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

David Baelde, Stéphanie Delaune, Ivan Gazeau, Steve Kremer

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], [37]. 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

The research leading to these results has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreements No 645865-SPOOC and No 714955-POPSTAR), as well as from the French National Research Agency (ANR) under the project JCJC VIP n^o 11 JS02 006 01 and the project Sequoia.

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 [32] 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 V). 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-mentionned 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 impossibily 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 II, 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:

- $\mathcal{N}$ is a set of *names*, partitioned into the disjoint sets $\mathcal{N}_{\mathsf{prv}}$ and $\mathcal{N}_{\mathsf{pub}}$ of *private* and *public names*;
- $\mathcal{X}$ is the set of *message variables*, denoted x, y, etc.;
- $\mathcal{W} = \{\mathsf{w}_1, \mathsf{w}_2, \ldots\}$ is the set of parameters.

Intuitively, private names in $\mathcal{N}_{\mathsf{prv}}$ represent nonces or keys generated by honest participants, while public names in $\mathcal{N}_{\mathsf{pub}}$ 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 Σ (i.e., a finite set of function symbols together with their arity) and a set of atoms $\mathcal{A}$ we denote by $\mathcal{T}(\Sigma, \mathcal{A})$ the set of *terms*, defined as the smallest set that contains $\mathcal{A}$ and that is closed under application of function symbols in Σ. We denote by $vars(t)$ 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 σ to a term u is written $u\sigma$, and we denote $dom(\sigma)$ its *domain*, i.e. $dom(\sigma) = \{x \mid \sigma(x) \neq x\}$.

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 Σ. It consists of a finite set of equations of the form $M = N$ where $M, N \in \mathcal{T}(\Sigma, \mathcal{X})$, and induces an equivalence relation over terms: $=_\mathsf{E}$ 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 $\Sigma_{\mathsf{xor}} = \{\oplus, \mathsf{0}\}$, and the equational theory $\mathsf{E}_{\mathsf{xor}}$ below:

$$\begin{array}{rclcrcl} x \oplus x & = & \mathsf{0} & \quad & x \oplus (y \oplus z) & = & (x \oplus y) \oplus z \\ x \oplus \mathsf{0} & = & x & & x \oplus y & = & y \oplus x \end{array}$$

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 $\Sigma^+_{\mathsf{xor}} = \Sigma_{\mathsf{xor}} \uplus \{\langle \cdot, \cdot \rangle, \mathsf{proj}_1, \mathsf{proj}_2, \mathsf{h}\}$, and consider the equational theory $\mathsf{E}^+_{\mathsf{xor}}$ extending $\mathsf{E}_{\mathsf{xor}}$ with the equations $\mathsf{proj}_1(\langle x, y \rangle) = x$, and $\mathsf{proj}_2(\langle x, y \rangle) = y$. The symbol $\langle \cdot, \cdot \rangle$ models pairs; the proj_i symbols model projections; the unary symbol h models a hash function. Take $id \in \mathcal{N}_{\mathsf{pub}}$ to model the identity of a participant, and $r_1, r_2, k \in \mathcal{N}_{\mathsf{prv}}$ to represent two random numbers and a key, a priori unknown to the attacker. Let $t_0 = \langle id \oplus r_2,\ \mathsf{h}(\langle r_1, k \rangle) \oplus r_2 \rangle$. We have that $(\mathsf{proj}_1(t_0) \oplus id) \oplus \mathsf{proj}_2(t_0) =_{\mathsf{E}^+_{\mathsf{xor}}} \mathsf{h}(\langle r_1, k \rangle)$.

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

$$\mathsf{E} = \mathsf{E}_{\mathsf{xor}} \cup \{M = N \mid M, N \in \mathcal{T}(\Sigma \smallsetminus \Sigma_{\mathsf{xor}}, \mathcal{X})\}.$$

Hence, E models xor in combination with any other equational theory that is disjoint from $\mathsf{E}_{\mathsf{xor}}$.

B. Finite variant property

A *rewrite system* $\mathcal{R}$ is a set of rewrite rules of the form $\ell \to r$ where $\ell, r \in \mathcal{T}(\Sigma, \mathcal{X})$, and $vars(r) \subseteq vars(\ell)$. A term t can be rewritten in one step (modulo AC) to u, denoted $t \to_{\mathcal{R},\mathsf{AC}} u$, if there exists a position p in term t, a rule $\ell \to r$ in $\mathcal{R}$ and a substitution σ such that $t|_p =_{\mathsf{AC}} \ell\sigma$ and $u = t[r\sigma]_p$, 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 $t|_p$ with $r\sigma$. The relation $\to^*_{\mathcal{R},\mathsf{AC}}$ denotes the transitive and reflexive closure of $\to_{\mathcal{R},\mathsf{AC}}$.

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

$$u =_\mathsf{E} v \ \Leftrightarrow\ u{\downarrow}_{\mathcal{R},\mathsf{AC}} =_{\mathsf{AC}} v{\downarrow}_{\mathcal{R},\mathsf{AC}}.$$

Example 2: The equational theory $\mathsf{E}^+_{\mathsf{xor}}$ of Example 1 can be represented by the following AC-convergent system:

$$\mathcal{R}^+_{\mathsf{xor}} = \left\{ \begin{array}{cc} x \oplus (x \oplus y) \to y & \\ x \oplus \mathsf{0} \to x & \mathsf{proj}_1(\langle x, y \rangle) \to x \\ x \oplus x \to \mathsf{0} & \mathsf{proj}_2(\langle x, y \rangle) \to y \end{array} \right.$$

Let $t_0 = \langle id \oplus r_2, \mathsf{h}(\langle r_1, k \rangle) \oplus r_2 \rangle$. We have that:

$$\begin{array}{rl} & \mathsf{proj}_1(t_0) \oplus \mathsf{proj}_2(t_0) \\ \to_{\mathcal{R}^+_{\mathsf{xor}},\mathsf{AC}} & (id \oplus r_2) \oplus \mathsf{proj}_2(t_0) \\ \to_{\mathcal{R}^+_{\mathsf{xor}},\mathsf{AC}} & (id \oplus r_2) \oplus (\mathsf{h}(\langle r_1, k \rangle) \oplus r_2) \\ \to_{\mathcal{R}^+_{\mathsf{xor}},\mathsf{AC}} & id \oplus \mathsf{h}(\langle r_1, k \rangle) \end{array}$$

Note that the first rule in $\mathcal{R}^+_{\mathsf{xor}}$ is essential in last rewriting step above, and more generally for $\mathcal{R}^+_{\mathsf{xor}}$ to represent $\mathsf{E}_{\mathsf{xor}}$.

Given an AC-convergent rewrite system $\mathcal{R}$, we define *complete sets of variants*, first introduced in [22].

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

- $x\omega{\downarrow} =_{\mathsf{AC}} x\sigma{\downarrow}\tau$ for any $x \in vars(T)$, and
- $(t\omega){\downarrow} =_{\mathsf{AC}} (t\sigma){\downarrow}\tau$ for any $t \in T$.

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 σ 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 $\mathrm{variants}_{\mathcal{R},\mathsf{AC}}(t_1, \dots, t_n)$ instead of $\mathrm{variants}_{\mathcal{R},\mathsf{AC}}(\{t_1, \dots, t_n\})$.

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

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

- $\sigma_1 = \{x \mapsto y \oplus z\}$, $\sigma'_1 = \{y \mapsto x \oplus z\}$,
- $\sigma_2 = \{x \mapsto x' \oplus z, y \mapsto y' \oplus z\}$,
- $\sigma_3 = \{y \mapsto x\}$, and
- $\sigma_4 = \{x \mapsto \mathsf{0}\}$, $\sigma'_4 = \{y \mapsto \mathsf{0}\}$.

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 $\mathcal{R}$ and a set of equations $\Gamma = \{u_1 = v_1, \dots, u_k = v_k\}$. A set of substitutions S is a *complete set of* $\mathcal{R}, \mathsf{AC}$*-unifiers* for Γ if:

1) for each $\sigma \in S$ and $i \in \{1, \dots, k\}$, $u_i\sigma{\downarrow} =_{\mathsf{AC}} v_i\sigma{\downarrow}$;

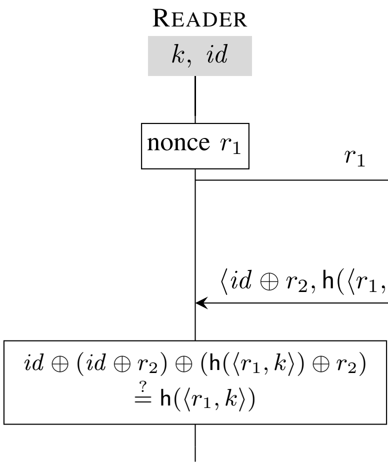

Fig. 1: The KCL protocol

2) for each θ such that $u_i\theta{\downarrow} =_{\mathsf{AC}} v_i\theta{\downarrow}$ for all $i \in \{1, \dots, k\}$, there exists $\sigma \in S$ and a substitution τ such that $x\theta{\downarrow} =_{\mathsf{AC}} x\sigma\tau{\downarrow}$ for any $x \in vars(\Gamma)$.

Such a set is denoted $\mathrm{csu}_{\mathcal{R},\mathsf{AC}}(\Gamma)$, or $\mathrm{csu}_{\mathsf{AC}}(\Gamma)$ when $\mathcal{R} = \emptyset$.

C. Process calculus

Let $\mathcal{C}h$ be a set of public channels. A protocol is modeled by a finite set of *processes* generated by the following grammar:

$$
\begin{array}{lcll}
P, P', P_1, P_2 & ::= & \mathbf{0} & \text{null process} \\
& & \mathbf{in}(c, x).P & \text{input} \\
& & \mathbf{out}(c, t).P & \text{output} \\
& & [s = t].P & \text{test}
\end{array}
$$

where $x \in \mathcal{X}, s, t \in \mathcal{T}(\Sigma, \mathcal{N} \cup \mathcal{X})$, and $c \in \mathcal{C}h$.

As usual, a receive action $\mathbf{in}(c, x)$ acts as a binding construct for the variable x. We assume the usual definitions of *free* and *bound variables* for processes. We also assume that each variable is bound at most once. A process is *ground* if it does not contain any free variables. For sake of conciseness, we sometimes omit the null process at the end of a process.

Following [18], we only consider a minimalistic core calculus that does not include parallel composition. Given that we only consider a bounded number of sessions (i.e., a process calculus without replication) 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 this paper, a protocol is simply a finite set of ground processes.

Example 4: Following a description given in [41], we consider the RFID protocol depicted in Figure 1. The reader and the tag id share the secret key k. The reader starts by sending a nonce r_1. The tag generates a nonce r_2 and computes the message $t_0 = \langle id \oplus r_2,\ \mathsf{h}(\langle r_1, k\rangle) \oplus r_2\rangle$ introduced in Example 1. When receiving such a message, the reader will be able to retrieve r_2 from the first component, and by xoring it with the second component, he obtains $\mathsf{h}(\langle r_1, k\rangle)$.

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

$$
\begin{array}{l}
P_{\mathsf{tag}} = \mathbf{in}(c, x).\ \mathbf{out}(c, \langle id \oplus r_2,\ \mathsf{h}(\langle x, k\rangle) \oplus r_2\rangle).\ \mathbf{0} \\
P_{\mathsf{reader}} = \mathbf{out}(c, r_1).\ \mathbf{in}(c, y). \\
\qquad [(\mathsf{proj}_1(y) \oplus id) \oplus \mathsf{proj}_2(y) = \mathsf{h}(\langle r_1, k\rangle)].\ \mathbf{0}
\end{array}
$$

where $r_1, r_2, k \in \mathcal{N}_{\mathsf{prv}}$, $id \in \mathcal{N}_{\mathsf{pub}}$, 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.

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 $t_1, \dots, t_\ell \in \mathcal{T}(\Sigma, \mathcal{N})$. Such a sequence is organised into a *frame* $\varphi = \{\mathsf{w}_1 \mapsto t_1, \dots, \mathsf{w}_\ell \mapsto t_\ell\}$ that is a substitution of *size* $|\varphi| = \ell$.

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

Intuitively, an attacker is able to deduce new messages by applying function symbols in Σ to names in $\mathcal{N}_{\mathsf{pub}}$ and terms stored in φ. The term R is called a *recipe*.

Example 5: Continuing Example 4, consider the frame

$$\varphi = \{\mathsf{w}_1 \mapsto r_1,\ \mathsf{w}_2 \mapsto \langle id \oplus r_2,\ \mathsf{h}(\langle r_1, k\rangle) \oplus r_2\rangle\}.$$

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

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

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

$$
\begin{array}{lll}
\textsc{Recv} & (\mathbf{in}(c, x).P, \varphi) \xrightarrow{\mathbf{in}(c,R)} (P\{x \mapsto t{\downarrow}\}, \varphi) & \text{if } \varphi \vdash_R t \\
\textsc{Send} & (\mathbf{out}(c, t).P, \varphi) \xrightarrow{\mathbf{out}(c)} (P, \varphi \cup \{\mathsf{w}_{|\varphi|+1} \mapsto t{\downarrow}\}) & \\
\textsc{Test} & ([s = t].P, \varphi) \xrightarrow{\mathbf{test}} (P, \varphi) & \text{if } s{\downarrow} =_{\mathsf{AC}} t{\downarrow}
\end{array}
$$

The label $\mathbf{in}(c, R)$ indicates the input of a message sent by the attacker over the channel c where R is the recipe that the attacker uses to construct this message. The label $\mathbf{out}(c)$ indicates a message sent over channel c, and transition rule Send records the message sent in the frame. Finally, the rule Test checks equality of s and t in the equational theory and is labelled by the unobservable action **test**.

$$
\begin{aligned}
(P_{\mathsf{diff}}, \emptyset) &\xrightarrow{\mathbf{in}(c,r_1)} (\mathbf{out}(c, \langle id \oplus r_2, \mathsf{h}(\langle r_1, k\rangle) \oplus r_2\rangle).P'_{\mathsf{tag}}, \emptyset)\\
&\xrightarrow{\mathbf{out}(c)} (P'_{\mathsf{tag}}, \{\mathsf{w}_1 \mapsto \langle id \oplus r_2, \mathsf{h}(\langle r_1, k\rangle) \oplus r_2\rangle\})\\
&\xrightarrow{\mathbf{in}(c,r_1)} (\mathbf{out}(c, \langle id' \oplus r'_2, \mathsf{h}(\langle r_1, k'\rangle) \oplus r'_2\rangle).\mathbf{0}, \{\mathsf{w}_1 \mapsto \langle id \oplus r_2, \mathsf{h}(\langle r_1, k\rangle) \oplus r_2\rangle\})\\
&\xrightarrow{\mathbf{out}(c)} (\mathbf{0}, \{\mathsf{w}_1 \mapsto \langle id \oplus r_2, \mathsf{h}(\langle r_1, k\rangle) \oplus r_2\rangle,\ \mathsf{w}_2 \mapsto \langle id' \oplus r'_2, \mathsf{h}(\langle r_1, k'\rangle) \oplus r'_2\rangle\})
\end{aligned}
$$

$$
(P_{\mathsf{same}}, \emptyset) \xrightarrow{\mathbf{in}(c,r_1),\mathbf{out}(c),\mathbf{in}(c,r_1),\mathbf{out}(c)} (\mathbf{0}, \{\mathsf{w}_1 \mapsto \langle id \oplus r_2, \mathsf{h}(\langle r_1, k\rangle) \oplus r_2\rangle,\ \mathsf{w}_2 \mapsto \langle id \oplus r'_2, \mathsf{h}(\langle r_1, k\rangle) \oplus r'_2\rangle\})
$$

Fig. 2: Some derivations (see Example 6)

Example 6: Consider the ground process P_{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., $P_{\mathsf{diff}} = P_{\mathsf{tag}}.P'_{\mathsf{tag}}$ where:

$$
\begin{aligned}
P_{\mathsf{tag}} &= \mathbf{in}(c, x).\mathbf{out}(c, \langle id \oplus r_2, \mathsf{h}(\langle x, k\rangle) \oplus r_2\rangle)\\
P'_{\mathsf{tag}} &= \mathbf{in}(c, x').\mathbf{out}(c, \langle id' \oplus r'_2, \mathsf{h}(\langle x', k'\rangle) \oplus r'_2\rangle)
\end{aligned}
$$

We also consider the ground process P_{same} obtained from P_{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 $r_1 \in \mathcal{N}_{\mathsf{pub}}$, i.e. r_1 is a public name known by the attacker.

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

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 φ be a frame and R_1, R_2 be two terms in $\mathcal{T}(\Sigma, \mathcal{N}_{\mathsf{pub}} \cup dom(\varphi))$. The *test* $R_1 \overset{?}{=} R_2$ *holds* on frame φ, written $(R_1 = R_2)\varphi$, if $R_1\varphi\!\downarrow =_{\mathsf{AC}} R_2\varphi\!\downarrow$.

Example 7: Consider the frames $\varphi_{\mathsf{diff}} = \{\mathsf{w}_1 \mapsto t, \mathsf{w}_2 \mapsto t'\}$ and $\varphi_{\mathsf{same}} = \{\mathsf{w}_1 \mapsto t, \mathsf{w}_2 \mapsto t''\}$ where the terms t, t', and t'' are as follows:

- $t = \langle id \oplus r_2, \mathsf{h}(\langle r_1, k\rangle) \oplus r_2\rangle$,
- $t' = \langle id' \oplus r'_2, \mathsf{h}(\langle r_1, k'\rangle) \oplus r'_2\rangle$,
- $t'' = \langle id \oplus r'_2, \mathsf{h}(\langle r_1, k\rangle) \oplus r'_2\rangle$.

