) \in f_2^+ \) with some substitution \( \sigma_1 \) and another solved statement. The direct child of the ancestor is \( f_1 = k_w (X \sigma + Y, x \sigma + y) \iff B_1', \ldots, B_m', k_w (Y, y + u \sigma_1) \parallel M' \). The previous argument of the \( f_1^+ \) case leads to \( x \sigma \) has no root variable. Let \( f_2, \ldots, f_n \) be the \textsc{VIP} children obtained by resolution on \( f_{i-1} \) and a solved statement with substitution \( \sigma_2, \ldots, \sigma_n \) but where \( \text{ske} (f_{i-1}) \neq \text{ske} (s_1 \ldots s_n \ Y, y + u \sigma_1 \ldots \sigma_n) \). However all these \( f_i \) are unsolved since \( y + u \sigma_1 \ldots \sigma_n \) is not a variable. Once a resolution is done against the literal \( k_w (Y, y + u \sigma_1 \ldots \sigma_n) \) with substitution \( \sigma' \) the previous argument leads to \( y \sigma' \) having no root variable. Since neither \( x \) nor \( y \) have a root variable, \( t \) has no root variable. Therefore, the property also holds for any solved \textsc{VIP} statement whose ancestor is in \( f_2^+ \). \( \square \)

**Lemma 19:** Let \( f_0 \) be a solved \textsc{VIP} statement whose ancestor is \( f_0 = (k_w (X + Y, y) \iff k_w (X, x + u) \iff k_w (Y, y + u) \parallel M \in f_2^+ \). We observe that none of the two premisses are solved. As \( f_0 \) is solved both premisses must have been removed by resolution and the result is immediate.

If \( f_0 = k_w (X + Y, x) \iff k_w (X, x + u) \iff k_w (Y, y + u) \parallel M \in f_2^+ \), resolution can only be performed on \( k_w (X, x + u) \) against some \( g = k_w (R', t_1 \oplus t_2) \iff B_1', \ldots, B_m' \) where neither \( t_1 \) nor \( t_2 \) has a root variable. This implies that \( u \) will be substituted by a non variable term \( t' \). Therefore at least one resolution on \( k_w (Y, t') \) is required to get a solved statement.

**Lemma 20:** Let \( K \) be a saturated knowledge base and \( h \) a solved statement
\[
h = \left( k_w (R, t) \iff B_1, \ldots, B_n \right)
\]
such that \( h_\parallel = \left( k_w (R', t') \iff B_1', \ldots, B_m' \right) \) is in normal form and
- either \( h_\parallel \in K_{\text{solved}} \).
- or \( g = \left( k_w (R''', t'') \iff B_1', \ldots, B_m' \right) \) in \( \text{conseq} (K_{\text{solved}}) \) and hence \( h' = \left( i_w (R', R''') \iff B_1', \ldots, B_m' \right) \) is in \( K_{\text{solved}} \) for some \( R''' \).
Let $\sigma$ be a substitution grounding for $h$ such that $\text{skel}(h\sigma)$ is in normal form, and for all $1 \leq i \leq n$, we have that $B_i\sigma$ has a derivation tree $\pi_i \in \Pi'(K_{\text{solved}})$. We have that

$$(k_u(R, t))\sigma \in \mathcal{H}_e(K_{\text{solved}}).$$

**Proof.** First note that by definition of the canonical form $\{B'_1, \ldots, B'_m\} \subseteq \{B_1, \ldots, B_n\}$ and $\sigma$ is grounding for $h\downarrow$ too. We proceed by induction on the number of rules which have been applied to get $h\downarrow$ from $h$.

1) **Base case:** $h\downarrow = h$.

   a) If $h\downarrow$ is in $K$, we immediately conclude.

   b) Otherwise, by applying Proposition 3 to $g$ and $\sigma$, we have that

   $$k_u(R', t')\sigma \in \mathcal{H}(K_{\text{solved}}).$$

   Furthermore, as $h' \in K_{\text{solved}}$ and as all antecedents $B'_1(\sigma), \ldots, B'_m(\sigma)$ of $h'\sigma$ are in $\mathcal{H}(K_{\text{solved}})$, we have that

   $$i_u(R' \downarrow, R' \downarrow)\sigma \in \mathcal{H}(K_{\text{solved}}).$$

   It immediately follows that

   $$k_u(R', t')\sigma \in \mathcal{H}(K_{\text{solved}}),$$

   which is what we had to prove since $k_u(R', t')\sigma = k_u(R, t)\sigma$.

2) If the rule REMOVE has been applied, the conclusion is immediate as REMOVE only remove premises.

3) If there is a normalization of the recipe, we conclude with MOD-I and EQ. CONSEQ.

4) If the rule Shift has been applied to $h = k_u(R, t) \iff k_u(X, x), B_1, \ldots, B_n$ for some $w$ such that $u \subseteq u$ to get $h' = k_u(R \oplus X, t \oplus x) \iff k_u(X, x), B_1, \ldots, B_n$. By induction hypothesis, we have $k_u(R \oplus X, t \oplus x)\sigma \in \mathcal{H}(K_{\text{solved}})$. By definition of $\mathcal{H}_e$, there is $R''$ such that

$$k_{uR}(R'' \oplus X, t \oplus x)\sigma \in \mathcal{H}(K_{\text{solved}})$$

and $i_{uR}(R'' \oplus X) \in \mathcal{H}(K_{\text{solved}})$.

By the definitions of $\mathcal{H}$ and $\mathcal{H}_e$, when $w$ is solved, we have $\pi_i = \pi'_j$, so the inequality of the measure is just

$$S_i(\pi) \leq \sum_{\{i \mid \exists j, \pi_i = \pi'_j\}} S(\pi_i) + \sum_{\{i \mid \exists j, \pi_i \neq \pi'_j\}} S(\pi_i).$$

Finally, we prove the assertion for $f_1^+$ and $f_2^+$ by using Lemma 19.

**Base case.** If $f$ is a solved statement, the conclusion is immediate by the definitions of $\mathcal{H}$ and $\mathcal{H}_e$.

**Induction case.** If $f$ is not solved, we apply Lemma 16 to get $h, \omega$ and $\pi_i$ derivation trees of $B'_i \omega$ where

$$h = k_u(R_h, t_h) \iff B'_1, \ldots, B'_m \iff M'$$

such that $K$ has been updated by $h$ and which have the five properties stated in Lemma 16.