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

$$\mathsf{proj}_1(\mathsf{w}_1) \oplus \mathsf{proj}_2(\mathsf{w}_1) \overset{?}{=} \mathsf{proj}_1(\mathsf{w}_2) \oplus \mathsf{proj}_2(\mathsf{w}_2)$$

holds in φ_{same} but not in φ_{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 $\mathcal{Q}$, denoted $\mathcal{P} \sqsubseteq \mathcal{Q}$, if whenever $(\mathcal{P}, \emptyset) \overset{\ell_1,\dots,\ell_n}{\Longrightarrow} (P, \varphi)$ and $(R_1 = R_2)\varphi$, then there exists a configuration (Q', φ') such that $(\mathcal{Q}, \emptyset) \overset{\ell_1,\dots,\ell_n}{\Longrightarrow} (Q', \varphi')$ and $(R_1 = R_2)\varphi'$. We say that $\mathcal{P}$ and $\mathcal{Q}$ are *equivalent*, written $\mathcal{P} \approx \mathcal{Q}$, if $\mathcal{P} \sqsubseteq \mathcal{Q}$ and $\mathcal{Q} \sqsubseteq \mathcal{P}$.

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 [18].

Definition 6: We say that a protocol $\mathcal{P}$ is *determinate* if whenever $(\mathcal{P}, \emptyset) \overset{\ell_1,\dots,\ell_n}{\Longrightarrow} (P, \varphi)$, and $(\mathcal{P}, \emptyset) \overset{\ell_1,\dots,\ell_n}{\Longrightarrow} (P', \varphi')$, then for any test $R_1 \overset{?}{=} R_2$, 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 $\mathcal{P}_{\mathsf{same}} = \{P_{\mathsf{same}}\}$ and $\mathcal{P}_{\mathsf{diff}} = \{P_{\mathsf{diff}}\}$ 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 $\mathcal{P}_{\mathsf{same}} \not\sqsubseteq \mathcal{P}_{\mathsf{diff}}$. Indeed, we have shown that:

- $(P_{\mathsf{same}}, \emptyset) \xrightarrow{\mathbf{in}(c,r_1),\mathbf{out}(c),\mathbf{in}(c,r_1),\mathbf{out}(c)} (\mathbf{0}, \varphi_{\mathsf{same}})$; and
- $(\mathsf{proj}_1(\mathsf{w}_1) \oplus \mathsf{proj}_2(\mathsf{w}_1) {=} \mathsf{proj}_1(\mathsf{w}_2) \oplus \mathsf{proj}_2(\mathsf{w}_2))\varphi_{\mathsf{same}}$.

However, the only extended trace (P', φ') such that

$$(P_{\mathsf{diff}}, \emptyset) \xrightarrow{\mathbf{in}(c,r_1),\mathbf{out}(c),\mathbf{in}(c,r_1),\mathbf{out}(c)} (P', \varphi')$$

is $(\mathbf{0}, \varphi_{\mathsf{diff}})$ and we have seen that $\mathsf{proj}_1(\mathsf{w}_1) \oplus \mathsf{proj}_2(\mathsf{w}_1) \overset{?}{=} \mathsf{proj}_1(\mathsf{w}_2) \oplus \mathsf{proj}_2(\mathsf{w}_2)$ does not hold in φ_{diff} (see Example 7).

However, we have that $\mathcal{P}_{\mathsf{diff}} \sqsubseteq \mathcal{P}_{\mathsf{same}}$. This is a non trivial inclusion that has been checked using our tool.

III. MODELLING USING HORN CLAUSES

Our procedure is based on a fully abstract modelling of a process in first-order Horn clauses.

$$
\begin{array}{ll}
(P_0,\varphi_0) \models \mathsf{r}_{\ell_1,\ldots,\ell_n} & \text{if } (P_0,\varphi_0) \xrightarrow{L_1} (P_1,\varphi_1) \ldots \xrightarrow{L_n} (P_n,\varphi_n) \text{ such that } \ell_i{\downarrow} =_{\mathsf{AC}} L_i\varphi_{i-1}{\downarrow} \text{ for all } 1 \le i \le n \\
(P_0,\varphi_0) \models \mathsf{k}_{\ell_1,\ldots,\ell_n}(R,t) & \text{if when } (P_0,\varphi_0) \xrightarrow{L_1} (P_1,\varphi_1) \xrightarrow{L_2} \ldots \xrightarrow{L_n} (P_n,\varphi_n) \\
& \text{such that } \ell_i{\downarrow} =_{\mathsf{AC}} L_i\varphi_{i-1}{\downarrow} \text{ for all } 1 \le i \le n, \text{ then } \varphi_n \vdash_R t \\
(P_0,\varphi_0) \models \mathsf{i}_{\ell_1,\ldots,\ell_n}(R,R') & \text{if there exists } t \text{ such that } (P_0,\varphi_0) \models \mathsf{k}_{\ell_1,\ldots,\ell_n}(R,t) \text{ and } (P_0,\varphi_0) \models \mathsf{k}_{\ell_1,\ldots,\ell_n}(R',t) \\
(P_0,\varphi_0) \models \mathsf{ri}_{\ell_1,\ldots,\ell_n}(R,R') & \text{if } (P_0,\varphi_0) \models \mathsf{r}_{\ell_1,\ldots,\ell_n} \text{ and } (P_0,\varphi_0) \models \mathsf{i}_{\ell_1,\ldots,\ell_n}(R,R')
\end{array}
$$

Fig. 3: Semantics of atomic formulas

A. Predicates

We define *symbolic runs*, denoted u, v, w, as finite sequences of symbolic labels

$$\ell \in \{\mathbf{in}(c,t), \mathbf{out}(c), \mathbf{test} \mid t \in \mathcal{T}(\Sigma, \mathcal{N} \cup \mathcal{X}), c \in \mathcal{C}h\}.$$

The empty sequence is denoted ϵ. Intuitively, a symbolic run stands for a set of possible runs of the protocol. We denote $u \sqsubseteq_{\mathsf{AC}} v$ when u is a prefix (modulo AC) of v.

We assume a set $\mathcal{Y}$ of *recipe variables* disjoint from $\mathcal{X}$, and we use capital letters X, Y, Z to range over $\mathcal{Y}$. We assume that such variables may only be substituted by terms in $\mathcal{T}(\Sigma, \mathcal{N}_{\mathsf{pub}} \cup \mathcal{W} \cup \mathcal{Y})$. 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 $\mathcal{T}(\Sigma, \mathcal{N}_{\mathsf{pub}} \cup \mathcal{W} \cup \mathcal{Y})$, and t is a term in $\mathcal{T}(\Sigma, \mathcal{N} \cup \mathcal{X})$. Intuitively:

- *reachability predicate*: r_w holds when the run represented by w is executable;
- *intruder knowledge predicate*: $\mathsf{k}_w(R,t)$ 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*: $\mathsf{i}_w(R,R')$ holds if whenever the run w is executable, R and R' are recipes for the same term; and
- *reachable identity predicate*: $\mathsf{ri}_w(R,R')$ is a shortcut for $\mathsf{r}_w \wedge \mathsf{i}_w(R,R')$.

A (ground) atomic formula is interpreted over a pair consisting of a process P and a frame φ, and we write $(P,\varphi) \models f$ when the atomic formula f holds for (P,φ) or simply $P \models f$ when φ 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 $\mathcal{T}(\Sigma, \mathcal{N})$, while recipe variables in $\mathcal{Y}$ are mapped to recipes, i.e. terms in $\mathcal{T}(\Sigma, \mathcal{N}_{\mathsf{pub}} \cup \mathcal{W})$.

Example 9: Continuing our running example, let $w = \mathbf{in}(c,r_1), \mathbf{out}(c), \mathbf{in}(c,r_1), \mathbf{out}(c)$, $t_0 = id \oplus \mathsf{h}(\langle r_1, k\rangle)$, and $R_i = \mathsf{proj}_1(\mathsf{w}_i) \oplus \mathsf{proj}_2(\mathsf{w}_i)$ with $i \in \{1,2\}$. We have that:

$$(P_{\mathsf{same}}, \emptyset) \models \mathsf{r}_w \wedge \mathsf{k}_w(R_1,t_0) \wedge \mathsf{k}_w(R_2,t_0) \wedge \mathsf{ri}_w(R_1,R_2).$$

Consider $t = \langle id \oplus r_2, \mathsf{h}(\langle x,k\rangle) \oplus r_2\rangle$ and the formulas

- $f_1 = \forall X, x.\ \mathsf{r}_{\mathbf{in}(c,x),\mathbf{out}(c)} \Leftarrow \mathsf{k}_\epsilon(X,x)$;
- $f_2 = \forall X, x.\ \mathsf{k}_{\mathbf{in}(c,x),\mathbf{out}(c)}(\mathsf{w}_1,t) \Leftarrow \mathsf{k}_\epsilon(X,x)$.

We have that $P_{\mathsf{same}} \models f_1$ and $P_{\mathsf{same}} \models f_2$.

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 $H \Leftarrow \mathsf{k}_{u_1}(X_1,t_1), \ldots, \mathsf{k}_{u_n}(X_n,t_n)$ where:

- $H \in \{\mathsf{r}_{u_0},\ \mathsf{k}_{u_0}(R,t),\ \mathsf{i}_{u_0}(R,R'),\ \mathsf{ri}_{u_0}(R,R')\}$;
- $u_0, u_1, \ldots, u_n$ are symbolic runs such that $u_i \sqsubseteq_{\mathsf{AC}} u_0$ for any $i \in \{1,\ldots,n\}$;
- $t, t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{N} \cup \mathcal{X})$;
- $R, R' \in \mathcal{T}(\Sigma, \mathcal{N}_{\mathsf{pub}} \cup \mathcal{W} \cup \mathcal{Y})$; and
- $X_1, \ldots, X_n$ are distinct variables from $\mathcal{Y}$.

Lastly, $vars(t) \subseteq vars(t_1,\ldots,t_n)$ when $H = \mathsf{k}_{u_0}(R,t)$.

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 σ a grounding substitution for a statement $H \Leftarrow B_1,\ldots,B_n$ when σ is grounding for each of the atomic formulas $H, B_1, \ldots, B_n$. The *skeleton* of a statement f, denoted $\mathsf{skel}(f)$, is the statement where recipes are removed.

$$
\begin{array}{l}
\mathsf{r}_{\ell_1\sigma\tau{\downarrow},\ldots,\ell_m\sigma\tau{\downarrow}} \Leftarrow \{\mathsf{k}_{\ell_1\sigma\tau{\downarrow},\ldots,\ell_{j-1}\sigma\tau{\downarrow}}(X_j, x_j\sigma\tau{\downarrow})\}_{j\in R(m)} \\
\qquad \text{for all } 0 \le m \le n \\
\qquad \text{for all } \sigma \in \mathrm{csu}_{\mathcal{R},\mathsf{AC}}(\{s_k = t_k\}_{k\in T(m)}) \\
\qquad \text{for all } \tau \in \mathrm{variants}_{\mathcal{R},\mathsf{AC}}(\ell_1\sigma,\ldots,\ell_m\sigma) \\[1ex]
\mathsf{k}_{\ell_1\sigma\tau{\downarrow},\ldots,\ell_m\sigma\tau{\downarrow}}(\mathsf{w}_{|S(m)|}, t_m\sigma\tau{\downarrow}) \Leftarrow \\
\qquad\qquad \{\mathsf{k}_{\ell_1\sigma\tau{\downarrow},\ldots,\ell_{j-1}\sigma\tau{\downarrow}}(X_j, x_j\sigma\tau{\downarrow})\}_{j\in R(m)} \\
\qquad \text{for all } m \in S(n) \\
\qquad \text{for all } \sigma \in \mathrm{csu}_{\mathcal{R},\mathsf{AC}}(\{s_k = t_k\}_{k\in T(m)}) \\
\qquad \text{for all } \tau \in \mathrm{variants}_{\mathcal{R},\mathsf{AC}}(\ell_1\sigma,\ldots,\ell_m\sigma, t_m\sigma) \\[1ex]
\mathsf{k}_\epsilon(c,c) \Leftarrow \\
\qquad \text{for all public names } c \in \mathcal{N}^0_{\mathsf{pub}} \\[1ex]
\mathsf{k}_{\ell_1,\ldots,\ell_m}(\mathsf{f}(Y_1,\ldots,Y_k), \mathsf{f}(y_1,\ldots,y_k)\tau{\downarrow}) \Leftarrow \\
\qquad\qquad \{\mathsf{k}_{\ell_1,\ldots,\ell_m}(Y_j, y_j\tau{\downarrow})\}_{j\in\{1,\ldots,k\}} \\
\qquad \text{for all } 0 \le m \le n \\
\qquad \text{for all function symbols } \mathsf{f} \text{ of arity } k \\
\qquad \text{for all } \tau \in \mathrm{variants}_{\mathcal{R},\mathsf{AC}}(\mathsf{f}(y_1,\ldots,y_k)).
\end{array}
$$

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

$$
\begin{aligned}
f_0^+ :\quad & \mathsf{k}_w(X_1 \oplus X_2, x_1 \oplus x_2) \Leftarrow \mathsf{k}_w(X_1, x_1), \mathsf{k}_w(X_2, x_2) \\
f_1^+ :\quad & \mathsf{k}_w(X_1 \oplus X_2, x_1) \Leftarrow \mathsf{k}_w(X_1, x_1 \oplus x_2), \mathsf{k}_w(X_2, x_2) \\
f_2^+ :\quad & \mathsf{k}_w(X_1 \oplus X_2, x_1 \oplus x_2) \Leftarrow \\
& \qquad\qquad \mathsf{k}_w(X_1, x_1 \oplus x_3), \mathsf{k}_w(X_2, x_2 \oplus x_3) \\
f_3^+ :\quad & \mathsf{k}_w(X_1 \oplus X_2, 0) \Leftarrow \mathsf{k}_w(X_1, x), \mathsf{k}_w(X_2, x) \\
f_4^+ :\quad & \mathsf{k}_w(X_1 \oplus X_2, x) \Leftarrow \mathsf{k}_w(X_1, x), \mathsf{k}_w(X_2, 0)
\end{aligned}
$$

where $w = \ell_1, \ldots, \ell_m$ is as defined in Section III-B.

Fig. 5: Definition of f_0^+ and its variants.

give a set of statements $\mathsf{seed}(P)$ which will serve as a starting point for the modelling. We shall also establish that the set of statements $\mathsf{seed}(P)$ is a sound and (partially) complete abstraction of the ground process P. In order to formally define $\mathsf{seed}(P)$, we start by fixing some conventions.

Let $P = \mathsf{a}_1.\mathsf{a}_2.\ldots.\mathsf{a}_n$ be a ground process. We assume w.l.o.g. the following naming conventions:

1) if a_i is a receive action then $\mathsf{a}_i = \ell_i = \mathbf{in}(c_i, x_i)$;
2) if a_i is a send action then $\mathsf{a}_i = \mathbf{out}(c_i, t_i)$, $\ell_i = \mathbf{out}(c_i)$;
3) if a_i is a test actions then $\mathsf{a}_i = [s_i = t_i]$, and $\ell_i = \mathbf{test}$.

Moreover, we assume that $x_i \neq x_j$ for any $i \neq j$.

For each $0 \leq m \leq n$, let the sets $R(m)$, $S(m)$ and $T(m)$ respectively denote the set of indices of the receive, send and test actions amongst $\mathsf{a}_1, \ldots, \mathsf{a}_m$. Moreover, we denote by $|S|$ the number of elements in such a set. Formally,

- $R(m) = \{\ i \ \mid\ 1 \leq i \leq m \text{ and } \mathsf{a}_i = \mathbf{in}(c_i, x_i)\ \}$;
- $S(m) = \{\ i \ \mid\ 1 \leq i \leq m \text{ and } \mathsf{a}_i = \mathbf{out}(c_i, t_i)\ \}$;
- $T(m) = \{\ i \ \mid\ 1 \leq i \leq m \text{ and } \mathsf{a}_i = [s_i = t_i]\ \}$.

Given a set of public names $\mathcal{N}^0_{\mathsf{pub}} \subseteq \mathcal{N}_{\mathsf{pub}}$, the *set of seed statements* associated to P and $\mathcal{N}^0_{\mathsf{pub}}$, denoted $\mathsf{seed}(P, \mathcal{N}^0_{\mathsf{pub}})$, is defined in Figure 4. We may note that while constructing the set of seed statements, we compute a complete set of unifiers modulo the whole equational theory E w.r.t. all tests. In addition, we also apply finite variants. This allows us to get rid of the rewriting theory in the remainder of our procedure.

The first kind of seed statement models the fact that the run represented by $\ell_1\sigma\tau{\downarrow}, \ldots, \ell_m\sigma\tau{\downarrow}$ is executable as soon as the attacker is able to feed each input with terms that will allow one to successfully pass all the tests. Following the same idea, under the same hypotheses, the attacker will be able to learn the output term. The two last families of statements model the deduction capabilities of the attacker who knows all the public names, and is able to apply a function symbol on top of terms that he already knows. These abilities can be used at any stage. However, since we will give the attacker the abilities to transfer his knowledge, we only have to express the fact that he knows public names initially.

If $\mathcal{N}^0_{\mathsf{pub}} = \mathcal{N}_{\mathsf{pub}}$, then $\mathsf{seed}(P, \mathcal{N}_{\mathsf{pub}})$ is said to be the set of seed statements associated to P and in this case we write $\mathsf{seed}(P)$ as a shortcut for $\mathsf{seed}(P, \mathcal{N}_{\mathsf{pub}})$.

Example 10: Continuing our running example, let

- $u = \mathbf{in}(c, x).\mathbf{out}(c)$, $v = \mathbf{in}(c, x').\mathbf{out}(c)$;
- $t = \langle id \oplus r_2, \mathsf{h}(\langle x, k\rangle) \oplus r_2\rangle$;
- $t' = \langle id \oplus r_2, \mathsf{h}(\langle x', k\rangle) \oplus r_2\rangle$.

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

$$
\begin{aligned}
\mathsf{r}_u &\Leftarrow \mathsf{k}_\epsilon(X, x) & \mathsf{r}_{uv} &\Leftarrow \mathsf{k}_w(X', x'), \mathsf{k}_\epsilon(X, x) \\
\mathsf{k}_u(\mathsf{w}_1, t) &\Leftarrow \mathsf{k}_\epsilon(X, x) & \mathsf{k}_{uv}(\mathsf{w}_2, t') &\Leftarrow \mathsf{k}_w(X', x'), \mathsf{k}_\epsilon(X, x)
\end{aligned}
$$

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

$$
\begin{aligned}
\mathsf{k}_\epsilon(\mathsf{proj}_1(X), \mathsf{proj}_1(x)) &\Leftarrow \mathsf{k}_\epsilon(X, x) \\
\mathsf{k}_\epsilon(\mathsf{proj}_1(X), x_1) &\Leftarrow \mathsf{k}_\epsilon(X, \langle x_1, x_2\rangle)
\end{aligned}
$$

Figure 5 shows those obtained for $\mathsf{f} = \oplus$ and an arbitrary w.

C. Soundness and completeness

We show that as far as reachability and intruder knowledge predicates are concerned, the set $\mathsf{seed}(P)$ 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:

$$
\textsc{Conseq}\ \frac{\begin{array}{c} f = \Big(H \Leftarrow B_1, \ldots, B_n\Big) \in K \\ \sigma \text{ grounding for } f \text{ with } \mathsf{skel}(f\sigma) \text{ in normal form} \\ B_1\sigma \in \mathcal{H}(K),\ \ldots,\ B_n\sigma \in \mathcal{H}(K) \end{array}}{H\sigma \in \mathcal{H}(K)}
$$

$$
\textsc{Extend}\ \frac{\mathsf{k}_u(R, t) \in \mathcal{H}(K)}{\mathsf{k}_{uv}(R, t) \in \mathcal{H}(K)}
$$

Theorem 1: Let P be a ground process.

- $P \models f$ for any $f \in \mathsf{seed}(P) \cup \mathcal{H}(\mathsf{seed}(P))$;
- If $(P, \emptyset) \xrightarrow{L_1, \ldots, L_m} (Q, \varphi)$ for some (Q, φ), then
  1) $\mathsf{r}_{L_1\varphi\downarrow, \ldots, L_m\varphi\downarrow} \in \mathcal{H}(\mathsf{seed}(P))$; and
  2) if $\varphi \vdash_R t$ then $\mathsf{k}_{L_1\varphi\downarrow, \ldots, L_m\varphi\downarrow}(R, t{\downarrow}) \in \mathcal{H}(\mathsf{seed}(P))$.

We will see that the completeness of $\mathsf{seed}(P)$ 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 $\mathcal{P}$ 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 $f = (H \Leftarrow B_1, \ldots, B_n)$,

- f is said to be *solved* if for all $1 \leq i \leq n$, we have that $B_i = \mathsf{k}_{\ell_1,\dots,\ell_{j_i}}(X_i, x_i)$ 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 $H = \mathsf{k}_{\ell_1,\dots,\ell_k}(R, t)$, we have that $t \notin \mathcal{X}$.

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

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 $\mathsf{sat}(K)$ 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):

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

where $h = \Big((H \Leftarrow B_1, \dots, B_n, B_{n+1}, \dots, B_m)\sigma\Big)$.

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 $\uplus$ simply adds h to K. A naive approach to add the xor operator consists in replacing the condition $\sigma = \mathrm{mgu}(\mathsf{k}_u(X,t), \mathsf{k}_w(R,t'))$ by $\sigma \in \mathrm{csu}_{\mathsf{AC}}(\mathsf{k}_u(X,t), \mathsf{k}_w(R,t'))$, 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 $P = \mathbf{in}(c, z_1).\mathbf{in}(c, z_2).[z_2 = a \oplus z_1].0$ and $w = \mathbf{in}(c, z_1), \mathbf{in}(c, a \oplus z_1)$. The set $\mathsf{seed}(P)$ contains (among others) the following statements:

$$\begin{array}{rcl} \mathsf{r}_{w,\mathbf{test}} & \Leftarrow & \mathsf{k}_\epsilon(Z_1, z_1), \mathsf{k}_w(Z_2, a \oplus z_1) \\ \mathsf{k}_w(X \oplus Y, x \oplus y) & \Leftarrow & \mathsf{k}_w(X, x), \mathsf{k}_w(Y, y) \end{array}$$

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

$$\sigma = \{x \mapsto a \oplus z_{11},\ y \mapsto z_{12},\ z_1 \mapsto z_{11} \oplus z_{12}\}$$

resulting in the following statement:

$$\mathsf{r}_{w\sigma} \Leftarrow \mathsf{k}_\epsilon(Z_1, z_{11} \oplus z_{12}), \mathsf{k}_{w\sigma}(X, a \oplus z_{11}), \mathsf{k}_{w\sigma}(Y, z_{12}).$$

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 [18]. 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 $\mathsf{k}_w(X,t)$ 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

$$H \Leftarrow B_1, \dots B_n \parallel \mathcal{M}$$

where $\mathcal{M} \subseteq \{B_1, \dots, B_n\}$ 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 RESOLUTION+, i.e., when performing resolution against a statement in f_0^+: to decide which of the two new hypotheses has to be marked we rely on the following notions.

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

When performing 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.

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

$$\mathsf{r}_{w\sigma} \Leftarrow \mathsf{k}_\epsilon(Z_1, z_{11} \oplus z_{12}), \mathsf{k}_{w\sigma}(X, a \oplus z_{11}), \mathsf{k}_{w\sigma}(Y, z_{12}) \parallel \mathcal{M}$$

where $\mathcal{M} = \{\mathsf{k}_{w\sigma}(X, a \oplus z_{11})\}$.

This forbids the use of RESOLUTION+ on $\mathsf{k}_{w\sigma}(X, a \oplus z_{11})$. Provided that our function sel returns $\mathsf{k}_{w\sigma}(X, a \oplus z_{11})$, the saturation procedure can now only do a RESOLUTION rule on the next statement and therefore the non termination issue mentioned in Example 11 is avoided.

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 RESOLUTION+ rule between f_1^+ (and f_0^+) on the atom $\mathsf{k}_w(X_1, x_1 \oplus x_2)$. However, the term $x_1 \oplus x_2$ has no rigid factor, and among the resulting statements, we will find the following one:

$$\begin{aligned}\mathsf{k}_{w\sigma}(X_1 \oplus X_2 \oplus X_3, x_{11} \oplus x_{12}) \Leftarrow\ & \mathsf{k}_{w\sigma}(X_1, x_{11} \oplus x_{21}),\\ & \mathsf{k}_{w\sigma}(X_2, x_{12} \oplus x_{22}),\\ & \mathsf{k}_{w\sigma}(X_3, x_{21} \oplus x_{22}) \parallel \emptyset\end{aligned}$$

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

Finally, the RESOLUTION and 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 VIP *statement*, and they will deserve a privileged treatment in the update.

2) Update: We will now define the update operator $\Cup$ 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.

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

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

$$\text{RES}\ \frac{\begin{array}{c}\mathsf{k}_u(R,t) \Leftarrow B_1, \ldots, B_n \in K \qquad \sigma \text{ a substitution}\\ B_i\sigma \Leftarrow C_1, \ldots, C_m \in \mathbf{conseq}(K)\ \ 1 \le i\ \le n\end{array}}{\mathsf{k}_{uv}(R,t)\sigma \Leftarrow C_1, \ldots, C_m \in \mathbf{conseq}(K)}$$

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.

Definition 12: The canonical form $f\Downarrow$ of a statement

$$f = \Big(H \Leftarrow B_1, \ldots, B_n \parallel \mathcal{M}\Big)$$

is the statement obtained by first normalizing all the recipes, then applying the rule REMOVE as many times as possible, and for solved deduction statement, applying the rule SHIFT as many times as possible.

$$\text{REMOVE}\ \frac{\begin{array}{c}H \Leftarrow \mathsf{k}_{uv}(X,t), \mathsf{k}_u(Y,t), B_1, \ldots, B_n \parallel \mathcal{M}\\ \text{with } X \notin vars(H)\end{array}}{H \Leftarrow \mathsf{k}_u(Y,t), B_1, \ldots, B_n \parallel \mathcal{M} \setminus \mathsf{k}_{uv}(X,t)}$$

$$\text{SHIFT}\ \frac{\mathsf{k}_{uv}(R,t) \Leftarrow \mathsf{k}_u(X,x), B_1, \ldots, B_n \text{ with } x \in \mathsf{factor}(t)}{\mathsf{k}_{uv}(R \oplus X, t \oplus x\downarrow) \Leftarrow \mathsf{k}_u(X,x), B_1, \ldots, B_n}$$

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

- $K \Cup f = K \cup \{f\Downarrow\}$;
- $K \Cup f = K \cup \{\mathsf{i}_w(R\downarrow, R'\downarrow) \Leftarrow B_1, \ldots B_n\}$ provided that
  - $f\Downarrow = \big(\mathsf{k}_w(R,t) \Leftarrow B_1, \ldots B_n\big)$, and
  - f is a solved statement but not a VIP one, and
  - $\big(\mathsf{k}_w(R',t) \Leftarrow B_1, \ldots B_n\big) \in \mathbf{conseq}(K_{\mathsf{solved}})$.

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.

Definition 14: Let P be a ground process, and $\mathcal{N}^0_{\mathsf{pub}}$ be a set of names. We have that

$$\mathsf{seed}(P, \mathcal{N}^0_{\mathsf{pub}}) = f_0^+ \uplus f_1^+ \uplus f_2^+ \uplus f_3^+ \uplus f_4^+ \uplus S$$

$$\textsc{Resolution}\ \frac{\begin{array}{c} f = \Big(H \Leftarrow \mathsf{k}_{uv}(X,t), B_1, \ldots, B_n \parallel \mathcal{M}\Big) \in K \text{ such that } \mathsf{k}_{uv}(X,t) = \mathsf{sel}(f) \\ g = \Big(\mathsf{k}_w(R,t') \Leftarrow B_{n+1}, \ldots, B_m\Big) \in K_{\mathsf{solved}} \smallsetminus f_0^+ \end{array}}{\begin{array}{c} K = K \Cup \{h\sigma \mid \sigma \in \mathrm{csu}_{\mathsf{AC}}(\mathsf{k}_u(X,t), \mathsf{k}_w(R,t'))\} \\ \text{where } h = \Big(H \Leftarrow B_1, \ldots, B_n, B_{n+1}, \ldots, B_m \parallel \mathcal{M} \smallsetminus \{\mathsf{k}_{uv}(X,t)\}\Big) \end{array}}$$

$$\textsc{Resolution+}\ \frac{f = \Big(H \Leftarrow \mathsf{k}_u(X,t), B_1, \ldots, B_n \parallel \mathcal{M}\Big) \in K \text{ such that } \mathsf{k}_u(X,t) = \mathsf{sel}(f) \text{ and } \mathsf{k}_u(X,t) \notin \mathcal{M}}{\begin{array}{c} K = K \Cup \{h\sigma \mid \sigma \in \mathrm{csu}_{\mathsf{AC}}(\langle X,t\rangle, \langle X_1 \oplus X_2, x_1 \oplus x_2\rangle)\} \\ \text{where } h = \Big(H \Leftarrow B_1, \ldots, B_n, \mathsf{k}_u(X_1,x_1), \mathsf{k}_u(X_2,x_2) \ \parallel\ \mathcal{M}'\Big) \\ \text{and } \mathcal{M}' = \mathcal{M} \cup \{\mathsf{k}_u(X_i,x_i) \mid \mathrm{rigid}(t)\sigma \in \mathsf{factor}(x_i\sigma) \text{ and } i \in \{1,2\}\} \end{array}}$$

$$\textsc{Equation}\ \frac{f,g \in (K_{\mathsf{solved}} \smallsetminus f_0^+) \qquad f = \Big(\mathsf{k}_u(R,t) \Leftarrow B_1, \ldots, B_n\Big) \qquad g = \Big(\mathsf{k}_{u'v'}(R',t') \Leftarrow B_{n+1}, \ldots, B_m\Big)}{K = K \Cup \{h\sigma \mid \sigma \in \mathrm{csu}_{\mathsf{AC}}(\langle u,t\rangle, \langle u',t'\rangle)\} \text{ where } h = (\mathsf{i}_{u'v'}(R,R') \Leftarrow B_1, \ldots, B_m)}$$

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

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 $f_0^+ \cap \mathsf{seed}(P, \mathcal{N}^0_{\mathsf{pub}})$;
2) deduction statements in $(f_1^+ \cup f_2^+) \cap \mathsf{seed}(P, \mathcal{N}^0_{\mathsf{pub}})$ with their two hypotheses marked:

$$\mathsf{k}_w(X_1 \oplus X_2, t) \Leftarrow B_1, B_2 \parallel \{B_1, B_2\}.$$

The *initial knowledge base* associated to $\mathsf{seed}(P, \mathcal{N}^0_{\mathsf{pub}})$, denoted $K_{\mathsf{init}}(\mathsf{seed}(P, \mathcal{N}^0_{\mathsf{pub}}))$, is defined to be K_0 updated by the set S, i.e.,

$$K_{\mathsf{init}}(\mathsf{seed}(P, \mathcal{N}^0_{\mathsf{pub}})) = (((K_0 \Cup g_1) \Cup g_2) \ldots g_k)$$

where $g_1, \ldots, g_k$ is an enumeration of the statements in S. We sometimes write $K_{\mathsf{init}}(P)$ for $K_{\mathsf{init}}(\mathsf{seed}(P, \mathcal{N}^0_{\mathsf{pub}}))$.

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

$$\textsc{Refl}\ \frac{}{\mathsf{i}_w(R,R) \in \mathcal{H}_{\mathsf{e}}(K)} \qquad \textsc{Ext}\ \frac{\mathsf{i}_u(R,R') \in \mathcal{H}_{\mathsf{e}}(K)}{\mathsf{i}_{uv}(R,R') \in \mathcal{H}_{\mathsf{e}}(K)}$$

$$\textsc{Cong}\ \frac{\mathsf{i}_w(R_1,R_1'), \ldots, \mathsf{i}_w(R_n,R_n') \in \mathcal{H}_{\mathsf{e}}(K) \ \ \mathsf{f} \in \Sigma}{\mathsf{i}_w(\mathsf{f}(R_1, \ldots R_n), \mathsf{f}(R_1', \ldots R_n')) \in \mathcal{H}_{\mathsf{e}}(K)}$$

$$\textsc{Mod-I}\ \frac{\mathsf{i}_w(R_1,R_2) \in \mathcal{H}_{\mathsf{e}}(K) \ \ R_i{\downarrow} =_{\mathsf{AC}} R_i'{\downarrow} \ \ i \in \{1,2\}}{\mathsf{i}_w(R_1',R_2') \in \mathcal{H}_{\mathsf{e}}(K)}$$

$$\textsc{Mod-RI}\ \frac{\mathsf{ri}_w(R_1,R_2) \in \mathcal{H}_{\mathsf{e}}(K) \ \ R_i{\downarrow} =_{\mathsf{AC}} R_i'{\downarrow} \ \ i \in \{1,2\}}{\mathsf{ri}_w(R_1',R_2') \in \mathcal{H}_{\mathsf{e}}(K)}$$

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

Fig. 7: Rules of $\mathcal{H}_{\mathsf{e}}(K)$

C. Soundness and completeness

We shall show that the solved statements of any $K \in \mathsf{sat}(K_{\mathsf{init}}(P))$ 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 $\mathcal{H}_{\mathsf{e}}(K)$ 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 $K \in \mathsf{sat}(K_{\mathsf{init}}(P))$ for some ground process P. We have that $P \models f$ for any $f \in K \cup \mathcal{H}_{\mathsf{e}}(K)$ and, if $(P, \emptyset) \xrightarrow{L_1,\ldots,L_n} (Q, \varphi)$, then:

1) $\mathsf{r}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow} \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$;
2) if $\varphi \vdash_R t$ then $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, t{\downarrow}) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$;

3) if $\varphi \vdash_R t$ and $\varphi \vdash_{R'} t$, then $\mathsf{i}_{L_1\varphi\downarrow,\dots,L_n\varphi\downarrow}(R,R') \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$.

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_0^+ statements in derivation trees of $\mathcal{H}(K)$. 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, $\mathcal{N}^P_{\mathsf{pub}} \subseteq \mathcal{N}_{\mathsf{pub}}$ be the finite set of public names occurring in P. We have that:

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

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

- $\mathsf{k}_\epsilon(n,n) \Leftarrow$ for any $n \in \mathcal{N}_{\mathsf{pub}} \smallsetminus \mathcal{N}^P_{\mathsf{pub}}$;
- $\mathsf{i}_\epsilon(n,n) \Leftarrow$ for any $n \in \mathcal{N}_{\mathsf{pub}} \smallsetminus \mathcal{N}^P_{\mathsf{pub}}$;
- $\mathsf{ri}_u(n,n) \Leftarrow B_1,\dots,B_n$ for any $\mathsf{r}_u \Leftarrow B_1,\dots,B_n \in K$ in solved form, any $n \in \mathcal{N}_{\mathsf{pub}} \smallsetminus \mathcal{N}^P_{\mathsf{pub}}$.