1) **Case where $h$ is not solved.** Due to Property 3, $M''$ corresponds to the premises which has a derivation tree in $\Pi'_{\text{solved}}$ and due to the Property 4 the measure is strictly smaller so we can apply our induction hypothesis on $h\downarrow$ to get $k_u(R', t'_{h\downarrow}) \omega \in \mathcal{H}(K_{\text{solved}})$. Finally, according to the Property 5 we have $k_u(R_h, t_h)\omega = k_u(R, t)\sigma$ and by definition of $\downarrow$ when the statement is not solved, we have $k_u(R'_{h\downarrow}, t'_{h\downarrow})\omega = k_u(R_h, t_h)\omega$, therefore, by definition of $\mathcal{H}_e$, we have $k_u(R, t)\omega \in \mathcal{H}(K_{\text{solved}})$. Finally, we consider that $f$ is a VIP statement. In that case, $t_h$ cannot be of the form $t + x$ according to Lemma 18 so the rule SHIFT is not applied. From our induction hypothesis, we have

$$S_i(\pi) \leq \sum_{\{i \mid \exists j, \pi_i = \pi'_j\}} S(\pi'_i) + \sum_{\{i \mid \exists j, \pi_i \neq \pi'_j\}} S(\pi'_i).$$
From Remark 2 since \( \forall x, S(x) > \hat{S}_i(x) \), we have
\[
\hat{S}_i(\pi) \leq \sum_{i | W(\pi_i) \geq l^i} S_j(\pi_i) + \sum_{i | W(\pi_i) > l^i} \hat{S}_i(\pi_i).
\]

2) Case where \( h \) is solved. If \( f \) is a VIP statement, due to Lemma 18 no SHIFT rule has been used and by definition \( h \) is a VIP statement, so \( h \) has been added to \( K \). We can then conclude since \( h \) is just the renormalization of the head and the removal of some premises (with Remark 2 of Lemma 16 we have \( \hat{S}_i(\pi) \leq \sum_j S(\pi_j) \)). If \( f \) is not a VIP statement, we conclude by Lemma 20.

The following lemma will be used in the proof of Theorem 3 but is not used to prove the completeness.

**Lemma 22:** Let \( K \) be a saturated knowledge base. If \( r_w \in \mathcal{H}(K_{\text{solved}}) \), \( i_w(R, R') \in \mathcal{H}(K_{\text{solved}}) \) then there exist a derivation tree \( \pi \in \Pi'(K_{\text{solved}}) \) of \( i_w(R, R') \) and \( n_{i_w}(R, R') \in \mathcal{H}_{\{1, \kappa\}}(K_{\text{solved}}) \) with \( \kappa = S_{|w|}(\pi) \).

**Proof:** Let \( u = \ell_1, \ldots, \ell_k \) and \( v = \ell_{k+1}, \ldots, \ell_l \). As \( k_w(R, s) \in \mathcal{H}(K_{\text{solved}}) \), it follows from its derivation tree that there exist
\[
f = (k_w(s, s) \leq B_1, \ldots, B_m) \in K_{\text{solved}}
\]
and a substitution \( \sigma \) grounding for \( f \) such that \( \text{skel}(f) \) is in normal form, \( B_i \sigma \in \mathcal{H}(K_{\text{solved}}) \) and the derivation tree \( \pi_i \) and \( k_w(s, s) \sigma = k_w(R, t) \) for some \( u' \subseteq u \) a prefix of \( u \).

Similarly, as \( k_{uv}(R', t) \in \mathcal{H}(K_{\text{solved}}) \), it follows that there exist
\[
f' = (k_{uv}(s', s') \leq B'_1, \ldots, B'_{m'}) \in K_{\text{solved}}
\]
and a substitution \( \sigma' \) grounding for \( f' \) such that \( \text{skel}(f') \) is in normal form, \( B_i' \sigma' \in \mathcal{H}(K_{\text{solved}}) \) and the derivation tree \( \pi_i' \) and \( k_{uv}(s', s') \sigma' = k_{uv}(R', t) \) for some \( u'' \subseteq u' \) a prefix of \( u' \).

We have that \( w \sigma \subseteq u \), which trivially implies \( w \sigma \subseteq w'v \). We also have \( w' \sigma \subseteq u \). Let \( w = \ell_1', \ldots, \ell_p' \) and \( w' = \ell_1'', \ldots, \ell_{p'}'' \) and let \( r = \min\{p, q\} \). We have that \( (\ell_1', \ldots, \ell_r') \sigma = (\ell_1'', \ldots, \ell_{r''}) \sigma' \).

As \( \sigma \cup \sigma' \) is a unifier of \( k_{t_1', \ldots, t_{r'}'}(s, s') \) and \( k_{t_1'', \ldots, t_{r''}''}(s, s') \), it follows that \( \pi \in \text{cns}(k_{t_1', \ldots, t_{r'}'}(s, s'), k_{t_1'', \ldots, t_{r''}''}(s, s')) \). As \( K \) is saturated and \( f, f' \neq f_0 \), it follows that the statement
\[
h = (\ell_1', \ldots, \ell_r') \sigma = (\ell_1'', \ldots, \ell_{r''}) \sigma' \subseteq B_1, \ldots, B_n, B'_1, \ldots, B'_{m'}
\]
resulting from applying the EQUATION saturation rule to \( f \) and \( f' \) has been generated during the saturation process. The knowledge base has been updated with this statement. Since we know that \( \text{skel}(f) \) and \( \text{skel}(f') \) are in normal form, we have that \( \text{skel}(h) \) is in normal form, and therefore \( h \) is in \( K \) (since \( S \) and \( S' \) are necessarily in normal form).

As \( \sigma \cup \sigma' \) is a unifier of \( k_{t_1', \ldots, t_{r'}'}(s, s') \) and \( k_{t_1'', \ldots, t_{r''}''}(s, s') \), it follows that there exists \( \sigma \) such that \( \sigma \cup \sigma' = \emptyset \).

We have that \( h \) is grounding for \( h \) and that \( B_1 \tau \omega, \ldots, B_n \tau \omega, B'_1 \tau \omega, \ldots, B'_{m} \tau \omega \in \mathcal{H}(K_{\text{solved}}) \) with derivation trees \( \pi_1, \ldots, \pi_n, \pi'_1, \ldots, \pi'_{m} \) and
\[
\sum_{i | W(\pi_i) \geq l} \{S(\pi_i) + \sum_{i \in \mathcal{H}(K_{\text{solved}}) \text{ with \text{skel}(\pi_i) \neq \emptyset}} S(\pi_i) = \hat{S}_i(\pi) + \hat{S}(\pi')\}
\]
where \( B_i = k_w(X_i, x_i) \) and \( B'_i = k_w(x_i, x'_i) \). Therefore, we have by Lemma 17 that
\[
i_{\ell_1', \ldots, \ell_r'}(s, s') \tau \omega = i_{\ell_1''', \ldots, \ell_{r''}'''}(R, R') \in \mathcal{H}_{\{1, \kappa\}}(K_{\text{solved}}).
\]
where \( \kappa = \sum_{i \in \mathcal{H}(K_{\text{solved}}) \geq l} S(\pi_i) \). As \( (\ell_1', \ldots, \ell_r') \sigma \) is a prefix of \( uv \), we conclude.

**Lemma 24:** Let \( K \) be a saturated knowledge base. Let \( \{k_w(R_i, t_i) \mid 1 \leq i \leq n\} \) be a set of deduction statement such that \( \pi_i \in \Pi'(K_{\text{solved}}) \) is a derivation tree for \( k_w(R_i, t_i) \), \( \bigoplus_{i \in \{1, \ldots, n\}} t_i = 0 \) and \( u \subseteq \{u, uv\} \) for some \( u, v \). We
have that $i_{uv}(\bigoplus R_i, 0) \in H_{e}(t, \kappa)_{(\ast, \kappa)}(K_{\text{solved}})$ for all $l$ and with $k \leq \sum_{\downarrow} S_l(\pi_l)$.

Proof. We denote $k_{uv}(R_i, t_i)$ by $K_i$. W.l.o.g. for all $i$, we suppose that $t_i = \bigoplus_{1 \leq j \leq l} t_{i,j}^{l_i}$. We prove this lemma by induction on $\sum_{\downarrow} s(i)$. If there is some $k_{uv}(R_i, 0)$, from the seed, we have $k_{uv}(0, 0)$ which has a derivation tree $\pi_0$ in $\Pi_\kappa(K_{\text{solved}})$ and $\hat{S}_l(\pi_0) = 0$. By Lemma 23, we get $i_{uv}(R_i, 0) \in H_{e}(t, \kappa)_{(\ast, \kappa)}(K_{\text{solved}})$ with $k \leq \hat{S}_l(\pi_l) + \hat{S}_l(\pi_0)$. We conclude by induction hypothesis on $\{K_i\}_{i \neq \downarrow}$ and by use the CONG rule with $\oplus$.

Otherwise, since $\bigoplus t_{i,j} = 0$, there exist at least $K_k$ of derivation tree $\pi_k$. Derivation tree $\pi_k$ with $k \leq \sum_{\downarrow} S_l(\pi_l)$ of maximal size which satisfies the former relation. We define $S_k$ to be the set $\{i | \oplus t_{i,j} = t_{i,k}\}$ and define $S_l$ similarly.

If $t_k = t_j$, from Lemma 23 we get $i_{uv}(R_k, R_l) \in H_{e}(t, \kappa)_{(\ast, \kappa)}(K_{\text{solved}})$ with $k \leq \sum_{\downarrow} S_l(\pi_l) + \sum_{\downarrow} S_l(\pi_j)$. From Lemma 15 we get $i_{uv}(R_k \oplus R_l, 0) \in H_{e}(t, \kappa)_{(\ast, \kappa)}(K_{\text{solved}})$. If the set contained other predicate that $K_k$ and $K_j$, we use our induction hypothesis on the set $\{K_i | i \neq \downarrow, j\}$. We get $i_{uv}(\bigoplus_{i \neq \downarrow} S_l(\pi_l), 0) \in H_{e}(t, \kappa)_{(\ast, \kappa)}(K_{\text{solved}})$ with $k \leq \sum_{\downarrow} S_l(\pi_l)$ and $S_j$.

We apply our induction hypothesis on the set $\{K_i \} | i \neq \downarrow, j\}$ and obtain that $i_{uv}(\bigoplus_{i \neq \downarrow} S_l(\pi_l), R_l, 0) \in H_{e}(t, \kappa)_{(\ast, \kappa)}(K_{\text{solved}})$ with $\kappa \leq \sum_{\downarrow} S_l(\pi_l) + \sum_{\downarrow} S_l(\pi_j)$.

We apply our induction hypothesis on the set $\{K_i \} | i \neq \downarrow, j\}$ and obtain that $i_{uv}(\bigoplus_{i \neq \downarrow} S_l(\pi_l), 0) \in H_{e}(t, \kappa)_{(\ast, \kappa)}(K_{\text{solved}})$ with $\kappa \leq \sum_{\downarrow} S_l(\pi_l) + \sum_{\downarrow} S_l(\pi_j)$. Finally, using CONG, MOD-I and Lemma 15, we get $i_{uv}(\bigoplus_{i \neq \downarrow} S_l(\pi_l), R_l \oplus R_k, 0) \in H_{e}(t, \kappa)_{(\ast, \kappa)}(K_{\text{solved}})$ with $\kappa \leq \sum_{\downarrow} S_l(\pi_l) + \sum_{\downarrow} S_l(\pi_j)$.

We apply our induction hypothesis on the set $\{K_i \} | i \neq \downarrow, j\}$ and obtain that $i_{uv}(\bigoplus_{i \neq \downarrow} S_l(\pi_l), 0) \in H_{e}(t, \kappa)_{(\ast, \kappa)}(K_{\text{solved}})$ with $\kappa \leq \sum_{\downarrow} S_l(\pi_l) + \sum_{\downarrow} S_l(\pi_j)$.

Proof. As $k_{uv}(R, t) \in H_{e}(K)$, it follows that $k_{uv}(R', t) \in H_{e}(K)$ and $i_{uv}(R', 0) \in H_{e}(K)$ for some $R'$. By the EXTENDK rule, we have that $k_{uv}(R', t) \in H_{e}(K)$ and by the EXTENDK rule, we have that $i_{uv}(R', R) \in H_{e}(K)$. We conclude by rule EQUATIONAL CONSEQUENCE that $k_{uv}(R, t) \in H_{e}(K)$, which is what we had to show.

Definition 19: Given a term $t = \bigoplus_{1 \leq i \leq n} t_i$ where no $t_i$ is a sum, then $width(t) = n - 1$.

Lemma 26: Let $K$ be a saturated knowledge base. If $\{k_{uv}(R_i, t_i) \, | \, i \in \{1, \ldots, m\}\}$ is a set of deduction statements with derivation trees $\pi_i \in \Pi_\kappa(K_{solved})$ then

$$\bigoplus_{1 \leq i \leq m} k_{uv}(R_i, 0) = \bigoplus_{1 \leq i \leq m} t_i \downarrow \in H_{e}(K_{\text{solved}}).$$

Proof. We proceed by induction on $\sum_{1 \leq i \leq m} width(t_i)$. Base case. $\bigoplus_{1 \leq i \leq m} t_i \downarrow = \bigoplus_{1 \leq i \leq m} t_i$. We show the base case by an induction on $m$. If $m = 1$ the result is immediate. If $m > 1$ then for any pair $(j, k)$, we can build a new derivation tree of $k_{uv}(R_j \oplus R_k, t_j \oplus t_k)$ from $t_j \oplus t_k$, $\pi_j$ and $\pi_k$. We conclude by applying our induction hypothesis.

Inductive case. Suppose $\bigoplus_{1 \leq i \leq m} t_i \downarrow = \bigoplus_{1 \leq i \leq m} t_i$. If there exists $j$ such that $\pi_j \in \Pi_\kappa(K_{\text{solved}})$, then we consider $\pi_0$ and $\pi_0$ its immediate sub-trees which prove $k_{uv}(R_0, t_0)$ and $k_{uv}(R_0, t_0)$. We conclude by induction hypothesis on the set where $k_{uv}(R_j, t_j)$ has been replaced by $k_{uv}(R_0, t_0)$ and $k_{uv}(R_0, t_0)$. Otherwise there exist $j, k$, $\pi_j \in \Pi_\kappa(K_{\text{solved}})$ and $\pi_k \in \Pi_\kappa(K_{\text{solved}})$ such that $(t_j \oplus t_k) \downarrow = t_j \oplus t_k$.