Another issue is that, when computing the update operator, we need to check whether there exists R such that the statement $\mathsf{k}_w(R,t) \Leftarrow B_1,\dots B_n$ 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:

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

Among others, the set of seed statements will contain:

$$\mathsf{r}_w \Leftarrow \mathsf{k}_{w_1}(X, y \oplus z), \mathsf{k}_{w_2}(Y,y), \mathsf{k}_{w_3}(Z,z) \parallel \emptyset$$

where $w = \mathbf{in}(c, y \oplus z).\mathbf{in}(c,y).\mathbf{in}(c,z).\mathbf{test}$, 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:

$$\begin{aligned}\mathsf{r} \Leftarrow\ & \mathsf{k}(X_1, y_1 \oplus z_1),\ \mathsf{k}(X_2, y_2 \oplus z_2),\\ & \mathsf{k}(Y, y_1 \oplus y_2),\ \mathsf{k}(Z, z_1 \oplus z_2)\ \parallel\ \emptyset\end{aligned}$$

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

The test REACH-IDENTITY$(K^0_{\mathsf{solved}}, \mathcal{P})$ checks whether each reachable identity $\mathsf{ri}_{\ell_1,\dots,\ell_n}(R,R') \Leftarrow B_1,\dots,B_m$ in K^0_{solved} holds in $\mathcal{P}$. 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 $\ell_1\sigma,\dots,\ell_n\sigma$. The substitution σ replaces variables by fresh names in order to close the run. Next we check whether the corresponding trace $(M_1,\dots,M_n)$ is executable in $\mathcal{P}$ and whether the test $R\omega \stackrel{?}{=} R'\omega$ holds in the resulting frame φ, i.e., the frame reached by $\mathcal{P}$ after performing $M_1,\dots,M_n$. If all the tests succeed, P is trace included in $\mathcal{P}$. If one test fails, the algorithm returns this test as a witness of non equivalence.

$\text{Reach-Identity}(K^0_{\mathsf{solved}}, \mathcal{P})$
For all $\mathsf{ri}_{\ell_1,\ldots,\ell_n}(R, R') \Leftarrow \mathsf{k}_{w_1}(X_1, x_1), \ldots, \mathsf{k}_{w_m}(X_m, x_m) \in K^0_{\mathsf{solved}}$

let $c_1, \ldots, c_k$ be fresh public names such that $\sigma : vars(\ell_1, \ldots, \ell_n) \cup \{x_1, \ldots, x_m\} \to \{c_1, \ldots, c_k\}$ is a bijection

for all i such that $\ell_i = \mathbf{in}(d_i, t_i)$, let R_i be such that $\mathsf{k}_{\ell_1\sigma,\ldots,\ell_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(K^0_{\mathsf{solved}} \cup \{\mathsf{k}_\epsilon(c_i, c_i) \Leftarrow \ \mid 1 \leq i \leq k\})$

let $M_i = \begin{cases} \ell_i & \text{if } \ell_i \in \{\mathbf{test}, \mathbf{out}(c) \mid c \in \mathcal{C}h\} \\ \mathbf{in}(d_i, R_i) & \text{if } \ell_i = \mathbf{in}(d_i, t_i) \end{cases}$

check that $(\mathcal{P}, \emptyset) \xRightarrow{M_1,\ldots,M_n} (P, \varphi)$ and $R\omega\varphi\!\downarrow =_{\mathsf{AC}} R'\omega\varphi\!\downarrow$ where $\omega = \{X_i \mapsto x_i\sigma\}$.

Fig. 8: Test for checking $P \sqsubseteq \mathcal{P}$

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

- if $P \sqsubseteq \mathcal{P}$ then $\text{Reach-Identity}(K^0_{\mathsf{solved}}, \mathcal{P})$ holds;
- if $\mathcal{P}$ is determinate and $\text{Reach-Identity}(K^0_{\mathsf{solved}}, \mathcal{P})$ holds then $P \sqsubseteq \mathcal{P}$.

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 $\mathsf{k}_w(R, t) \in \mathcal{H}(K)$ given w, t, K. This can be achieved using a simple recursive backward search, as in [18].

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 ($\sim 3\%$ 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 ($P \mathbin{++} Q$), 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 [16]. 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 $\mathtt{tag}_1$ or $\mathtt{tag}_2$ assuming that the attacker has previously observed a session between the reader and $\mathtt{tag}_1$. We use the process P_{tag} (Example 4) to model the tag. The processes P_{tag_1} and P_{tag_2} are slightly different versions of P_{tag}. More precisely, they are obtained from P_{tag} by replacing id and k by id_1 and k_1 (resp. id_2 and k_2). Then, we consider a process P_{init} to model the outputs observed by an attacker during an honest interaction between $\mathtt{tag}_1$ and the reader:

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

Using non-deterministic choice we model that the reader may engage a session with tag_1 (using id_1 and key k_1) or tag_2 (using id_2 and key k_2):

$$\begin{aligned} P_{\mathsf{reader}} = {} & \mathbf{out}(c, r_1).\ \mathbf{in}(c, y). \\ & ([(\mathsf{proj}_1(y) \oplus id) \oplus \mathsf{proj}_2(y) = \mathsf{h}(\langle r_1, k_1\rangle)].\ \mathbf{0}) \\ & \mathbin{++} ([(\mathsf{proj}_1(y) \oplus id) \oplus \mathsf{proj}_2(y) = \mathsf{h}(\langle r_1, k_2\rangle)].\ \mathbf{0}) \end{aligned}$$

Unlinkability can finally be expressed as the following process equivalence:

$$P_{\mathsf{init}} :: (P_{\mathsf{tag}_1} \parallel P_{\mathsf{reader}}) \approx P_{\mathsf{init}} :: (P_{\mathsf{tag}_2} \parallel P_{\mathsf{reader}})$$

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 $\mathsf{h}(\langle r_1, k\rangle) \oplus r_2$ by $\mathsf{h}(\langle r_1, k \oplus r_2\rangle)$. 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.*:

$$P_{\mathsf{init}} :: (P_{\mathsf{tag}_1} \parallel P_{\mathsf{reader}}) \sqsubseteq P_{\mathsf{init}} :: (P_{\mathsf{tag}_2} \parallel P_{\mathsf{reader}})$$

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:

$$\begin{aligned} &P_{\mathsf{init}} :: \big((P_{\mathsf{tag}_1} \parallel P_{\mathsf{reader}}) \gg (P_{\mathsf{tag}_1} \parallel P_{\mathsf{reader}})\big) \\ \approx\ &P_{\mathsf{init}} :: \big((P_{\mathsf{tag}_2} \parallel P_{\mathsf{reader}}) \gg (P_{\mathsf{tag}_2} \parallel P_{\mathsf{reader}})\big) \end{aligned}$$

RFID Protocol	# sessions	# traces	time	result
KCL (Ex. 4)	1	1	1s	✕
KCL	1	20	3s	✕
KCL fixed	1	20	<1s	✓
KCL fixed	2	980	1m47	✓
LD	1	20	<1s	✕
OTYT	1	20	<1s	✕
YPL	1	20	<1s	✕
LAK	1	20	<1s	✓
LAK	2	980	30m	✓
MW	1	40	7s	✓
MW	2	4280	10h	✓
AT	1	20	1s	✓
AT	2	1040	33s	✓

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.

$$\begin{array}{ll} A \to B: & ra, \{pub\}_{pw} \\ B \to A: & \{B, A, nb_1, nb_2, cb, \{ra\}_{pw}\}_{pub} \\ A \to B: & \{nb_1, k \oplus nb_2\}_{pw} \\ B \to A: & \{f_1(ra), rb\}_k \\ A \to B: & \{f_2(rb)\}_k \end{array}$$

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:

$$(P_A \parallel P_B) \gg \mathbf{out}(c, pw) \approx (P_A \parallel P_B) \gg \mathbf{out}(c, pw')$$

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

In total, we modelled 5 password-based protocols: the Toy, Nonce, 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.*:

$$(P_A \parallel P_B) \gg \mathbf{out}(c, pw) \sqsubseteq (P_A \parallel P_B) \gg \mathbf{out}(c, pw')$$

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 (Nonce 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

	passive case		active case			
	time	result	#sessions	# tr.	time	result
Toy	<1s	✓	1	5	<1s	×
Nonce	<1s	✓	1	90	4s	✓
Sec PubKey	<1s	✓	1	90	4s	✓
Direct Auth	<1s	✓	1	29	<1s	✓
			2	18k	9m	✓
Gong	26s	×	not relevant			

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

Fig. 10: Summary for password-based protocols

we are able to find (known) attacks. The attack on NSL-xor is a variant of Lowe's classical man in the middle attack which is prevented on NSL, but possible on NSL-xor. To find the attack we analyse a scenario where A starts a session with the attacker and B a session with A.

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.

$$\mathbf{in}(c, x_1).\mathbf{in}(c, x_2) :: \mathsf{NSL}\{^{x_1}/_{n_a}\} \\ \approx \\ \mathbf{in}(c, x_1).\mathbf{in}(c, x_2) :: \mathsf{NSL}\{^{x_2}/_{n_a}\}$$

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.

Protocol	# sessions	# traces	time	result
LAK auth	1	1	<1s	×
KCL fixed auth	1	1	<1s	✓
KCL fixed auth	2	36	3s	✓
NSL xor				
-auth	1	3	<1s	×
-strong secrecy n_a	1	6	9s	×
-strong secrecy n_b	1	6	<1s	×

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

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 [38].

References

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

[2] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. Comput.*, 148(1):1–70, 1999.

[3] AKISS. https://github.com/akiss/akiss.

[4] M. Arapinis, T. Chothia, E. Ritter, and M. D. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Comp. Soc., 2010.

[5] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. T. Abad. Formal analysis of SAML 2.0 web browser single sign-on: Breaking the SAML-based single sign-on for Google apps. In *6th Workshop on Formal Methods in Security Engineering (FMSE'08)*, pages 1–10. ACM, 2008.

[6] A. Armando et al. The AVISPA tool for the automated validation of internet security protocols and applications. In *17th International Conference on Computer Aided Verification (CAV'05)*, LNCS, pages 281–285. Springer, 2005.

[7] M. Asadpour and M. Torabi Dashti. A privacy-friendly rfid protocol using reusable anonymous tickets. In *10th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 206–213. IEEE Comp. Soc., 2011.

[8] D. Baelde, S. Delaune, and L. Hirschi. A reduced semantics for deciding trace equivalence using constraint systems. In *3rd International Conference on Principles of Security and Trust (POST'14)*, volume 8414 of *LNCS*, pages 1–21. Springer, 2014.

[9] D. Baelde, S. Delaune, and L. Hirschi. Partial order reduction for security protocols. In *26th International Conference on Concurrency Theory (CONCUR'15)*, volume 42 of *LIPICS*, pages 497–510. Leibniz-Zentrum für Informatik, 2015.

[10] D. A. Basin, C. Cremers, and S. Meier. Provably repairing the ISO/IEC 9798 standard for entity authentication. *Journal of Computer Security*, 21(6):817–846, 2013.

[11] D. A. Basin, J. Dreier, and R. Sasse. Automated symbolic proofs of observational equivalence. In *22nd Conference on Computer and Communications Security (CCS'15)*, pages 1144–1155. ACM, 2015.

[12] M. Baudet. Deciding security of protocols against off-line guessing attacks. In *12th Conference on Computer and Communications Security (CCS'05)*, pages 16–25. ACM, 2005.

[13] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Pironti, and P.-Y. Strub. Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS. In *35th Symposium on Security and Privacy (S&P'14)*, pages 98–113. IEEE Comp. Soc., 2014.

[14] B. Blanchet. Automatic proof of strong secrecy for security protocols. In *Symposium on Security and Privacy (S&P'04)*, pages 86–100. IEEE Comp. Soc., 2004.

[15] B. Blanchet, M. Abadi, and C. Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *Symposium on Logic in Computer Science (LICS'05)*, pages 331–340. IEEE Comp. Soc., 2005.

[16] B. Blanchet, B. Smyth, and V. Cheval. *Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*, 2016.

[17] M. Bruso, K. Chatzikokolakis, and J. den Hartog. Analysing unlinkability and anonymity using the applied pi calculus. In *23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Comp. Soc., 2010.

[18] R. Chadha, V. Cheval, Ş. Ciobâcă, and S. Kremer. Automated verification of equivalence properties of cryptographic protocol. *ACM Transactions on Computational Logic*, 23(4), 2016.

[19] V. Cheval, H. Comon-Lundh, and S. Delaune. Trace equivalence decision: Negative tests and non-determinism. In *18th Conference on Computer and Communications Security (CCS'11)*, pages 321–330. ACM, 2011.

[20] V. Cheval, V. Cortier, and S. Delaune. Deciding equivalence-based properties using constraint solving. *Theoretical Computer Science*, 492:1–39, 2013.

[21] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In *18th Symposium on Logic in Computer Science (LICS'03)*, pages 261–270. IEEE Comp. Soc., 2003.

[22] H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *LNCS*, pages 294–307. Springer, 2005.

[23] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *18th Symposium on Logic in Computer Science (LICS'03)*, pages 271–280. IEEE Comp. Soc., 2003.

[24] R. Corin, J. Doumen, and S. Etalle. Analysing password protocol security against off-line dictionary attacks. *Electr. Notes Theor. Comput. Sci.*, 121:47–63, 2005.

[25] V. Cortier, S. Delaune, and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.

[26] S. Delaune, S. Kremer, and D. Pasaila. Security protocols, constraint systems, and group theories. In *6th International Joint Conference on Automated Reasoning (IJCAR'12)*, volume 7364 of *LNAI*, pages 164–178. Springer, 2012.

[27] S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[28] S. Escobar, C. Meadows, and J. Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In *Foundations of Security Analysis and Design V*, volume 5705 of *LNCS*, pages 1–50. Springer, 2009.

[29] S. Escobar, R. Sasse, and J. Meseguer. Folding variant narrowing and optimal variant termination. *J. Log. Algebr. Program.*, 81(7-8):898–928, 2012.

[30] L. Gong. Using one-way functions for authentication. *Computer Communication Review*, 19:8–11, 1989.

[31] L. Gong, T. M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.

[32] A. González-Burgueño, S. Santiago, S. Escobar, C. Meadows, and J. Meseguer. Analysis of the ibm cca security api protocols in maude-npa. In *1st International Conference on Security Standardisation Research (SSR'14)*, volume 8893 of *LNCS*, pages 111–130. Springer, 2014.

[33] R. Küsters and T. Truderung. Reducing protocol analysis with XOR to the xor-free case in the Horn theory based approach. *J. Autom. Reasoning*, 46(3-4):325–352, 2011.

[34] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, 1996.

[35] The Maude system. http://maude.cs.illinois.edu/.

[36] D. Molnar and D. Wagner. Privacy and security in library rfid: Issues, practices, and architectures. In *11th ACM Conference on Computer and Communications Security (CCS'04)*, pages 210–219. ACM, 2004.

[37] S. Santiago, S. Escobar, C. Meadows, and J. Meseguer. A formal definition of protocol indistinguishability and its verification using Maude-NPA. In *10th International Workshop on Security and Trust Management (STM'14)*, volume 8743, pages 162–177. Springer, 2014.

[38] B. Schmidt, S. Meier, C. Cremers, and D. Basin. The TAMARIN prover for the symbolic analysis of security protocols. In *25th International Conference on Computer Aided Verification (CAV'13)*, volume 8044 of *LNCS*, pages 696–701. Springer, 2013.

[39] A. Tiu and J. Dawson. Automating open bisimulation checking for the spi-calculus. In *23rd Computer Security Foundations Symposium (CSF'10)*, pages 307–321. IEEE Comp. Soc., 2010.

[40] D. Unruh. The impossibility of computationally sound XOR. *IACR Cryptology ePrint Archive*, 2010:389, 2010.

[41] T. van Deursen and S. Radomirovic. Attacks on RFID protocols. *IACR Cryptology ePrint Archive*, 2008:310, 2008.

APPENDIX
PROOFS REGARDING SEED STATEMENTS

As the construction of the seed is similar to the one in [18], the proof closely follows the original proof, generalising it to reason modulo AC.

A. Soundness

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 $f \in \mathsf{seed}(P)$.

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 $f \in \mathsf{seed}(P)$ 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 $0 \leq m \leq n$, σ and τ be substitutions such that $\sigma \in \mathrm{csu}_{\mathcal{R},\mathsf{AC}}(\{s_k = t_k\}_{k \in T(m)})$ and $\tau \in \mathrm{variants}_{\mathcal{R},\mathsf{AC}}(\ell_1\sigma, \ldots, \ell_m\sigma)$. Let f be the following statement:

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

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

$$P \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_p\sigma\tau\downarrow})\omega \text{ for all } 0 \leq p \leq m.$$

We proceed by induction on p. When $p = 0$, we have that $(\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_p\sigma\tau\downarrow})\omega = \mathsf{r}$, and $P \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_p\sigma\tau\downarrow})\omega$ trivially. Now, assume that $p > 0$, and $P \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{p-1}\sigma\tau\downarrow})\omega$. We have to show that $P \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_p\sigma\tau\downarrow})\omega$ by case analysis on a_p. Before, we do the case analysis, let us first fix some notations.

Let $(P_1, \varphi_1) = (P, \emptyset)$. As $P \models (\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{p-1}\sigma\tau\downarrow})\omega$, we have that there exist $L_1, \ldots, L_{p-1}$ such that for all $1 \leq i < p$

$$(P_i, \varphi_i) \xrightarrow{L_i} (P_{i+1}, \varphi_{i+1}) \text{ and } L_i\varphi_i\downarrow =_{\mathsf{AC}} \ell_i\sigma\tau\downarrow\omega\downarrow$$

where

- $P_i = (\mathsf{a}_i.\ldots.\mathsf{a}_n)\{x_j \mapsto x_j\sigma\tau\downarrow\omega\downarrow\}_{j \in R(i-1)}$; and
- φ_i extends φ_{i-1}.

We can now do the case analysis.

1) Case $\mathsf{a}_p = \mathbf{out}(c_p, t_p)$. We have that $\ell_p = \mathbf{out}(c_p)$. Let $L_p = \mathbf{out}(c_p)$, $P_{p+1} = (\mathsf{a}_{p+1}.\ldots.\mathsf{a}_n)\{x_j \mapsto x_j\sigma\tau\downarrow\omega\downarrow\}_{j \in R(p)}$ and $\varphi_{p+1} = \varphi_p \cup \{\mathsf{w}_{|dom(\varphi_p)|+1} \mapsto t_p\sigma\tau\downarrow\omega\downarrow\}$. We have:

$$(P_p, \varphi_p) \xrightarrow{L_p} (P_{p+1}, \varphi_{p+1})$$

which is what we wanted to prove.

2) Case $\mathsf{a}_p = [s_p = t_p]$. We have that $\ell_p = \mathbf{test}$. Let $P_{p+1} = (\mathsf{a}_{p+1}.\ldots.\mathsf{a}_n)\{x_j \mapsto x_j\sigma\tau\downarrow\omega\downarrow\}_{j \in R(p)}$ and let $\varphi_{p+1} = \varphi_p$. As $\sigma \in \mathrm{csu}_{\mathcal{R},\mathsf{AC}}(\{s_k = t_k\}_{k \in T(m)})$, we have

$s_p\sigma\downarrow =_{\mathsf{AC}} t_p\sigma\downarrow$ and therefore $s_p\sigma\tau\downarrow\omega\downarrow =_{\mathsf{AC}} t_p\sigma\tau\downarrow\omega\downarrow$. Hence, we have that

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

which is what we wanted to prove.

3) Case $\mathsf{a}_p = \mathbf{in}(c_p, x_p)$. We have $p \in R(p)$ and $\ell_p = \mathbf{in}(c_p, x_p)$.
Let $P_{p+1} = (\mathsf{a}_{p+1}.\ldots.\mathsf{a}_n)\{x_j \mapsto x_j\sigma\tau\downarrow\omega\downarrow\}_{j\in R(p)}$, and $\varphi_{p+1} = \varphi_p$. As $p \in R(p)$, we have that

$$P \models (\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{p-1}\sigma\tau\downarrow}(X_p, x_p\sigma\tau\downarrow))\omega$$

(this is an antecendent of f). Therefore $\varphi_p \vdash_{X_p\omega} x_p\sigma\tau\downarrow\omega$ and, by letting $L_p = \mathbf{in}(c_p, X_p\omega)$, we obtain by the definition of $\rightarrow$ that

$$(P_p, \varphi_p) \xrightarrow{L_p} (P_{p+1}, \varphi_{p+1})$$

which is what we wanted to prove.

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

Case 2. Let $m \in S(n)$, $\sigma \in \mathrm{csu}_{\mathcal{R},\mathsf{AC}}(\{s_k = t_k\}_{k\in T(m)})$, and $\tau \in \mathrm{variants}_{\mathcal{R},\mathsf{AC}}(\ell_1\sigma, \ldots, \ell_m\sigma, t_m\sigma)$. Let f be the following statement:

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

We have to show that $P \models f$. Let ω be an arbitrary substitution grounding for f. We assume furthermore that $P \models (\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow))\omega$ for any $j \in R(m)$. We have to show that $P \models (\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow}(\mathsf{w}_{|S(m)|}, t_m\sigma\tau\downarrow))\omega$.

Let $P_i = (\mathsf{a}_i.\ldots.\mathsf{a}_n)\{x_j \mapsto x_j\sigma\tau\downarrow\omega\downarrow\}_{j\in R(i-1)}$ and let

$$\varphi_i = \cup_{1\le j\le |S(i-1)|}\{\mathsf{w}_j \mapsto t_{o(j)}\sigma\tau\omega\downarrow\}$$

where $o(j) = \min\{x \mid |S(x)| = j\}$, *i.e.* $o(j)$ denotes the index of the j^{th} send action.

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 $L_i\varphi_i\downarrow =_{\mathsf{AC}} \ell_i\sigma\tau\downarrow\omega\downarrow$ for all $1 \le i \le m$, we have that

$$\varphi_{m+1}(\mathsf{w}_{|S(m)|}) = t_{o(|S(m)|)}\sigma\tau\omega\downarrow = t_m\sigma\tau\omega\downarrow$$

and we have that $\varphi_{m+1} \vdash_{\mathsf{w}_{|S(m)|}} t_m\sigma\tau\omega\downarrow$ and therefore $\varphi_{m+1} \vdash_{\mathsf{w}_{|S(m)|}} t_m\sigma\tau\downarrow\omega$ which implies that

$$P \models (\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow}(\mathsf{w}_{|S(m)|}, t_m\sigma\tau\downarrow))\omega.$$

2) otherwise, we trivially have that:

$$P \models \mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow}(\mathsf{w}_{|S(m)|}, (t_m\sigma\tau)\downarrow)\omega.$$

We have shown that $T \models f$.

Case 3. Let c be a public name. We have that $\emptyset \vdash_c c$, and therefore $P \models \left(\mathsf{k}_\epsilon(c, c) \Leftarrow\right)$ trivially holds.

Case 4. Let m be such that $0 \le m \le n$. Let f be a function symbol of arity k and let $\tau \in \mathrm{variants}_{\mathcal{R},\mathsf{AC}}(\mathsf{f}(y_1, \ldots, y_k))$. Let f be the following statement:

$$\mathsf{k}_{\ell_1,\ldots,\ell_m}(\mathsf{f}(Y_1, \ldots, Y_k), \mathsf{f}(y_1, \ldots, y_k)\tau\downarrow) \Leftarrow \mathsf{k}_{\ell_1,\ldots,\ell_m}(Y_1, y_1\tau\downarrow), \ldots, \mathsf{k}_{\ell_1,\ldots,\ell_m}(Y_k, y_k\tau\downarrow)$$

Let ω be an arbitrary substitution grounding for f. We assume that $P \models \mathsf{k}_{\ell_1,\ldots,\ell_m}(Y_j, y_j\tau\downarrow)\omega$ for all $1 \le j \le k$. 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 $L_i\varphi_i\downarrow =_{\mathsf{AC}} \ell_i\sigma\tau\downarrow\omega\downarrow$ for all $1 \le i \le m$, we have that $\varphi_{m+1} \vdash_{Y_j\omega} y_j\tau\downarrow\omega$ for all $1 \le j \le k$. This implies that $\varphi_{m+1} \vdash_{\mathsf{f}(Y_1\omega,\ldots,Y_k\omega)} \mathsf{f}(y_1\tau\downarrow\omega, \ldots, y_k\tau\downarrow\omega)$, and therefore $P \models (\mathsf{k}_{\ell_1,\ldots,\ell_m}(\mathsf{f}(Y_1, \ldots, Y_k), \mathsf{f}(y_1, \ldots, y_k)\tau\downarrow))\omega$.

2) otherwise, we trivially have that:

$$P \models (\mathsf{k}_{\ell_1,\ldots,\ell_m}(\mathsf{f}(Y_1, \ldots, Y_k), \mathsf{f}(y_1, \ldots, y_k)\tau\downarrow))\omega.$$

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

Hence, for any statement $f \in \mathsf{seed}(P)$, we have shown that $P \models f$. ☐

Lemma 3: Let P 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 prooftree witnessing the fact that $f \in \mathcal{H}(K)$. Let π be such a proof tree.

Base case. The prooftree π is reduced to a leaf meaning that it has been obtained by applying the rule CONSEQ on a statement $f' = (H \Leftarrow)$ (*i.e.* with $n = 0$). We have that $f = f'\sigma$ where σ 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 $f' = (H \Leftarrow B_1 \ldots B_n) \in K$, σ is a substitution grounding for f' such that $f = H\sigma$, $B_i\sigma \in \mathcal{H}(K)$ for $1 \le i \le n$, and $\mathsf{skel}(f'\sigma)$ 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 $1 \le i \le n$. Hence, we conclude that $P \models H\sigma$.
- EXTEND: We have $f = \mathsf{k}_{uv}(R, t)$ with $\mathsf{k}_u(R, t) \in \mathcal{H}(K)$. By induction hypothesis, we have that $P \models \mathsf{k}_u(R, t)$, and it follows from the semantics of k that $P \models \mathsf{k}_{uv}(R, t)$.

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

(A) $\mathsf{r}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow} \in \mathcal{H}(\mathsf{seed}(P))$;
(B) if $\varphi \vdash_R t$ then $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, t\downarrow) \in \mathcal{H}(\mathsf{seed}(P))$.

Proof. We prove the two statements by induction on n. We assume that the two statements hold for any index less than n and we prove them for n. By hypothesis, we have that $(P, \emptyset) \xrightarrow{L_1,\ldots,L_n} (Q, \varphi)$, and therefore there exists ω such that:

- $(L_1\varphi\downarrow, \ldots, L_n\varphi\downarrow) =_{\mathsf{AC}} (\ell_1, \ldots, \ell_n)\omega\downarrow$; and
- $s_k\omega\downarrow =_{\mathsf{AC}} t_k\omega\downarrow$ for all $k \in T(n)$.

First, we prove statement (A). We have $s_k\omega\downarrow =_{\mathsf{AC}} t_k\omega\downarrow$ for all $k \in T(n)$. Therefore, there exists $\sigma \in \mathrm{csu}_{\mathcal{R},\mathsf{AC}}(\{s_k \stackrel{?}{=} t_k\}_{k\in T(n)})$ and a substitution π such that:

1) $s_k\sigma\downarrow =_{\mathsf{AC}} t_k\sigma\downarrow$ for all $k \in T(n)$; and
2) $x\omega\downarrow =_{\mathsf{AC}} x\sigma\pi\downarrow$ for any $x \in vars(\{s_k, t_k\}_{k\in T(n)})$.

It follows that $(\ell_1, \ldots, \ell_n)\omega\downarrow =_{\mathsf{AC}} (\ell_1, \ldots, \ell_n)\sigma\pi\downarrow$. By definition of $\mathrm{variants}_{\mathcal{R},\mathsf{AC}}((\ell_1, \ldots, \ell_n)\sigma)$, we know that there exists $\tau \in \mathrm{variants}_{\mathcal{R},\mathsf{AC}}((\ell_1, \ldots, \ell_n)\sigma)$ such that

$$(\ell_1, \ldots, \ell_n)\sigma\pi\downarrow =_{\mathsf{AC}} (\ell_1, \ldots, \ell_n)\sigma\tau\downarrow\tau'$$

for some substitution τ'. Let f be the following statement.

$$\mathsf{r}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_n\sigma\tau\downarrow} \Leftarrow \mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow)_{j\in R(n)}$$

By the definition of $\mathsf{seed}(P)$, we have that $f \in \mathsf{seed}(P)$. Let τ'' be the substitution that extends τ' by $\{X_j \mapsto R_j\}_{j\in R(n)}$ where R_j are recipes for $x_j\omega$. Applying our induction hypothesis, we have that $\mathsf{k}_{\ell_1\sigma\tau\downarrow\tau'',\ldots,\ell_{j-1}\sigma\tau\downarrow\tau''}(X_j\tau'', x_j\sigma\tau\downarrow\tau'')$ is in $\mathcal{H}(\mathsf{seed}(P))$ for $j \in R(n)$. Therefore

$$\mathsf{r}_{\ell_1\sigma\tau\downarrow\tau'',\ldots,\ell_n\sigma\tau\downarrow\tau''} \in \mathcal{H}(\mathsf{seed}(P)).$$

We have that:

$$\begin{aligned}(L_1\varphi\downarrow, \ldots, L_n\varphi\downarrow) &=_{\mathsf{AC}} (\ell_1, \ldots, \ell_n)\omega\downarrow\\ &=_{\mathsf{AC}} (\ell_1, \ldots, \ell_n)\sigma\pi\downarrow\\ &=_{\mathsf{AC}} (\ell_1, \ldots, \ell_n)\sigma\tau\downarrow\tau'\\ &=_{\mathsf{AC}} \ell_1\sigma\tau\downarrow\tau'', \ldots, \ell_n\sigma\tau\downarrow\tau''\end{aligned}$$

Therefore, we conclude that $\mathsf{r}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow} \in \mathcal{H}(\mathsf{seed}(P))$.

We now prove statement (B). By structural induction on the recipe R, we show that:

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

1) If $R = c$ is a public name, and as$f = \left(\mathsf{k}_\epsilon(c, c) \Leftarrow\right)$ is in the set of seed statements by definition, we have that $\mathsf{k}_\epsilon(R, R\varphi\downarrow) = \mathsf{k}_\epsilon(c, c) \in \mathcal{H}(\mathsf{seed}(T))$, and therefore $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, R\varphi\downarrow) \in \mathcal{H}(\mathsf{seed}(T))$ by the EXTEND rule.
2) If $R = \mathsf{w}_j$, let m be the smallest index such that $|S(m)| = j$ (*i.e.* m is the index of the action a_m that outputs the content of w_j) and let t_m be the term such that $\mathsf{a}_m = \mathbf{out}(c, t_m)$ for some channel c.
   Let $\sigma \in \mathrm{csu}_{\mathcal{R},\mathsf{AC}}(\{s_k \stackrel{?}{=} t_k\}_{k\in T(m)})$ be such that
   $$(\ell_1, \ldots, \ell_n)\omega\downarrow =_{\mathsf{AC}} (\ell_1, \ldots, \ell_n)\sigma\pi\downarrow$$
   for some π. Let $\tau \in \mathrm{variants}_{\mathcal{R},\mathsf{AC}}((\ell_1, \ldots, \ell_m, t_m)\sigma)$ and τ' be a substitution such that
   $$(\ell_1, \ldots, \ell_m, t_m)\omega\downarrow =_{\mathsf{AC}} (\ell_1, \ldots, \ell_m, t_m)\sigma\tau\downarrow\tau'.$$
   Let h be the following statement:
   $$\begin{aligned}&\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_m\sigma\tau\downarrow}(\mathsf{w}_j, t_m\sigma\tau\downarrow)\\ &\qquad\Leftarrow \{\mathsf{k}_{\ell_1\sigma\tau\downarrow,\ldots,\ell_{k-1}\sigma\tau\downarrow}(X_k, x_k\sigma\tau\downarrow)\}_{k\in R(m)}\end{aligned}$$
   By definition of $\mathsf{seed}(P)$, we have that $h \in \mathsf{seed}(P)$.
   For $k \in R(m)$, let R_k be recipes of $x_k\sigma\tau\downarrow\tau' =_{\mathsf{AC}} x_k\omega\downarrow$ in the smallest possible prefix of φ. Let
   $$\tau'' = \tau' \cup \{X_k \mapsto R_k\}_{k\in R(m)}.$$
   We have that the antecedents of $h\tau''$ are in $\mathcal{H}(\mathsf{seed}(T))$ by the induction hypothesis. Therefore
   $$\begin{aligned}&\mathsf{k}_{\ell_1\sigma\tau\downarrow\tau'',\ldots,\ell_m\sigma\tau\downarrow\tau''}(\mathsf{w}_j, t_m\sigma\tau\downarrow\tau'')\\ =_{\mathsf{AC}}\ &\mathsf{k}_{\ell_1\sigma\tau\downarrow\tau',\ldots,\ell_m\sigma\tau\downarrow\tau'}(\mathsf{w}_j, t_m\sigma\tau\downarrow\tau')\\ =_{\mathsf{AC}}\ &\mathsf{k}_{\ell_1\omega\downarrow,\ldots,\ell_m\omega\downarrow}(\mathsf{w}_j, t_m\omega\downarrow) \in \mathcal{H}(\mathsf{seed}(P)).\end{aligned}$$
   But $(\ell_1, \ldots, \ell_m)\omega\downarrow$ is a prefix of $(\ell_1, \ldots, \ell_n)\omega\downarrow$ and by the EXTEND rule, $\mathsf{k}_{\ell_1\omega\downarrow,\ldots,\ell_n\omega\downarrow}(\mathsf{w}_j, t_m\omega\downarrow) \in \mathcal{H}(\mathsf{seed}(T))$, *i.e.* $\mathsf{k}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, R\varphi\downarrow) \in \mathcal{H}(\mathsf{seed}(T))$, which is what we had to prove.
3) If $R = \mathsf{f}(R_1, \ldots, R_k)$, let $\tau \in \mathrm{variants}_{\mathcal{R},\mathsf{AC}}(\mathsf{f}(y_1, \ldots, y_k))$ and τ' be such that $R\varphi\downarrow =_{\mathsf{AC}} (\mathsf{f}(y_1, \ldots, y_k)\tau)\downarrow\tau'$ and $R_i\varphi\downarrow =_{\mathsf{AC}} y_i\tau\downarrow\tau'$ for $1 \le i \le k$. Let g be the following statement:
   $$\begin{aligned}&\mathsf{k}_{\ell_1,\ldots,\ell_n}(\mathsf{f}(Y_1, \ldots, Y_k), \mathsf{f}(y_1, \ldots, y_k)\tau\downarrow)\\ &\qquad\Leftarrow \{\mathsf{k}_{\ell_1,\ldots,\ell_n}(Y_j, y_j\tau\downarrow)\}_{j\in\{1,\ldots,k\}}\end{aligned}$$
   By the definition of $\mathsf{seed}(P)$, we have that $g \in \mathsf{seed}(P)$. 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 $\mathcal{H}(\mathsf{seed}(T))$ by the induction hypothesis. Therefore, the head of $g\tau''$ is also in $\mathcal{H}(\mathsf{seed}(T))$.

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 $h = \left(\mathsf{k}(R, x) \Leftarrow B_1, \ldots, B_n\right)$. 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 $\varphi \vdash_R t$ implies $\varphi \vdash_{R'} t$ when $R' = R\downarrow$.
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:
   - $f = H \Leftarrow \mathsf{k}_{uv}(X, t), \mathsf{k}_u(Y, t), B_1, \ldots, B_n \parallel \mathcal{M}$;
   - $f' = H \Leftarrow \mathsf{k}_u(Y, t), B_1, \ldots, B_n \parallel (\mathcal{M} \smallsetminus \mathsf{k}_{uv}(X, t))$.