- If $t_j = t_k$, from Lemma 23 we have $i_{uv}(R_j, R_k) \in H_{e}(K_{\text{solved}})$.
- If $t_j \neq t_k$, there exist $t_j', t_k'$ such that $t_j = t_j' \downarrow + t_j'$ and $t_k = t_k' \downarrow + t_k'$. Applying Lemma 21 on the suitable $f \in f_{\downarrow}^\downarrow \cup f_{\downarrow}^\uparrow$ and the derivation trees $\pi_j$ and $\pi_k$ we obtain $k_{uv}(R_j \oplus R_k, t_j \oplus t_k) \in H_{e}(K_{\text{solved}})$.
- If $t_j = t_k$, from Lemma 15 we have $i_{uv}(R_j \oplus R_k-R', R') \in H_{e}(K_{\text{solved}})$.

Using CONG rule we obtain that $i_{uv}(R_j \oplus R_k, 0) \in H_{e}(K_{\text{solved}})$.

By induction hypothesis, we get $k_{uv}(R_j \oplus R_k, 0) \in H_{e}(K_{\text{solved}})$ and we conclude by Eq. Cons.

- If $t_j \neq t_k$, there exist $t_j', t_k'$ such that $t_j = t_j' \downarrow + t_j'$ and $t_k = t_k' \downarrow + t_k'$. Applying Lemma 21 on the suitable $f \in f_{\downarrow}^\downarrow \cup f_{\downarrow}^\uparrow$ and the derivation trees $\pi_j$ and $\pi_k$ we obtain $k_{uv}(R_j \oplus R_k, t_j \oplus t_k) \in H_{e}(K_{\text{solved}})$.

- If $t_j = t_k$, from Lemma 15 we have $i_{uv}(R_j \oplus R_k-R', R') \in H_{e}(K_{\text{solved}})$.

Using CONG rule we obtain that $i_{uv}(R_j \oplus R_k, 0) \in H_{e}(K_{\text{solved}})$.

Since $\sum_{1 \leq i \leq m} width(t_j \oplus t_k) \downarrow < width(t_j) + width(t_k)$, by induction hypothesis, we get $k_{uv}(R_j \oplus R_k-R', t_i \downarrow) \in H_{e}(K_{\text{solved}})$.
and using Eq. CONSEQ. we conclude that

\[ k_w( \bigoplus_{1 \leq i \leq m} R_i, \bigoplus_{1 \leq j \leq m} t_i \downarrow ) \in H_e(K_{\text{solved}}). \]

\[ \square \]

Lemma 27: Let S be a set of seed statements and let \( K \in \text{sat}(K_{\text{init}}(S)) \). Then if \( H \in H(S) \), we have that \( H \in H_e(K_{\text{solved}}) \).

Proof. We prove by induction on the derivation tree of \( H \in H(S) \) that each node of the derivation tree is in \( H_e(K_{\text{solved}}) \). We proceed by case distinction on the last rule that has been applied to derive \( H \).

Case: EXTEND. We have that \( H = k_w(R, t) \) and \( k_u(R, t) \in H(S) \) for some prefix \( u \) of \( w \), in which case by the induction hypothesis we have that \( k_u(R, t) \in H_e(K_{\text{solved}}) \) and we conclude by Proposition 5.

Case: SIMPLE CONSEQUENCE. There is a statement

\[ f = \left( H' \leftarrow B_1', \ldots, B_n' \right) \in S \]

and a substitution \( \sigma \) such that \( \text{ske}(f \sigma) \) is in normal form, \( H = H' \sigma \) and \( B_i' \sigma \in H(S) \). By the induction hypothesis, we have that \( B_i' \sigma \in H_e(K_{\text{solved}}) \). W.l.o.g. assume that \( B_i' = k_{w_i'}(X_i, t_i') \). As \( B_i' \sigma \in H_e(K_{\text{solved}}) \), we have by definition of \( H_e \) that there exist \( R_i' \) such that

\[ B_i'' = k_{w_i'}(R_i', t_i' \sigma) \in H_e(K_{\text{solved}}) \]

(3)

\[ i_{w_i'}(R_i', t_i' \sigma) \in H_e(K_{\text{solved}}) \]

(4)

for all \( 1 \leq i \leq n \) and \( \eta_i' \) be a derivation tree in \( \Pi_i'(K_{\text{solved}}) \) of \( B_i'' \).

But \( w_i' \sigma \) is a prefix of \( w \), where \( w \) is such that \( H = \text{predicate}_{w_i'}(\ldots) \) with \( \text{predicate} \in \{r, k\} \). Note that as \( S \) is a set of seed statements, \( \text{predicate} \not\in \{i, n\} \). By applying the EXTEND rule to Equation (4), we obtain

\[ i_w(R_i', X_i \sigma) \in H_e(K_{\text{solved}}) \]

(5)

Let \( \eta' \) be the substitution defined to be \( \sigma \) except that it maps \( X_i \) to \( R_i' \) for all \( 1 \leq i \leq n \). We distinguish several cases depending on the statement \( f \).

1) Case where \( f = f_j^+, f_j^-, f_j^+, f_j^+ \) or \( f_j^+ \). We conclude by applying Lemma 26 and rules of \( H_e \).

2) Case where \( f = f_j^+, f_j^+ \) or \( f_j^+ \).

We will show that \( H' \sigma' \in H_e(K_{\text{solved}}) \). As \( K \) was updated by \( f \), there are three cases:

a) if \( f \in K \), we conclude by Lemma 21 or Lemma 17 (depending on the predicate).

b) Otherwise, \( f \downarrow \in K_{\text{solved}} \) or \( f \downarrow = (k_w(R, t) \leftarrow C_1, \ldots, C_m) \) and there exists \( R' \) such that

\[ (k_w(R', t) \leftarrow B_1, \ldots, B_n) \in \text{conseq}(K_{\text{solved}}) \]

such that

\[ (i_w(R', R' \downarrow) \leftarrow B_1, \ldots, B_n) \in K_{\text{solved}}. \]

In this case we conclude by Lemma 20.

We have shown that \( H' \sigma' \in H_e(K_{\text{solved}}) \). We distinguish several cases depending on \( \text{predicate} \):

- \( \text{predicate} = r \): In such a case, we have that \( H' \sigma' = H' \sigma = H \) and we easily conclude.

- \( \text{predicate} = k \): In such a case, we have that \( H' \sigma' = k_{w}(R \sigma', t \sigma') \). Relying on Equation 5 and applying the rules CONG and REFL, we deduce that \( i_w(R \sigma, R \sigma') \in H_e(K_{\text{solved}}) \). Since \( k_w(R \sigma, t \sigma') \in H_e(K_{\text{solved}}) \) and \( t \sigma = t \sigma' \), using Proposition 4, we conclude that \( k_w(R \sigma, t \sigma) \in H_e(K_{\text{solved}}) \).

\[ \square \]

Appendix

Proof of the Algorithm

A. Proof of Lemma 2

A first issue is that \( K_{\text{init}}(P) \) is infinite. Indeed, the set \( \text{seed}(P) \) for a ground process \( P \) is infinite because \( N_{\text{pub}} \) contains an infinite set of names.

Lemma 28: Let \( K \in \text{sat}(K_{\text{init}}(\text{seed}(P, N_{\text{pub}}^{P}))) \), we have that \( K \cup \text{ext}(K) \) is a saturated knowledge base.