   Moreover, we know that $X \notin \mathit{vars}(H)$. Let τ be an arbitrary substitution such that $T \models \mathsf{k}_u(Y, t)\tau$, $P \models B_1\tau, \ldots, P \models B_n\tau$. We consider the substitution τ' to be identical to τ except that $\tau'(X) = \tau(Y)$. We have that $P \models \mathsf{k}_{uv}(X, t)\tau$ 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:
   - $f = \mathsf{k}_{uv}(R, t) \Leftarrow \mathsf{k}_u(X, x), B_1, \ldots, B_n$;
   - $f' = \mathsf{k}_{uv}(R \oplus X, t \oplus x\downarrow) \Leftarrow \mathsf{k}_u(X, x), B_1, \ldots, B_n$.

   Moreover, we know that $x \in \mathsf{factor}(t)$. Let τ be an arbitrary substitution such that
   $$P \models \mathsf{k}_u(X, x)\tau,\ P \models B_1\tau,\ \ldots,\text{ and } P \models B_n\tau.$$
   Therefore, we have $P \models \mathsf{k}_{uv}(R, t)\tau$, and thus we have $P \models \mathsf{k}_{uv}(R \oplus X, t \oplus x)\tau$, and $P \models \mathsf{k}_{uv}(R \oplus X, t \oplus x\downarrow)\tau$. 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 $f \in K_{\mathsf{solved}}$ then we have that $P \models f$ for all $f \in \mathbf{conseq}(K_{\mathsf{solved}})$.

Proof. We show that both inference rules are sound.
Case: AXIOM. Immediate from the semantics of k.

Case: RES. We consider $f = \mathsf{k}_u(R, t) \Leftarrow B_1, \ldots, B_n$ and $g_i = B_i\sigma \Leftarrow C_1, \ldots, C_m$ 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
$$P \models \Big(\mathsf{k}_u(R, t)\sigma \Leftarrow C_1, \ldots, C_m\Big)$$
Let τ 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 $P \models B_i\sigma\tau$ with $1 \leq i \leq n$. Since $P \models f$, we have that $P \models \mathsf{k}_u(R, t)\sigma\tau$ as well. This allows us to conclude that $P \models \mathsf{k}_{uv}(R, t)\sigma\tau$. □

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 σ 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:
$$\begin{aligned} f &= H \Leftarrow \mathsf{k}_{uv}(X, t), B_1, \ldots, B_n \parallel \mathcal{M} \\ g &= \mathsf{k}_w(R, t') \Leftarrow B_{n+1}, \ldots, B_m \\ h &= H \Leftarrow B_1, \ldots, B_m \parallel \mathcal{M} \smallsetminus \{\mathsf{k}_{uv}(X, t)\} \end{aligned}$$
Moreover, we have that $\mathsf{k}_{uv}(X, t) \in \mathsf{sel}(f)$, g is a solved statement but not in f_0^+, and $\sigma \in \mathrm{csu}_{\mathsf{AC}}(\mathsf{k}_u(X, t), \mathsf{k}_w(R, t'))$.

Let τ be a substitution grounding for $h\sigma$ and assume that $P \models B_1\sigma\tau, \ldots, B_m\sigma\tau$. We will show that $P \models H\sigma\tau$.

As $P \models B_{n+1}\sigma\tau, \ldots, B_m\sigma\tau$ and because $P \models g$, we have that $P \models \mathsf{k}_w(R, t')\sigma\tau$. But $\mathsf{k}_w(R, t')\sigma\tau = \mathsf{k}_u(X, t)\sigma\tau$, and therefore we have that $P \models \mathsf{k}_{uv}(X, t)\sigma\tau$ 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 τ 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 σ 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 σ 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:
$$\begin{aligned} f &= \mathsf{k}_u(R, t) \Leftarrow B_1, \ldots, B_n \\ g &= \mathsf{k}_{u'v'}(R', t') \Leftarrow B_{n+1}, \ldots, B_m \\ h &= \mathsf{i}_{u'v'}(R, R') \Leftarrow B_1, \ldots, B_m \end{aligned}$$
where $\sigma \in \mathrm{csu}_{\mathsf{AC}}(\langle u, t\rangle, \langle u', t'\rangle)$.

Let τ 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 $P \models \mathsf{i}_{u'v'}(R, R')\sigma\tau$.

As $P \models B_1\sigma\tau, \ldots, B_n\sigma\tau$ and because $P \models f$ we have that $P \models \mathsf{k}_u(R, t)\sigma\tau$. But $\mathsf{k}_u(R, t)\sigma\tau = \mathsf{k}_{u'}(R, t)\sigma\tau$ by choice of σ, and therefore $P \models \mathsf{k}_{u'}(R, t)\sigma\tau$, and we also have that $P \models \mathsf{k}_{u'v'}(R, t)\sigma\tau$. As $P \models B_{n+1}\sigma\tau\ \ldots, B_m\sigma\tau$ and because $P \models g$ we also obtain that $P \models \mathsf{k}_{u'v'}(R', t')\sigma\tau$.

As $P \models \mathsf{k}_{u'v'}(R, t)\sigma\tau$ and $P \models \mathsf{k}_{u'v'}(R', t')\sigma\tau$, we have by definition that $P \models \mathsf{i}_{u'v'}(R, R')\sigma\tau$. We have shown that the head of $h\sigma\tau$ is true in P. As τ 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 σ 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:

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

where $\sigma \in \mathrm{csu}_{\mathsf{AC}}(u,u')$.

Let τ be a substitution grounding for $h\sigma$. We assume that $P \models B_1\sigma\tau,\ldots,B_m\sigma\tau$ and we show that $P \models \mathsf{ri}_u(R,R')\tau$. Indeed, as $P \models B_1\sigma\tau,\ldots,B_n\sigma\tau$ and as $P \models f$, we have that $P \models \mathsf{i}_u(R,R')\sigma\tau$. As $P \models B_{n+1}\sigma\tau,\ldots,B_m\sigma\tau$ and as $P \models g$, we have that $P \models \mathsf{r}_{u'v'}\sigma\tau$.

But $\sigma \in \mathrm{csu}_{\mathsf{AC}}(u,u')$ and therefore $u\sigma\tau =_{\mathsf{AC}} u'\sigma\tau$. Hence, we immediately obtain $P \models \mathsf{ri}_{u'v'}(R,R')\sigma\tau$, which is what we wanted. It follows that $P \models h\sigma$. ☐

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 $P \models f$ for any $f \in (K \Cup g)$.

Proof. If $K \Cup g \subseteq K \cup \{g\Downarrow\}$, we immediately conclude by Lemma 5. Otherwise, we have that $K \Cup g = K \cup \{h\}$ where:

- $g\Downarrow = \mathsf{k}_u(R,t) \Leftarrow \mathsf{k}_{u_1}(X_1,x_1),\ldots,\mathsf{k}_{u_n}(X_n,x_n)$ for some $R,t,u,u_1,\ldots,u_n,X_1,\ldots,X_n,x_1,\ldots,x_n$; and
- $h = \mathsf{i}_u(R,R'\!\downarrow) \Leftarrow \mathsf{k}_{u_1}(X_1,x_1),\ldots,\mathsf{k}_{u_n}(X_n,x_n)$.

Moreover, we have that $g' \in \mathbf{conseq}(K_{\mathsf{solved}})$ where

$$g' = \mathsf{k}_u(R',t) \Leftarrow \mathsf{k}_{u_1}(X_1,x_1),\ldots,\mathsf{k}_{u_n}(X_n,x_n).$$

In order to conclude, we have to show that $P \models h$. Let τ 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 $g\Downarrow\tau$ and those of $g'\tau$, and as $P \models g$ and $P \models g'$ (thanks to Lemma 6), we have that $P \models \mathsf{k}_u(R,t)\tau$, $P \models \mathsf{k}_u(R',t)\tau$, and $P \models \mathsf{k}_u(R'\!\downarrow,t)\tau$ as well. But this immediately implies that $P \models \mathsf{i}_u(R,R'\!\downarrow)\tau$ (the head of $h\tau$). As τ was chosen arbitrarily, it follows that $P \models h$. ☐

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 $H \in \mathcal{H}_{\mathsf{e}}(K)$.

Proof. This result is proved by structural induction on the derivation tree witnessing the fact that $H \in \mathcal{H}_{\mathsf{e}}(K)$. ☐

B. Completeness

In this section we prove the completeness part of Theorem 2. The first two 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 $\mathcal{H}'(K)$, $\mathcal{H}'_+(K)$, $\mathcal{H}'_-(K)$, $\mathcal{H}'^{\mathsf{s}}_-(K)$ and $\mathcal{H}'^{\mathsf{e}}_-(K)$ to be the smallest sets of ground facts such that:

- $\mathcal{H}'_-(K) = \mathcal{H}'^{\mathsf{s}}_-(K) \cup \mathcal{H}'^{\mathsf{e}}_-(K)$,
- $\mathcal{H}'(K) = \mathcal{H}'_-(K) \cup \mathcal{H}'_+(K)$, and

$$\textsc{Std}\ \frac{f = \big(H \Leftarrow B_1,\ldots,B_n\big) \in K \setminus f_0^+ \qquad B_1\sigma,\ldots,B_n\sigma \in \mathcal{H}'(K) \text{ with } \mathsf{skel}(f\sigma) \text{ in normal form}}{H\sigma \in \mathcal{H}'^{\mathsf{s}}_-(K)}$$

$$\textsc{Sum}\ \frac{f = \big(H \Leftarrow B_1,B_2\big) \in f_0^+ \qquad B_1\sigma,B_2\sigma \in \mathcal{H}'(K) \text{ with } \mathsf{skel}(f\sigma) \text{ in normal form}}{H\sigma \in \mathcal{H}'_+(K)}$$

$$\textsc{Extend}'\ \frac{\mathsf{k}_u(R,t) \in \mathcal{H}'^{\mathsf{s}}_-(K)}{\mathsf{k}_{uv}(R,t) \in \mathcal{H}'^{\mathsf{e}}_-(K)}$$

The sets of derivations associated to $\mathcal{H}(K)$, $\mathcal{H}'(K)$, $\mathcal{H}'_-(K)$, $\mathcal{H}'^{\mathsf{s}}_-(K)$, $\mathcal{H}'^{\mathsf{e}}_-(K)$ and $\mathcal{H}'_+(K)$ are respectively denoted by $\Pi(K)$, $\Pi'(K)$, $\Pi'_-(K)$, $\Pi'^{\mathsf{s}}_-(K)$, $\Pi'^{\mathsf{e}}_-(K)$ and $\Pi'_+(K)$.

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

1) the *extraction* $\mathrm{extract}(\pi)$ *of a derivation tree* $\pi \in \Pi'(K)$ is the set of derivation trees $\{\pi\}$ in case $\pi \in \Pi'_-(K)$; and $\mathrm{extract}(\pi_1) \cup \mathrm{extract}(\pi_2)$ where π_1 and π_2 are the immediate sub-derivation trees of π in case $\pi \in \Pi'_+(K)$.
2) the *size* $\mathcal{S}(\pi)$ *of a derivation tree* $\pi \in \Pi'(K)$ is the number of nodes occurring in π excluding EXTEND' nodes.
3) the *real world* $\mathcal{W}(\pi)$ *of a derivation tree* $\pi \in \Pi'(K)$ whose root is labeled with $\mathsf{k}_w(R,t)$ is w in case $\pi \notin \Pi'^{\mathsf{e}}_-(K)$, and $\mathcal{W}(\pi) = \mathcal{W}(\pi')$ where π' is the immediate sub-derivation tree of π otherwise.
4) the *l-restricted size* $\hat{\mathcal{S}}_l(\pi)$ *of a derivation tree* $\pi \in \Pi'(K)$ whose immediate sub-derivation tree are $\pi_1,\ldots,\pi_n$ is

$$\sum_{1 \le i \le n} \{\mathcal{S}(\pi_i) \mid \mathcal{W}(\ \pi_i) \ge l\}$$

Such a measure allows us to define the set $\mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$ with $l,\kappa \in \mathbb{N}$ as the smallest set of facts that:

- contains any $H \in \mathcal{H}'(K)$ which has a derivation tree $\pi \in \Pi'(K)$ such that $\hat{\mathcal{S}}_l(\pi) \le \kappa$; and
- that is closed under the same rules as $\mathcal{H}_{\mathsf{e}}$.

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 $\mathcal{H}_{\mathsf{e}}(K)$ instead of $\mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$.

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 $\pi \in \Pi'_+(K)$ with its root labeled with $\mathsf{k}_u(R,\bigoplus_{i=1}^n t_i)$ (and no t_i is a sum). Let $k \in \{1,\ldots,n\}$. There exists a partition of

$\{1,\dots,n\}$ in A, B, and a derivation tree $\pi' \in \Pi'_+(K)$ with its root labeled with $\mathsf{k}_u(R', \bigoplus_{i=1}^n t_i)$ such that:

- $k \in A$;
- $R =_{\mathsf{AC}} R'$;
- $\mathcal{S}(\pi') = \mathcal{S}(\pi)$;
- the immediate sub-derivation trees of π' are:
  - $\pi'_1 \in \Pi'_-(K)$ with root $h'_1 = \mathsf{k}_u(R_A, \bigoplus_{i\in A} t_i)$; and
  - $\pi'_2 \in \Pi'(K)$ with root $h'_2 = \mathsf{k}_u(R_B, \bigoplus_{i\in B} t_i)$.

Proof. We proceed by induction on $\mathcal{S}(\pi)$. Let π_1 and π_2 be the immediate sub-derivations of π, respectively labeled with h_1 and h_2. We may assume w.l.o.g. that t_k occurs in h_1. If $\pi_1 \in \Pi'_-(K)$, we conclude immediately. We now consider that $\pi_1 \in \Pi'_+(K)$. We apply our induction hypothesis on π_1 to get $\pi_a \in \Pi'_-(K)$ and $\pi_b \in \Pi'(K)$ labeled respectively with h_a and h_b, and such that $\mathcal{S}(\pi_a) + \mathcal{S}(\pi_b) + 1 = \mathcal{S}(\pi_1)$. We apply SUM on π_b and π_2 to obtain π_3.

We can conclude with π'_1 being π_a, π'_2 being π_3, and π' the result of applying SUM on π'_1 and π'_2. We can check that $\mathcal{S}(\pi') = \mathcal{S}(\pi'_1) + \mathcal{S}(\pi'_2) + 1 = \mathcal{S}(\pi)$. ☐

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

Proposition 2: Let K be a solved knowledge base such that $f_0^+ \subseteq K$. If $\mathsf{k}_u(R,t) \in \mathcal{H}'(K)$, then $\mathsf{k}_{uv}(R,t) \in \mathcal{H}'(K)$ for any v.

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

1) Case $\pi \in \Pi'^{\mathsf{s}}_-(K)$. In such a case, we conclude by applying rule EXTEND' to extend the world with v.
2) Case $\pi \in \Pi'^{\mathsf{e}}_-(K)$. In such a case, its last rule is already EXTEND', extending some world u_1 to $u_1u_2 = u$; we conclude by modifying this last rule to directly extend u_1 with u_2v.
3) Case $\pi \in \Pi'_+(K)$. In such a case, its two premises are of the form $\mathsf{k}_u(R_i,t_i)$ with $i \in \{1,2\}$. By induction hypothesis, we have that $\mathsf{k}_{uv}(R_i,t_i) \in \mathcal{H}'(K)$ for $i \in \{1,2\}$, which allows us to conclude using rule SUM. ☐

Lemma 14: 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 π be a derivation tree in $\mathcal{H}(K)$ with root labeled with H. We proceed by case analysis on the last rule of π.

1) Case π ends with EXTEND. In such a case, we apply the induction hypothesis to its immediate sub-derivation, and conclude by Proposition 2.
2) Case π ends with CONSEQ. In such a case, let f be the statement and $B_1,\dots,B_n$ the premisses involved in this last step. Applying our induction hypothesis, we know that there are derivation tree $\pi'_1,\dots,\pi'_n \in \Pi'(K)$ labeled respectively with $B_1,\dots,B_n$. We then distinguish two cases:
   - If $f \notin f_0^+$ we can use STD on $\pi'_1,\dots,\pi'_n$ to obtain $H \in \mathcal{H}'_-(K)$.
   - If $f \in f_0^+$, we proceed similarly but using rule SUM instead of STD. ☐

Proposition 3: Let K be a solved knowledge base, and let $f = \Big(\mathsf{k}_{uv}(R,t) \Leftarrow C_1,\dots,C_m\Big)$ a statement such that $f \in \mathbf{conseq}(K)$ and τ be a substitution grounding for f such that $\mathsf{skel}(f\tau)$ is in normal form, and for all $1 \le i \le n$ we have that $C_i\tau$ has a derivation tree $\pi_i \in \mathcal{H}(K)$. Then $\mathsf{k}_{uv}(R,t)\tau$ has a derivation tree $\pi \in \mathcal{H}(K)$.

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

- If the AXIOM rule was used, we have that $C_i = \mathsf{k}_u(R,t)$ for some i and, by hypothesis, $C_i\tau \in \mathcal{H}(K)$. Using the EXTENDK rule we conclude.
- If the RES rule was used, we have that there exists a solved statement $g = \Big(\mathsf{k}_{u'}(R',t') \Leftarrow B_1,\dots,B_n\Big) \in K$ and a substitution σ such that $\mathsf{k}_u(R,t) = \mathsf{k}_{u'}(R',t')\sigma$ and $B_i\sigma \Leftarrow C_1,\dots,C_m \in \mathbf{conseq}(K)$ $(1 \le i \le n)$. We assume w.l.o.g. that σ only instantiates first-order variables that occur in u', and t'. Therefore we have that $\mathsf{skel}(B_i\sigma\tau)$ 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 $\mathsf{skel}(B_i\sigma\tau)$ is in normal form, we can apply our induction hypothesis, and conclude that $B_i\sigma\tau \in \mathcal{H}(K)$.
  As $\Big(\mathsf{k}_{u'}(R',t') \Leftarrow B_1,\dots,B_n\Big) \in K$ and $\mathsf{skel}(B_i\sigma\tau)$ are all in normal form, it follows that $\mathsf{k}_{u'}(R',t')\sigma\tau = \mathsf{k}_u(R,t)\tau \in \mathcal{H}(K)$ with derivation tree π. Using the EXTENDK rule we conclude. ☐

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

Lemma 15: Let K be a solved knowledge base.

1) If $\mathsf{i}_w(R_1, R_2 \oplus R_3) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$ then $\mathsf{i}_w(R_1 \oplus R_2, R_3) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$.
2) If $\mathsf{i}_w(R_1, R_2) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$ and $\mathsf{i}_w(R_2, R_3) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$ then $\mathsf{i}_w(R_1, R_3) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$.
3) If $\mathsf{i}_w(R_1, R_2) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$ then $\mathsf{i}_w(R_2, R_1) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$.

Proof. We prove the three items separately.

1) We apply CONG using $\mathsf{i}_w(R_2, R_2) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$ (obtained with REFL) to get $\mathsf{i}_w(R_1 \oplus R_2, R_2 \oplus R_3 \oplus R_2) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$, and conclude using MOD-I.
2) Using CONG, we get $\mathsf{i}_w(R_1 \oplus R_2, R_2 \oplus R_3) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$. We apply CONG again using $\mathsf{i}_w(R_2, R_2) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$ (obtained with REFL), and conclude using MOD-I.
3) We apply twice CONG using $\mathsf{i}_w(R_1, R_1) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$ and $\mathsf{i}_w(R_2, R_2) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$ (obtained with REFL) to get $\mathsf{i}_w(R_1 \oplus R_1 \oplus R_2, R_2 \oplus R_1 \oplus R_2) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K)$, and conclude using MOD-I. ☐

Proposition 4: Let K be a knowledge base. If $\mathsf{k}_w(R[T]_p, t) \in \mathcal{H}_{\mathsf{e}}(K)$ and $\mathsf{i}_w(T, T') \in \mathcal{H}_{\mathsf{e}}(K)$, then $\mathsf{k}_w(R[T']_p, t) \in \mathcal{H}_{\mathsf{e}}(K)$.

Proof. By induction on the context R.

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

$$\mathsf{k}_w(R'',t) \in \mathcal{H}(K) \tag{1}$$

and such that $\mathsf{i}_w(R,R'') \in \mathcal{H}_\mathsf{e}(K)$. But $\mathsf{i}_w(R,R') \in \mathcal{H}_\mathsf{e}(K)$ and therefore, by Lemma 15, we have that

$$\mathsf{i}_w(R'',R') \in \mathcal{H}_\mathsf{e}(K). \tag{2}$$

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

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

Definition 17: We write $w \sqsubseteq w'$ whenever w is a prefix of w': i.e. there exists $\ell_1,\ldots,\ell_n$ such that $w' =_{\mathsf{AC}} \ell_1,\ldots,\ell_n$ and $w =_{\mathsf{AC}} \ell_1,\ldots,\ell_m$ 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:

$$f = \Big(H \Leftarrow B_1,\ldots,B_n \,\|\, \mathcal{M}\Big)$$

and $B_i = \mathsf{k}_{w_i}(X_i,t_i)$. Let σ be a substitution such that

- σ is grounding for f,
- $\mathsf{skel}(f\sigma)$ is in normal form, and
- for all $1 \leq i \leq n$ there exist $\pi_i \in \Pi'(K_{\mathsf{solved}})$ to derive $B_i\sigma$ and if $B_i \in \mathcal{M}$ then $\pi_i \in \Pi'_-(K_{\mathsf{solved}})$.

Then we have that, during the saturation of K, there was an update with a statement $h = \Big(H' \Leftarrow B'_1,\ldots,B'_{n'} \,\|\, \mathcal{M}'\Big)$ where $B'_i = \mathsf{k}_{w'_i}(X'_i,t'_i)$, and there is a substitution ω grounding for h such that:

1) $\mathsf{skel}(h\omega)$ is in normal form;
2) $H'\omega = H\sigma$;
3) for all $1 \leq i \leq n'$, there exist $\pi'_i \in \Pi'(K_{\mathsf{solved}})$ to derive $B'_i\omega$, and if $B'_i \in Marked'$ then $\pi'_i \in \Pi'_-(K_{\mathsf{solved}})$;
4) $\sum_{i\in\{1,\ldots,n\}} \mathcal{S}(\pi_i) > \sum_{i\in\{1,\ldots,n'\}} \mathcal{S}(\pi'_i)$;
5) Let $\pi \in \Pi'(K_{\mathsf{solved}})$ be the derivation tree of $H\sigma \in \mathcal{H}(K)$ with sub-derivation trees $\pi_1,\ldots,\pi_n$ and $\pi' \in \Pi'(K_{\mathsf{solved}})$ be the derivation tree of $H'\omega \in \mathcal{H}(K)$ whose sub-derivation trees are $\pi'_1,\ldots,\pi'_{n'}$. For all l we have that $\hat{\mathcal{S}}_l(\pi') \leq \hat{\mathcal{S}}_l(\pi)$.

Proof. Let $B_j = \mathsf{k}_{w_j}(X_j,t_j) = \mathsf{sel}(f)$.

Case: $\pi_j \in \Pi'_-(K_{\mathsf{solved}})$. We consider the last node of π_j or the previous node if the last node is EXTEND' with some word v'. Let

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

$g \neq f_0^+$ and σ' the statement and substitution of this node (we have $u'v'\sigma' = w_j\sigma$). Note that by definition of a derivation tree and $\mathcal{S}(.)$, we have that $\mathcal{S}(\pi_j) = 1 + \sum_{i\in\{n+1,\ldots,m\}} \pi'_i$ where $(\pi'_i)_{n+1\leq i\leq m}$ are the immediate sub-derivation tree of π_j and also proofs of $(B_i\sigma) \in \Pi'(K_{\mathsf{solved}})$. As $\sigma \cup \sigma'$ is a unifier of $\mathsf{k}_{w_j}(X_j,t_j)$ and $\mathsf{k}_{u'}(R',t')$, it follows that $\tau \in \mathrm{csu}_{\mathsf{AC}}(\mathsf{k}_{w_j}(X_j,t_j),\mathsf{k}_{u'}(R',t'))$ exists. Therefore $\sigma \cup \sigma'$ must be an instance of the one element of the complete set of unifier, let ω be a substitution such that $\sigma \cup \sigma' = \tau\omega$.

As K has been saturated by selecting B_j and $g \neq f_0^+$, 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:

$$h = \Big(H \Leftarrow B_1,\ldots,B_{j-1},B_{j+1},\ldots,B_m \,\|\, \mathcal{M}\Big)\tau.$$

We have that ω is a substitution grounding for h, that $\mathsf{skel}(h\omega)$ is in normal form, that $\pi'_i \in \Pi'(K_{\mathsf{solved}})$ is a derivation tree for $B_i\tau\omega \in \mathcal{H}'(K_{\mathsf{solved}})$ for $i \in \{1,\ldots,j-1,j+1,\ldots,m\}$, and that

$$\sum_{i\in\{1,\ldots,j-1,j+1,\ldots,m\}} \mathcal{S}(\pi'_i) < \sum_{i\in 1\ldots n} \mathcal{S}(\pi_i).$$

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

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 $\Pi'_-(K_{\mathsf{solved}})$. Therefore, Condition 3 is also satisfied.

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

Let $\sigma^+ \in \mathrm{csu}_{\mathsf{AC}}(k_w(R_j,t_j), k_w(X \oplus Y, x \oplus y))$ such that t_x is an instance of $x\sigma^+$, t_y an instance of $y\sigma^+$, R_x an instance of $X\sigma^+$ and R_y an instance of $Y\sigma^+$. We denote by ω 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^+ with σ^+ as unifier.

We consider h' the resulting statement from which K has been updated:

$$\Big(H \Leftarrow B_x, B_y, B_1,\ldots,B_{j-1},B_{j+1},\ldots,B_n \,\|\, \mathcal{M} \cup \{B_x\}\Big)\sigma^+$$

where B_x is marked, B_y is not, and the other keep their status.

We have that ω is a substitution grounding for h such that $\mathsf{skel}(h\omega)$ is in normal form, that $B_i\sigma^+\omega$ has a derivation tree $\pi'_i \in \Pi'(K_{\mathsf{solved}})$ for $i \in \{1,\ldots,j-1,j+1,\ldots,m\}$ and that $\mathcal{S}(\pi'_x) + \mathcal{S}(\pi'_y) + \sum_{i\in\{1,\ldots,j-1,j+1,\ldots,m\}} \mathcal{S}(\pi'_i) < \sum_{i\in 1\ldots n} \mathcal{S}(\pi_i)$. 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 $\Pi'_-(K_{\mathsf{solved}})$. Therefore our Condition 3 is also satisfied. Similarly, Condition 5 is also satisfied: when $\mathcal{W}(\pi_i) \geq l$ 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 $f \in K$ be a statement such that

$$f = \Big(H \Leftarrow B_1, \ldots, B_n \,\|\, \mathcal{M}\Big)$$

where H is either $\mathsf{i}_w(R, R')$, $\mathsf{ri}_w(R, R')$ or r_w. Let σ be a substitution grounding for f such that

- $\mathsf{skel}(f\sigma)$ is in normal form,
- for all $1 \leq i \leq n$ there exists a derivation tree $\pi_i \in \Pi'(K_{\mathsf{solved}})$ of $B_i\sigma$ and if $B_i \in \mathit{marked}$ then $\pi_i \in \Pi'_{\_}(K_{\mathsf{solved}})$.

For all l we have that

$$H\sigma \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K_{\mathsf{solved}})$$

where $\kappa = \hat{\mathcal{S}}_l(\pi)$ where π is the derivation tree whose immediate sub-derivation trees are the π_i's

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

If f is a solved statement, the conclusion is immediate by the definition of $\mathcal{H}_{\mathsf{e}}^{(l,\kappa)}$.

Otherwise, we apply Lemma 16 on f and σ. We get the existence of ω and $h = (H' \Leftarrow B'_1, \ldots, B'_m \,\|\, \mathcal{M}')$ such that K has been updated by h and which have the six properties stated in Lemma 16. Let $h\Downarrow = (H'' \Leftarrow B''_1, \ldots, B''_{m'} \,\|\, \mathcal{M}'')$ such that $\{B''_1, \ldots, B''_{m'}\} \subseteq \{B'_1, \ldots, B'_m\}$. Since Property 1 grants that $\mathsf{skel}(H'')$ is in normal form and that canonization rules can only replace the term of the head by a subterm, the skeleton of the considered predicates $h\Downarrow$ is still in normal form and then is not removed: it is added in the knowledge base by the update. Then due to Property 3, $\mathcal{M}''$ corresponds to the premisses which has a derivation tree in $\Pi'_{\_}(K_{\mathsf{solved}})$ and due to the Property 4 the measure is strictly smaller so we can apply our induction hypothesis on $h\Downarrow$ to get $H''\omega \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa')}$ with $\kappa' \leq \kappa$ due to the property 5 and the set inclusion. Finally, according to the Property 2 we have $H'\omega = H\sigma$ and by definition of $\Downarrow$, we have $H'{\downarrow}\omega = H''\omega$, therefore, by definition of $\mathcal{H}_{\mathsf{e}}^{(l,\kappa)}$, we have $H\sigma \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}$. ☐

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

Lemma 18: Let K be a saturated knowledge base. If $h = \mathsf{k}_w(R, t) \Leftarrow B_1, \ldots, B_n \in K_{\mathsf{solved}}$ has been obtained by applying RESOLUTION or RESOLUTION+ between a VIP statement f and $g \in K_{\mathsf{solved}}$ then t has no root variable.

Proof. Recall that a VIP statement always stems from a resolution against a solved statement of an unsolved VIP statement, except for the initial f_1^+ and f_2^+ VIP statements.

Let $f = \mathsf{k}_w(X + Y, x) \Leftarrow \mathsf{k}_w(X, x \oplus u), \mathsf{k}_w(Y, u) \,\|\, \mathcal{M} \in f_1^+$. The resolution can only be done on $\mathsf{k}_w(X, x \oplus u)$ against some $g = \mathsf{k}_v(R', t_1 \oplus t_2) \Leftarrow B'_1, \ldots, B'_n$ where neither t_1 or t_2 has a root variable (the SHIFT 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 further substitution can lead to a term with a root variable: the property holds for any solved VIP statement whose ancestor is in f_1^+.

Let f be a solved VIP statement whose ancestor is $\mathsf{k}_w(X + Y, x+y) \Leftarrow \mathsf{k}_w(X, x\oplus u), \mathsf{k}_w(Y, y\oplus u) \,\|\, \mathcal{M} \in f_2^+$. By symmetry, assume that resolution has been performed on $\mathsf{k}_w(X, x \oplus u) \in f_2^+$ with some substitution σ_1 and another solved statement. The direct child of the ancestor is $f_1 = \mathsf{k}_w(X\sigma + Y, x\sigma + y) \Leftarrow B'_1, \ldots, B'_m, \mathsf{k}_{w\sigma_1}(Y, y \oplus u\sigma_1) \,\|\, \mathcal{M}'$. The previous argument of the f_1^+ case leads to $x\sigma$ has no root variable. Let $f_2, \ldots, f_n$ be the VIP children obtained by resolution on f_{i-1} and a solved statement with substitution $\sigma_2, \ldots, \sigma_n$ but where $\mathsf{sel}(f_{i-1)} \neq \mathsf{k}_{w\sigma_1\ldots\sigma_i}(Y, y \oplus u(\sigma_1 \ldots \sigma_i))$. However all these f_i are unsolved since $y \oplus u(\sigma_1 \ldots \sigma_i)$ is not a variable. Once a resolution is done against the literal $\mathsf{k}_w(Y, y \oplus u(\sigma_1 \ldots \sigma_n)$ with substitution σ' 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 VIP statement whose ancestor is in f_2^+. ☐

Lemma 19: Let f_n be a solved VIP statement whose ancestor is $f_0 = (\mathsf{k}_w(X \oplus Y, t) \Leftarrow B_{0,1}, B_{0,2} \,\|\, \{B_{0,1}, B_{0,2}\}) \in f_1^+ \cup f_2^+$ and f_i are the intermediate statements. By symmetry, we consider that $\mathsf{sel}(f_0) = B_{0,1}$ and therefore that there exists $B_{1,2} = B_{0,2}\sigma$ in the premisses of f_1. If $B_{i,2}$ exists in the premisses of f_i and $\mathsf{sel}(f_i) \neq B_{i,2}$ the resulting statement has a premisse $B_{i+1,2}$. The statement f_n has no premisse $B_{n,2}$.

Proof. We consider two possible cases for the ancestor.

- If $f_0 = \mathsf{k}_w(X + Y, x + y) \Leftarrow \mathsf{k}_w(X, x \oplus u), \mathsf{k}_w(Y, y \oplus u) \,\|\, \mathcal{M} \in f_2^+$ we observe that none of the two premisses are solved. As f_n is solved both premisses must have been removed by resolution and the result is immediate.
- If $f_0 = \mathsf{k}_w(X + Y, x) \Leftarrow \mathsf{k}_w(X, x \oplus u), \mathsf{k}_w(Y, u) \,\|\, \mathcal{M} \in f_1^+$, resolution can only be performed on $\mathsf{k}_w(X, x \oplus u)$ against some $g = \mathsf{k}_v(R', t_1 \oplus t_2) \Leftarrow B'_1, \ldots, B'_n$ 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 $\mathsf{k}_w(Y, t')$ is required to get a solved statement. ☐

Lemma 20: Let K be a saturated knowledge base and h a solved statement

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

such that $h\Downarrow = \Big(\mathsf{k}_u(R', t') \Leftarrow B'_1, \ldots, B'_m\Big)$, t is in normal form and

- either $h\Downarrow \in K_{\mathsf{solved}}$,
- or $g = \Big(\mathsf{k}_u(R'', t') \Leftarrow B'_1, \ldots, B'_m\Big) \in \mathbf{conseq}(K_{\mathsf{solved}})$ and $h' = \Big(\mathsf{i}_u(R'{\downarrow}, R''{\downarrow}) \Leftarrow B'_1, \ldots, B'_m\Big) \in K_{\mathsf{solved}}$ for some R''.