Proof. First, we have that \( K \cup \text{ext}(K) \) is a knowledge base. Second, we show that any rule triggered from \( K \cup \text{ext}(K) \) is either a rule involving two statements of \( K \) as premisses (and thus this has already been done during the saturation of \( K \)) or produces a statement in \( \text{ext}(K) \).

1) if the rule RESOLUTION is triggered, we have \( f, g \in K \). First, no statement \( (k(m, m) \leftarrow \ldots) \in \text{ext}(K) \) can play the role of \( g \) in the RESOLUTION saturation rule since \( t' = m \) must unify with \( t \not\in X \). Therefore \( t \) must be \( m \), but \( m \not\in \text{names}(K) \) by hypothesis and therefore \( t \) cannot be \( m \). Second, no statement in \( \text{ext}(K) \) can play the role of \( f \) in the RESOLUTION saturation rule since they have no antecedents. Therefore \( f, g \in K \).

2) if the rule EQUATION is triggered, we distinguish 3 cases:

a) if a statement \( (k(m, m) \leftarrow \ldots) \in \text{ext}(K) \) plays the role of \( g \) in the EQUATION rule, then we have that \( t = m \). As \( t' \) unifies with \( m \), we have that either \( t' = m \) or \( t' \) is a variable. The second case is not possible since \( g \) is well-formed. Therefore \( t' = m \). As \( m \not\in \text{names}(K) \) by hypothesis it follows that \( g \in \text{ext}(K) \) and therefore \( g = k(m, m) \). Therefore the resulting statement is \( i(m, m) \in \text{ext}(K) \).

b) if a statement \( (k(m, m) \leftarrow \ldots) \in \text{ext}(K) \) plays the role of \( g \), the reasoning is analogous to the case above

c) otherwise \( f, g \in K \) but since \( K \) is saturated this saturation step has already occurred.

3) if rule TEST triggered, we distinguish two cases:

a) if \((i(m, m) \leftarrow \ldots) \in \text{ext}(K) \) plays the role of \( f \), then \( g = t_a \in B_1, \ldots, B_n \in K \) and the result is in \( \text{ext}(K) \).

b) otherwise \( f \in K \). The statement \( g \) must also be in \( K \) since \( g \) is a reachability statement and \( \text{ext}(K) \) does not contain reachability statements. Therefore the rule has already been applied.

\[ \square \]
We denote by $\Rightarrow$ one step of the saturation relation, and by $\Rightarrow^*$ the transitive closure of $\Rightarrow$.

Relying on Lemma 28, we are now able to prove Lemma 1.

Lemma 1: Let $P$ be a ground process, $N'_{\text{pub}} \subseteq N_{\text{pub}}$ be the finite set of public names occurring in $P$. We have that:

$$\text{sat}(K_{\text{init}}(P)) \supseteq \{ K \cup \text{ext}(K) \mid K \in \text{sat}(K_{\text{init}}(\text{seed}(P, N'_{\text{pub}}))) \}$$

where $\text{ext}(K)$ is the set containing the following statements:

1. $k_i(n, n) = K$ for any $n \in N_{\text{pub}} \setminus N'_{\text{pub}}$;
2. $i_i(n, n) = K$ for any $n \in N_{\text{pub}} \setminus N'_{\text{pub}}$;
3. $ri_i(n, n) = B_1, \ldots, B_n$ for any $r_i \subseteq B_1, \ldots, B_n \in K$ in solved form, any $n \in N_{\text{pub}} \setminus N'_{\text{pub}}$.

Proof. Let $K_1 = \{ k_i(n, n) = \mid n \in N_{\text{pub}} \setminus N'_{\text{pub}} \}$. We have $K_{\text{init}}(P) = K_{\text{init}}(\text{seed}(P, N'_{\text{pub}})) \cup K_1$. We apply saturation and we trigger only saturation rules whose premises are not in $\text{ext}(K)$. We get:

$$K_{\text{init}}(P) = K_{\text{init}}(\text{seed}(P, N'_{\text{pub}})) \cup K_1 \Rightarrow^* K \cup K_1$$

where $K = \text{sat}(K_{\text{init}}(\text{seed}(P, N'_{\text{pub}})))$. Then we can show that $K \cup K_1 \Rightarrow^* K \cup \text{ext}(K)$ (we apply the rule EQUATION on the duplication of each element of $K_1$, then the rule TEST between each reach statement of $K$ and the obtained identical). Finally, from Lemma 28, we have that $K \cup \text{ext}(K)$ is saturated. \(\square\)

B. Correlation of the algorithm

In order to prove Theorem 3, we need the following technical lemmas.

Lemma 29: Let $P$ be a ground process and $K \in \text{sat}(K_{\text{init}}(P))$. Then for any statement $f \in K$, we have that:

1. if $f = \left( ri_i, \ldots, i_n(R, R') \right) \subseteq \{ k_{w_i}, \{ X_i, t_i \} \}_{i \in \{1, \ldots, m\}}$ and $x \in \text{vars}(l_k)$ then there exists $w_j = t_i, \ldots, t_{k'}$ with $k' < k$ such that $x \in \text{vars}(t_j)$.
2. if $f = \left( k_{i_1}, \ldots, i_n(R, t) \right) \subseteq \{ k_{w_i}, \{ X_i, t_i \} \}_{i \in \{1, \ldots, m\}}$ and $x \in \text{vars}(t)$ then $x \in \text{vars}(t_1, \ldots, t_m)$.

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

Lemma 30: Let $P_0$ be a ground process, $\varphi_0 = \emptyset$ the empty frame, and $\{ c_1, \ldots, c_k \}$ names such that $c_i \notin \text{names}(P_0)$ for all $1 \leq i \leq k$. If

$$(P_0, \varphi_0) \xrightarrow{\theta} (P_1, \varphi_1) \xrightarrow{\theta} \ldots \xrightarrow{\theta} (P_n, \varphi_n)$$

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

- either $c_i \notin \text{names}(L_1, \ldots, L_n)$
- or there exist $R_i$ and $t_i$ such that $\varphi_{\text{id}(c_i)-1} \vdash R_i t_i$ where $\theta = \{ c_i \mapsto t_i \}_{i \in \{1, \ldots, k\}}$ and $\text{id}(c_i) = \min\{ j \mid c_i \in \text{names}(L_j) \}$

then

$$(P_0, \varphi_0) \xrightarrow{\varphi_{\text{id}(c_i)-1} \theta} (P_1, \varphi_1) \xrightarrow{\varphi_{\text{id}(c_i)-1} \theta} \ldots \xrightarrow{\varphi_{\text{id}(c_i)-1} \theta} (P_n, \varphi_n)$$

where $\varphi_{\text{id}(c_i)-1} = \{ c_i \mapsto R_i \}_{i \in \{1, \ldots, k\}}$.

Proof. By induction on the length $n$ of the derivation. \(\square\)

Lemma 31: Let $P$ be a ground process, $\{ c_1, \ldots, c_k \}$ be public names not occurring in $P$, $\theta : \{ c_1, \ldots, c_k \} \rightarrow T(\Sigma, \mathcal{N})$, and $\theta' : \{ c_1, \ldots, c_k \} \rightarrow T(\Sigma, \mathcal{N}_{\text{pub}} \cup \mathcal{W})$. If $P \models \psi$, $P \models \psi(w, r, t)$, and $P \models \psi(c, \theta, c, \theta)$ then $P \models \psi(c, \theta', c, \theta)$.

Proof. Suppose that $P \models \psi$. Otherwise, the conclusion trivially follows from the semantics of the rule $T$ predicate. Let $w = t_1, \ldots, t_n$.

As $P \models \psi$, we have that $(P, \psi) \xrightarrow{\theta_1, \ldots, \theta_n} (Q, \varphi)$ such that for all $1 \leq i \leq n$ it holds that $L_i \varphi \models \mathcal{N}_{\text{pub}} \cup \mathcal{W}$.

As $P \models \psi$, we have that $(P, \psi) \xrightarrow{\theta_1, \ldots, \theta_n} (Q', \varphi')$ such that for all $1 \leq i \leq n$ it holds that $L_i \varphi' \models \mathcal{N}_{\text{pub}} \cup \mathcal{W}$.

By induction on $n$ we can show that $\varphi' \models_{\mathcal{N}_{\text{pub}}} \psi$. Finally, we show by induction on $R$ that $\varphi \models_{\mathcal{N}_{\text{pub}}} \psi$ and therefore the transition still holds. \(\square\)

Theorem 3: Let $P$ be a ground process, and $N'_{\text{pub}} \subseteq N_{\text{pub}}$ be the finite set of public names occurring in $P$. Let $P$ be a protocol, and $K^0 \in \text{sat}(K_{\text{init}}(\text{seed}(P, N'_{\text{pub}})))$. We have that:

- if $P \subseteq P'$ then REACH-IDENTITY$(K_{\text{solved}}, P'; P)$ holds;
- if $P$ is determinate and REACH-IDENTITY$(K_{\text{solved}}, P)$ holds then $P \subseteq P'$.

Proof. Let $K \in \text{sat}(K_{\text{init}}(P))$. By Lemma 1, we have that $K_{\text{solved}} = K_{\text{solved}} \cup K_{\text{R}}$ for some $K_{\text{R}}$ as defined in Lemma 1. We first prove that if REACH-IDENTITY fails then $P \not\subseteq P'$. In such a case, we have that

$$(ri_{i_1}, \ldots, i_n(R, R') \leftarrow \{ k_{w_i}, \{ X_i, t_i \} \}_{i \in \{1, \ldots, m\}} \in K_{\text{solved}}$$

1. either $(P, \psi) \xrightarrow{\theta_1, \ldots, \theta_n} (P', \varphi)$ for any $(P', \varphi)$. However, by Theorem 2 (soundness of $K_{\text{solved}}$ and thus of $K^0$), there exists $(P', \varphi')$ such that $(P, \psi) \xrightarrow{\theta_1, \ldots, \theta_n} (P', \varphi')$. Hence, we have that $P \not\subseteq P'$.
2. or for any $(P', \varphi)$ such that $(P, \psi) \xrightarrow{\theta_1, \ldots, \theta_n} (P', \varphi)$ and any grounding substitution $\omega$ we have that $R_{\omega} \varphi \not\models_{\mathcal{N}_{\text{pub}}} R_{\omega} \varphi'$. By Theorem 2 we have that there exists $(P', \varphi')$ such that $(P, \psi) \xrightarrow{\theta_1, \ldots, \theta_n} (P', \varphi')$ and $R_{\omega} \varphi' \not\models_{\mathcal{N}_{\text{pub}}} R_{\omega} \varphi'$. Hence, we have that $P \not\subseteq P'$.

Next, we prove that if $P \not\subseteq P$ and $P$ determinate, then REACH-IDENTITY fails. We assume that $P \not\subseteq P$, and that REACH-IDENTITY holds, and we derive a contradiction. As $P \not\subseteq P$, it follows that there exist $L_1, \ldots, L_n$, and $\varphi$ such that $(P, \psi) \xrightarrow{\theta_1, \ldots, \theta_n} (P', \varphi')$ and $(R = R') \varphi'$. Let $\psi$ be the frame reached by $P$ after the execution of $L_1, \ldots, L_n$.

We consider such a witness of minimal length. Note that, since $P$ is determinate, even if such a frame $\psi$ is not unique,
they all satisfy the same tests. Note also that we can w.l.o.g. consider that at least one test holds in $P'$ (possibly considering a trivial one, i.e. $0 = 0$).

As $(P, 0) \xrightarrow{L_1, \ldots, L_n} (P_1, \emptyset_n) = (P', \emptyset)$ and $(R = E R') \emptyset_n$, by completeness, we have that:

$$i_{L, \emptyset_n} (R', R') \in \text{He}(K_{\text{Solved}}).$$

In the second case, we also have that $(R \notin E R') \emptyset$. By the fact that $L, \emptyset_n \in \text{He}(K_{\text{Solved}})$ and $(R \notin E R') \emptyset$, we can show that there exist recipes $R_A, R_A'$ and $k \leq n$ such that $i_{L, \emptyset_n} (R, R') \in \text{He}(K_{\text{Solved}})$ but $(R_A \neq R_A') \emptyset$. Actually, we have that $k = n$ in order to not contradict the minimality of $n$.

We now choose among all the minimal witnesses (w.r.t. the length) of non-inclusion, one such that $S_n(\pi)$ is minimal too. More formally, we consider a witness of non-inclusion such that $(n, S_n(\pi))$ is minimal (with a lexical order) where $n$ is the length of the witness and $\pi$ is the derivation tree in $\text{He}(K_{\text{Solved}})$ of $i_{L, \emptyset_n} (R, R')$.

We denote by $\pi_m$ the derivation tree of such a minimal witness $i_{L, \emptyset_n} (R, R') \in \text{He}(K_{\text{Solved}})$.

In the first case (item 1), we simply have to consider a witness of non-inclusion of minimal length, and we may assume that $R_A = R_A' = 0$ (in which case its derivation tree has size $S_n(\pi_m) = 0$), and of course, we do not have that $(R_A \neq R_A') \emptyset$. Now, considering such a minimal witness, we have that:

$$(P, 0) \xrightarrow{L_1, \ldots, L_n} (P_1, \emptyset_1) \xrightarrow{} \ldots (P_{n-1}, \emptyset_{n-1}) \xrightarrow{L_n} (P_n, \emptyset_n)$$

By Theorem 2 (completeness), we know that:

$$\text{ri}_{L, \emptyset_n} (R, R') \in \text{He}(K_{\text{Solved}}).$$

By the definition of $\text{He}$, we have that it contains no reachability statement in addition to those in $H$. Therefore, we have that:

- $\text{ri}_{L, \emptyset_n} (R, R') \in \text{He}(K_{\text{Solved}})$, and $i_{L, \emptyset_n} (R, R') \in \text{He}(K_{\text{Solved}})$.

Thanks to Lemma 22, we deduce that:

$$\text{ri}_{L, \emptyset_n} (R, R') \in \text{He}(n, S_n(\pi_m)) (K_{\text{Solved}}).$$

Therefore there exists a statement

$$f = \left( \text{ri}_{L, \ldots, L_n} (R_B, R_B') \right) \in K_{\text{Solved}}$$

and a substitution $\tau$ grounding $f$ such that:

- $k_{\omega_1, \tau} (X_1, \tau, \emptyset) \in \text{He}(K_{\text{Solved}})$ of derivation tree $\pi_1$ (for all $1 \leq i \leq m$),
- $L_1, \ldots, L_n, R_B \tau, L_B, R_B' \tau \in \text{He}(K_{\text{Solved}})$.

Due to the shape of $f$ (note that $(R_B, R_B') \neq (n, n)$ for any public name $n$), we have that $f \in K_{\text{Solved}}^0$. We suppose w.l.o.g. that $i \leq j$ implies $w_i \subseteq w_j$ for $f$.

We also build another substitution $\beta$ as follows: for each $1 \leq i \leq m$, consider the set $\text{comp}_{i, j} = \{ j \mid x_j = x_i \}$. Consider the map $\text{least} : \{ 1 \leq i \leq m \} \rightarrow \{ 1 \leq i \leq m \}$ defined as $\text{least}(i) = x_i$ where $r = \min\{ j \mid w_j \subseteq \text{comp}_{i, j} \}$. Finally $\beta$ is the substitution such that $\beta(X_i) = X_{\text{least}(i)}$.

Let $c_1, \ldots, c_k$ be fresh public names and let

$$\sigma : \text{vars}(l_1, \ldots, l_n) \cup \{ x_1, \ldots, x_m \} \rightarrow \{ c_1, \ldots, c_k \}$$

be a bijection of inverse $\sigma^{-1}$. We have that:

- $k(c_j, c_j) \in K_{\text{Solved}}$ for all $1 \leq j \leq k$;
- $k_{\emptyset, \emptyset} (X_i \sigma', x_i \sigma) \in \text{He}(K_{\text{Solved}})$ for all $1 \leq i \leq m$;
- $\text{dom}(\sigma') = \{ X_1, \ldots, X_m \}$ and $\sigma'(X_i) = x_i \sigma$ for all $1 \leq i \leq m$. We also define the mapping $\sigma''^{-1}$ to be such that $\text{dom}(\sigma''^{-1}) = \{ c_1, \ldots, c_k \}$ and $\sigma''^{-1} (c_i) = X_j$ where $j$ is such that $\sigma'(X_j) = c_i$ and $X_j \beta = X_j$. Instantiating $f$ with $\sigma \cup \sigma'$, we obtain that

$$\text{ri}_{i, \sigma, \ldots, \sigma_n} (R_B \sigma', R_B' \sigma') \in \text{He}(K_{\text{Solved}}).$$

By Theorem 2 (soundness), $P \models \text{ri}_{i, \sigma, \ldots, \sigma_n} (R_B \sigma', R_B' \sigma')$. Therefore, there exist recipes $R_i'$ for all $1 \leq i \leq n$ such that $l_i = \text{in}(d_i, t_i)$ such that $P \models k_{i, \sigma, \ldots, \sigma_n} (R_i', t_i \sigma)$. By Theorem 2 (completeness) and definition of $\text{He}$, there exist recipes $R_i$ such that $k_{i, \sigma, \ldots, \sigma_n} (R_i, t_i \sigma) \in \text{He}(K_{\text{Solved}})$ and $k_{i, \sigma, \ldots, \sigma_n} (R_i, t_i \sigma) \in \text{He}(K_{\text{Solved}})$.

Let $M_i = l_i$, if $l_i \in \{ \text{test, out} (c) \mid c \in C \}$ and let $M_i = \text{in}(d_i, t_i)$ if $l_i = \text{in}(d_i, t_i)$ for all $1 \leq i \leq n$ where the recipes $R_i'$ correspond to those computed during the algorithm. As $\text{REACH-IDENTITY}(K_{\text{Solved}}, P)$ holds there exists $Q_0' \in P$ such that

$$\text{Q} \models (Q_0', \psi_0') \Rightarrow (Q_1', \psi_1') \Rightarrow \ldots \Rightarrow (Q_n', \psi_n')$$

where $\psi_0' = \emptyset$.

Let $i$ be such that $l_i = \text{in}(d_i, t_i)$. Applying Lemma 29 to $f$ we have that for all $x \in \text{vars}(t_i)$ there exists $w_j$ such that $|w_j| < i$ and $x = x_j$. We have that $k_{\text{in}_{\tau, \tau}} (X_j, x_j) \tau, x_j \tau) \in \text{He}(K_{\text{Solved}})$, and $k_{\text{in}_{\tau, \tau}} (X_j, x_j) \tau, x_j \tau) \in \text{He}(K_{\text{Solved}})$ by choice of $f$, $\tau$ and $\beta$. By Theorem 2 (soundness), we obtain that $P \models k_{i, \sigma_1, \ldots, \sigma_{i-1}} (X_j, x_j \sigma, x_j \sigma') \tau, x_j \sigma' \tau)$. We rewrite it as $P \models k_{i, \sigma_1, \ldots, \sigma_{i-1}} (X_j, x_j \sigma, x_j \sigma') \tau, x_j \sigma^{-1} \tau) \tau, x_j \sigma^{-1} \tau)$. We already established that $P \models k_{i, \sigma_1, \ldots, \sigma_{i-1}} (R_i, t_i \sigma) \in \text{He}(K_{\text{Solved}})$ and therefore by Theorem 2 (soundness) we have that $P \models k_{i, \sigma_1, \ldots, \sigma_{i-1}} (R_i, t_i \sigma)$. We apply Lemma 31 to obtain that

$$P \models k_{i, \sigma_1, \ldots, \sigma_{i-1}} (R_i, t_i \sigma) \tau, t_i \sigma).$$

Now, we consider a mapping with $\text{dom}(\theta) = \{ c_1, \ldots, c_k \}$ such that $\theta = \theta''$ and $\theta''$ is defined as:

- $\theta''$ is the identity function; and $\theta''(c_i) = \sigma''^{-1}(c_i) \psi_{\text{dom}(c_i)} - \theta''^{-1}$ where $1 \leq j \leq n$, $\text{dom}(\theta) = \{ c_i \mid \text{id}(c_i) \leq j \}$ and $\text{id}(c_i) = \min\{ k \mid c_i \in \text{names}(M_k) \}$.

By applying Lemma 30 inductively on $n$ we have that

$$(Q_0', \psi_0') \Rightarrow (Q_1', \psi_1') \Rightarrow \ldots \Rightarrow (Q_n', \psi_n').$$

Now we show by induction on $n$ that

$$(Q_0', \psi_0') \Rightarrow (Q_1', \psi_1') \Rightarrow \ldots \Rightarrow (Q_n', \psi_n').$$
We assume by the induction hypothesis that 
\((Q'_i, \theta, \psi_i \theta) \xrightarrow{L_i} (Q'_i, \theta, \psi'_i \theta) \xrightarrow{L_{i-1}} \ldots \xrightarrow{L_1} (Q'_1, \theta, \psi'_{1-1} \theta)\)
and we show that 
\((Q'_{i-1}, \theta, \psi'_{i-1} \theta) \xrightarrow{L_i} (Q'_i, \theta, \psi'_i \theta)\).

We will show that \(L_i\) and \(M_i\sigma'^{-1}\tau\) satisfy the conditions of Lemma \([32]\) which allows us to conclude. Indeed, either \(L_i = M_i\sigma'^{-1}\tau\) (in the case of a test or out action), or \(L_i = \mathbf{in}(\pi_i, R'_i)\) and \(M_i\sigma'^{-1}\tau = \mathbf{in}(d_i, R_i\sigma'^{-1} \tau)\) (in the case of a \(\mathbf{in}\) action). In the second case, we need to show that 
\((R_i\sigma'^{-1} \tau = R'_i)\)(\(\psi'_{i-1} \theta).\) By the definition of \|=,\ we have that \(P |= k_i, \ldots, i_{i-1} (R_i, t_i \tau)\). We have previously shown in \([6]\) that \(P |= k_i, \ldots, i_{i-1} (R_i\sigma'^{-1} \tau, t_i \tau)\) and therefore \(P |= k_i, \ldots, i_{i-1} (R_i\sigma'^{-1} \tau, R'_i)\), or, equivalently, \((R_i\sigma'^{-1} \tau = R'_i)\)(\(\psi'_{i-1} \theta).\) By the hypothesis, we have that there exist \(Q \in \mathcal{P}\) such that 
\[(Q, \theta) \xrightarrow{L_1} (Q_1, \psi_1) \xrightarrow{L_2} \ldots \xrightarrow{L_{i-1}} (Q_{i-1}, \psi_{i-1} \theta),\]
and \((R_i\sigma'^{-1} \tau = R'_i)\)(\(\psi'_{i-1} \theta).\) By determinacy of \(Q\) it follows that \(\psi_{i-1} \approx_\ast \psi'_{i-1} \theta\) and therefore \((R_i\sigma'^{-1} \tau = R'_i)\)(\(\psi'_{i-1} \theta\) as well. As the hypothesis of Lemma \([32]\) are satisfied, we can conclude.

We have shown that \((\mathcal{P}, \theta) \xrightarrow{L_1 \ldots L_n} (Q'_n, \theta, \psi'_n \theta)\), therefore the Hypothesis \([1]\) cannot hold.

We know show that Hypothesis \([2]\) can not hold too.

As the equational theory is stable by substitution of terms for names and we know that \((R_B \sigma' =_E R'_B \sigma')\)(\(\psi'_n \theta),\) we deduce that \(\mathcal{E}(R_B \sigma'^{-1} \tau =_E R'_B \sigma'^{-1} \tau)\)(\(\psi'_n \theta).\)

We claim that \((R_B \sigma' \psi'_n \theta) = (R_B \beta \tau)\)(\(\psi'_n \theta).\) The proof is by induction on the size of \(R_B.\) The only interesting case is the case when \(R_B = X_i\) for some \(1 \leq i \leq m.\) In this case, we have that \((R_B \sigma' \psi'_n \theta) = c_j \theta\) where \(c_j = \sigma(x_j).\)

By construction of \(\theta, c_j \theta = \tau \circ \sigma'^{-1}(c_j)\)(\(\psi'_n \theta) and \(\tau \circ \sigma'^{-1}(c_j)\) is (by construction) \(X_i \beta \tau.\) Hence we get that \(X_i \sigma' \psi'_n \theta = (X_i \beta \tau)\)(\(\psi'_n \theta).\) Similarly, we can show that \((R_B \sigma' \psi'_n \theta) = (R'_B \beta \tau)\)(\(\psi'_n \theta).\) Hence, 
\((R_B \beta \tau)(\psi'_n \theta) =_E (R'_B \beta \tau)(\psi'_n \theta).\)

Now, thanks to determinacy, we have that \((R'_B \beta \tau =_E R'_B \beta \tau)\psi\) where \(\psi\) is the frame (up to static equivalence) reached by \(\mathcal{P}\) after \(L_1, \ldots, L_n.\)

We now show that:
\((R_B \beta \tau =_E R'_B \beta \tau)\psi\) implies \((R_B \tau =_E R'_B \tau)\psi.\)

Going back to the statement \(f,\) we consider all pairs \((i, j)\) such that \(x_i = x_j.\) We show that \((X_i \tau =_E X_j \tau)\psi.\) We distinguish three cases:

1) Case: \(W(\pi_i) < n\) and \(W(\pi_j) < n.\) In such a case, from \(\textbf{Corollary 1}\) we have \(i_{\pi_i, \pi_n} \ldots, i_{\pi_n} (X_i \tau, X_j \tau) \in \mathcal{H}_{e^{(}\kappa, n)}(K_{\text{solution}})\) for some \(k < n\) and \(\kappa.\) Hence we have \((X_i \tau =_E X_j \tau)\psi.\) By minimality of \(n\) we also have that \((X_i \tau =_E X_j \tau)\psi.\)

2) Case: \(W(\pi_i) = n\) and \(W(\pi_j) = n.\) In such a case, from \(\textbf{Corollary 1}\) we have \(i_{\pi_i, \pi_n} \ldots, i_{\pi_n} (X_i \tau, X_j \tau) \in \mathcal{H}_{e^{(}\kappa, n)}(K_{\text{solution}})\) with \(\kappa \leq S_n(\pi_i) + S_n(\pi_j).\) Hence we have \((X_i \tau =_E X_j \tau)\psi.\) Since \(S_n(\pi_i) \leq S(\pi_i) - 1\) (the root node is not counted) and \(S(\pi_i) + S(\pi_j) \leq S_n(\pi_m),\) we have \(\kappa < S_n(\pi_m).\) By minimality of our witness, we also have that \((X_i \tau =_E X_j \tau)\psi.\)

3) Case \(W(\pi_i) = n\) and \(W(\pi_j) < n.\) In such a case, from \(\textbf{Corollary 1}\) we have \(i_{\pi_i, \pi_n} \ldots, i_{\pi_n} (X_i \tau, X_j \tau) \in \mathcal{H}_{e^{(}\kappa, n)}(K_{\text{solution}})\) with \(\kappa < S_n(\pi_i) + S_n(\pi_j).\) Hence we have \((X_i \tau =_E X_j \tau)\psi.\) Since \(S_n(\pi_i) \leq S(\pi_i) - 1\) (the root node is not counted) and \(S(\pi_i) \leq S_n(\pi_m),\) we have \(\kappa < S_n(\pi_m).\) By minimality of our witness, we also have that \((X_i \tau =_E X_j \tau)\psi.\)

From the above equalities, we have that
\((R_B \beta \tau =_E R_B \beta \tau)\psi,\)
\((R'_B \beta \tau =_E R'_B \beta \tau)\psi.\)

Therefore, we get \((R_B \tau =_E R_B \tau)\psi\) and hence \((R_A =_E R'_A)\psi,\) thus obtaining a contradiction.

As both cases yield a contradiction, it follows that if \(P \not\subseteq \mathcal{P}\) then \textsc{reach-identity}(\(K_{\text{solution}}, \mathcal{P}\)) fails. \(\square\)