Let σ be a substitution grounding for h such that $\mathsf{skel}(h\sigma)$ is in normal form, and for all $1 \leq i \leq n$, we have that $B_i\sigma$ has a derivation tree $\pi_i \in \Pi'(K_{\mathsf{solved}})$. We have that

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

Proof. First note that by definition of the canonical form $\{B'_1,\dots,B'_m\} \subseteq \{B_1,\dots,B_n\}$ and σ 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 σ, we have that
   $$\mathsf{k}_u(R'',t')\sigma \in \mathcal{H}(K_{\mathsf{solved}}).$$
   Furthermore, as $h' \in K_{\mathsf{solved}}$ and as all antecedents $B'_1\sigma,\dots,B'_m\sigma$ of $h'\sigma$ are in $\mathcal{H}(K_{\mathsf{solved}})$, we have that
   $$\mathsf{i}_u(R'\downarrow,R''\downarrow)\sigma \in \mathcal{H}(K_{\mathsf{solved}}).$$
   It immediately follows that
   $$\mathsf{k}_u(R',t')\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}),$$
   which is what we had the prove since $\mathsf{k}_u(R',t')\sigma = \mathsf{k}_u(R,t)\sigma$.
2) If the rule REMOVE has been applied, the conclusion is immediate as REMOVE only remove premises.
3) If there is a renormalization of the recipe, we conclude with MOD-I and EQ. CONSEQ.
4) If the rule SHIFT has be applied to $h = \mathsf{k}_u(R,t) \Leftarrow \mathsf{k}_w(X,x),B_1,\dots,B_n$ for some w such that $w \sqsubseteq u$ to get $h' = \mathsf{k}_u(R\oplus X, t\oplus x\downarrow) \Leftarrow \mathsf{k}_w(X,x),B_1,\dots,B_n$. By induction hypothesis, we have $\mathsf{k}_u(R\oplus X,t\oplus x\downarrow)\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. By definition of $\mathcal{H}_{\mathsf{e}}$, there is R''' such that $\mathsf{k}_{u\sigma}(R''',t\oplus x\downarrow\sigma) \in \mathcal{H}(K_{\mathsf{solved}})$ and $\mathsf{i}_{u\sigma}(R''',R\oplus X\sigma) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. We distinguish two cases.
   - If $t \neq x$, since we also have $\mathsf{k}_u(X,x)\sigma \in \mathcal{H}(K_{\mathsf{solved}})$, we get by the rule CONSEQ on f_0^+ that $\mathsf{k}_{u\sigma}(R'''\oplus X\sigma,((t\oplus x\downarrow)\oplus x)\sigma) \in \mathcal{H}(K_{\mathsf{solved}})$ (and $(t\oplus x\downarrow)\oplus x =_{\mathsf{AC}} t$). From $\mathsf{i}_{u\sigma}(R''',R\oplus X\sigma) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$, $\mathsf{i}_u(X,X)\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$ and the CONG with $\oplus$, we get $\mathsf{i}_{u\sigma}(R'''\oplus X\sigma,(R\oplus X\oplus X)\sigma) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. From MOD-I, we get $\mathsf{i}_{u\sigma}(R'''\oplus X\sigma, R\sigma) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. Finally from $\mathsf{k}_{u\sigma}(R'''\oplus X\sigma,((t\oplus x\downarrow)\oplus x)\sigma) \in \mathcal{H}(K_{\mathsf{solved}})$, applying EQ. CONSEQ., we obtain that $\mathsf{k}_u(R,t)\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$.
   - If $t = x$, from $\mathsf{k}_{u\sigma}(R''',0) \in \mathcal{H}(K_{\mathsf{solved}})$ and since the base is saturated and contains $\mathsf{k}_w(0,0)$, rule EQUATION has been applied between $\mathsf{k}_w(0,0) \Leftarrow$ and f the statement of the root node of the derivation tree of $\mathsf{k}_{u\sigma}(R''',0) \in \mathcal{H}(K_{\mathsf{solved}})$. This leads to $\mathsf{i}_{u\sigma}(R''',0) \in \mathcal{H}(K_{\mathsf{solved}})$. By Lemma 15, from $\mathsf{i}_{u\sigma}(R''',0) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$ and $\mathsf{i}_{u\sigma}(R''',R\oplus X\sigma) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$, we get $\mathsf{i}_u(R,X)\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. Since $\mathsf{k}_u(X,x)\sigma \in \mathcal{H}(K_{\mathsf{solved}})$, we conclude. □

Lemma 21: Let K be a saturated knowledge base and $f \in K$ be a statement such that

$$f = \left(\mathsf{k}_w(R,t) \Leftarrow B_1,\dots,B_n \parallel \mathcal{M}\right)$$

and $B_i = \mathsf{k}_{w_i}(X_i,t_i)$. Let σ be a substitution grounding for f such that $\mathsf{skel}(f\sigma)$ is in normal form, and for all $1 \leq i \leq n$, we have that $B_i\sigma$ has a derivation tree π_i in $\Pi'(K_{\mathsf{solved}})$, and if $B_i \in \mathcal{M}$ then $\pi_i \in \Pi'_-(K_{\mathsf{solved}})$. We have that

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

Moreover, if $f \in f_1^+ \cup f_2^+$, we have for all l that $\hat{\mathcal{S}}_l(\pi) \leq \hat{\mathcal{S}}_l(\pi_1) + \hat{\mathcal{S}}_l(\pi_2)$.

Proof. Let $\mathcal{G} = \sum_{i\in\{1,\dots,n\}} \mathcal{S}(\pi_i)$. We prove the lemma by induction on $\mathcal{G}$ with a stronger assumption to prove the measure on f_1^+ and f_2^+. If f is a VIP statement, from $(\mathsf{k}_w(R,t))\sigma \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$, we extract the derivation tree π of $(\mathsf{k}_w(R',t))\sigma \in \mathcal{H}'(K_{\mathsf{solved}})$ with $R'\downarrow = R\downarrow$ whose root is $h^f = H^f \Leftarrow B_1^f,\dots,B_m^f,\sigma^f$ and $\pi_i^f \in \Pi'(K_{\mathsf{solved}})$ derivation trees of $B_i^f\sigma^f$, then for all n we have that

$$\begin{aligned}\hat{\mathcal{S}}_l(\pi) \leq\ & \textstyle\sum_{\{i\mid\mathcal{W}(\pi_i)\geq l\wedge\exists j,\ \pi_i=\pi_j^f\}} \mathcal{S}(\pi_i)+\\ & \textstyle\sum_{\{i\mid\mathcal{W}(\pi_i)\geq l\wedge\neg\exists j,\ \pi_i=\pi_j^f\}} \hat{\mathcal{S}}_l(\pi_i).\end{aligned}$$

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 $\mathcal{H}_{\mathsf{e}}$: when f is solved, we have $\pi_i = \pi_i^f$, so the inequality of the measure is just $\hat{\mathcal{S}}_l(\pi) \leq \sum_{\{i\mid\mathcal{W}(\pi_i)\geq l\}} \mathcal{S}(\pi_i)$ which is the definition of $\hat{\mathcal{S}}_l(\pi)$.

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

$$h = \left(\mathsf{k}_u(R_h,t_h) \Leftarrow B'_1,\dots,B'_m \parallel \mathcal{M}'\right)$$

such that K has been updated by h and which have the five properties stated in Lemma 16. Let $h\Downarrow = (\mathsf{k}_u(R'_h,t'_h) \Leftarrow B''_1,\dots,B''_{m'} \parallel \mathcal{M}'')$ such that $\{B''_1,\dots,B''_{m'}\} \subseteq \{B'_1,\dots,B'_m\}$. Since Property 1 grants that $\mathsf{skel}(h\omega)$ is in normal form and that the SHIFT rule cannot produce non normal skeletons while the other rule does not alter $\mathsf{skel}(\mathsf{k}_u(R'_h,t'_h))$, $h\Downarrow$ is not removed.

1) Case where h is not solved. Due to Property 3, $\mathcal{M}''$ corresponds to the premisses which has a derivation tree in $\Pi'_-(K_{\mathsf{solved}})$ and due to the Property 4 the measure is strictly smaller so we can apply our induction hypothesis on $h\Downarrow$ to get $\mathsf{k}_u(R'_h,t'_h)\omega \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. Finally, according to the Property 2, we have $\mathsf{k}_u(R_h,t_h)\omega = \mathsf{k}_w(R,t)\sigma$ and by definition of $\Downarrow$ when the statement is not solved, we have $\mathsf{k}_u(R'_h\downarrow,t'_h)\omega = \mathsf{k}_u(R_h,t_h)\omega$, therefore, by definition of $\mathcal{H}_{\mathsf{e}}$, we have $\mathsf{k}_u(R,t)\omega \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. Finally, we consider that f is a VIP statement. In that case, t_h cannot be of the form $t\oplus x$ according to Lemma 18, so the rule SHIFT is not applied. From our induction hypothesis, we have

$$\hat{\mathcal{S}}_l(\pi) \leq \sum_{\{i\mid\exists j,\ \pi'_i=\pi_j^f\}} \mathcal{S}(\pi'_i) + \sum_{\{i\mid\neg\exists j,\ \pi'_i=\pi_j^f\}} \mathcal{S}(\pi'_i).$$

From Remark 2, since $\forall x, \mathcal{S}(x) > \hat{\mathcal{S}}_l(x)$, we have

$$\hat{\mathcal{S}}_l(\pi) \leq \quad \sum_{\{i \mid \mathcal{W}(\pi_i) \geq l \wedge \exists j,\ \pi_i = \pi_j^f\}} \mathcal{S}(\pi_i) + \sum_{\{i \mid \mathcal{W}(\pi_i) \geq l \wedge \neg \exists j,\ \pi_i = \pi_j^f\}} \hat{\mathcal{S}}_l(\pi_i).$$

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 $h\Downarrow$ has been added to K. We can then conclude since $h\Downarrow$ is just the renormalization of the head and the removal of some premisses (with Remark 2 of Lemma 16, we have $\hat{\mathcal{S}}_l(\pi) \leq \sum_i \mathcal{S}(\pi_i)$). 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 $\mathsf{r}_u \in \mathcal{H}(K_{\mathsf{solved}})$, $\mathsf{i}_u(R, R') \in \mathcal{H}(K_{\mathsf{solved}})$ then there exist a derivation tree $\pi \in \Pi'(K_{\mathsf{solved}})$ of $\mathsf{i}_u(R, R')$ and $\mathsf{ri}_u(R, R') \in \mathcal{H}_{\mathsf{e}}^{(|u|,\kappa)}(K_{\mathsf{solved}})$ with $\kappa = \hat{\mathcal{S}}_{|u|}(\pi)$.

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

As $\mathsf{i}_u(R, R') \in \mathcal{H}(K_{\mathsf{solved}})$, there exists, by Lemma 14, $\pi \in \Pi'(K_{\mathsf{solved}})$ of $\mathsf{i}_u(R, R')$. π is such that there exists at its root or below the EXTEND' root node a solved statement

$$g = \Big(\mathsf{i}_w(T, T') \Leftarrow B_{n+1}, \ldots, B_m\Big)$$

and a substitution τ grounding for g such that $\mathsf{skel}(g\tau)$ is in normal form, and $B_i\tau \in \mathcal{H}(K_{\mathsf{solved}})$ with derivation tree π_i for all $n+1 \leq i \leq m$ and such that $u \sqsupseteq u' \sqsupseteq w\tau$, $R = T\tau$ and $R' = T'\tau$.

As $v\sigma = u \sqsupseteq w\tau$, it follows that $v = v_0v_1$ such that v_0 and w are unifiable ($\sigma \cup \tau$ is such a unifier). Let $\omega \in \mathrm{csu}_{\mathsf{AC}}(v_0, w)$ and let θ be such that $\sigma \cup \tau = \omega \circ \theta$.

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

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

We have that $\mathsf{skel}(h\theta)$ is in normal form, and therefore $\mathsf{skel}(h)$ is in normal form too. As h is not a deduction fact, and T, T' are in normal form, the update must have simply added h to K and therefore $h \in K$.

We have that $B_i\omega\theta = B_i\sigma \in \mathcal{H}(K_{\mathsf{solved}})$ for all $1 \leq i \leq n$ and that $B_i\omega\theta = B_i\tau \in \mathcal{H}(K_{\mathsf{solved}})$ for all $n+1 \leq i \leq m$ with derivation tree π_i. By applying Lemma 17 to the statement h and the substitution θ (note that its head is in normal form and $|w_i| < |u|$ for $i \in \{1, \ldots, n\}$), we obtain that $\mathsf{ri}_v(T, T')\omega\theta\downarrow = \mathsf{ri}_u(R, R')\downarrow$ and therefore $\mathsf{ri}_u(R, R') \in \mathcal{H}_{\mathsf{e}}^{(|u|,\kappa)}(K_{\mathsf{solved}})$ with $\kappa \leq \sum_{i \in \{n+1, \ldots, m\}} \mathcal{S}(\pi_i) = \hat{\mathcal{S}}_{|u|}(\pi)$. □

Lemma 23: Let K be a saturated knowledge base. If there is a derivation tree π in $\Pi'_{\_}(K_{\mathsf{solved}})$ of $\mathsf{k}_u(R, t)$ and a derivation tree π' in $\Pi'_{\_}(K_{\mathsf{solved}})$ of $\mathsf{k}_{uv}(R', t)$ then for all l we have that $\mathsf{i}_w(R, R') \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K_{\mathsf{solved}})$ for some $w \sqsubseteq uv$ with $\kappa \leq \hat{\mathcal{S}}_l(\pi) + \hat{\mathcal{S}}_l(\pi')$.

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

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

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

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

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

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

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

We have that $\sigma \cup \sigma'$ is a unifier of $\mathsf{k}_{\ell'_1,\ldots,\ell'_r}(\_, s)$ and $\mathsf{k}_{\ell''_1,\ldots,\ell''_r}(\_, s')$, it follows that $\tau \in \mathrm{csu}_{\mathsf{AC}}(\mathsf{k}_{\ell'_1,\ldots,\ell'_r}(\_, s), \mathsf{k}_{\ell''_1,\ldots,\ell''_r}(\_, s'))$ exists. As K is saturated and $f, f' \neq f_0^+$, it follows that the statement

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

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 $\mathsf{skel}(f\sigma)$ and $\mathsf{skel}(f'\sigma')$ are in normal form, we have that $\mathsf{skel}(h)$ 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 $\mathsf{k}_{\ell'_1,\ldots,\ell'_r}(\_, s)$ and $\mathsf{k}_{\ell''_1,\ldots,\ell''_r}(\_, s')$ and as $\tau \in \mathrm{csu}_{\mathsf{AC}}(\mathsf{k}_{\ell'_1,\ldots,\ell'_r}(\_, s), \mathsf{k}_{\ell''_1,\ldots,\ell''_r}(\_, s'))$, it follows that there exists ω such that $\sigma \cup \sigma' = \tau\omega$.

We have that ω is grounding for h and that $B_1\tau\omega, \ldots, B_n\tau\omega, B'_1\tau\omega, \ldots, B'_m\tau\omega \in \mathcal{H}(K_{\mathsf{solved}})$ with derivation trees $\pi_1, \ldots, \pi_n, \pi'_1, \ldots, \pi'_m$ and

$$\sum_{\{i \mid \mathcal{W}(\pi_i) \geq l\}} \mathcal{S}(\pi_i) + \sum_{\{i \mid \mathcal{W}(\pi'_i) \geq l\}} \mathcal{S}(\pi'_i) = \hat{\mathcal{S}}_l(\pi) + \hat{\mathcal{S}}_l(\pi')$$

where $B_i = \mathsf{k}_{w_i}(X_i, x_i)$ and $B'_i = \mathsf{k}_{w'_i}(X'_i, x'_i)$. Therefore, we have by Lemma 17 that

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

where $\kappa \leq \sum_{\{i \mid \mathcal{W}(\pi_i) \geq l\}} \mathcal{S}(\pi_i)$. As $(\ell'_1, \ldots, \ell'_r)\sigma$ is a prefix of uv, we conclude. □

Lemma 24: Let K be a saturated knowledge base. Let $\{\mathsf{k}_{w_i}(R_i, t_i) \mid 1 \leq i \leq n\}$ be a set of deduction statement such that $\pi_i \in \Pi'_{\_}(K_{\mathsf{solved}})$ is a derivation tree for $\mathsf{k}_{w_i}(R_i, t_i)$, $\bigoplus_{i \in \{1, \ldots, n\}} t_i\downarrow = 0$ and $w_i \in \{u, uv\}$ for some u, v. We have that $\mathsf{i}_{uv}(\bigoplus R_i, 0) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K_{\mathsf{solved}})$ for all l and with $\kappa \leq \sum_i \hat{\mathcal{S}}_l(\pi_i)$.

Proof. We denote $\mathsf{k}_{w_i}(R_i, t_i)$ by K_i. W.l.o.g. for all i, we suppose that $t_i = \bigoplus_{1 \leq j \leq s(i)} t_i^j$. We prove this lemma by induction on $\sum_i s(i)$. If there is some $\mathsf{k}_{w_i}(R_i, 0)$, from the seed, we have $\mathsf{k}_{w_i}(0, 0)$ which has a derivation tree π_0 in $\Pi'_{-}(K_{\mathsf{solved}})$ and $\hat{\mathcal{S}}_l(\pi_0) = 0$. By Lemma 23, we get $\mathsf{i}_w(R_i, 0) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K_{\mathsf{solved}})$ with $\kappa \leq \hat{\mathcal{S}}_l(\pi_i) + \hat{\mathcal{S}}_l(\pi_0)$. We conclude by induction hypothesis on $\{K_p | p \neq i\}$ and by use the CONG rule with $\oplus$.

Otherwise, since $\bigoplus_i t_i\downarrow = 0$, there exist at least K_k of derivation tree π_k, K_j of derivation tree π_j such that $t_k = t'_s \oplus t'_k$ and $t_j = t'_s \oplus t'_j$ for some t'_s, t'_k, t'_j. Let t_s be the t'_s of maximal size which satisfies the former relation. We define S_k to be the set $\{i | \bigoplus_i t_k^i = t_s\}$ and define S_j similarly.

If $t_k = t_j$, from Lemma 23, we get $\mathsf{i}_w(R_k, R_j) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K_{\mathsf{solved}})$ with $\kappa \leq \hat{\mathcal{S}}_l(\pi_k) + \hat{\mathcal{S}}_l(\pi_j)$. From Lemma 15, we get $\mathsf{i}_w(R_k \oplus R_j, 0) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K_{\mathsf{solved}})$. If the set contained other predicate than K_k and K_j, we use our induction hypothesis on the set $\{K_i | i \notin \{k, j\}\}$, we get $\mathsf{i}_{uv}(\bigoplus_{i \notin \{k,j\}} R_i, 0) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa')}(K_{\mathsf{solved}})$ with $\kappa' \leq \sum_{i \notin \{k,j\}} \hat{\mathcal{S}}_l(\pi_i)$. Then we conclude by applying the CONG rule with $\oplus$, on the two above identical predicates. Otherwise we conclude directly.

Otherwise $t_k \neq t_j$. We denote by $t_N = \bigoplus_{i \notin S_k} t_k^i \oplus \sum_{i \notin S_k} t'^i_j$. We apply Lemma 21 on the suitable f_1^+ or f_2^+ statement (f_1^+ if $t_s = t_k$ or $t_s = t_j$, f_2^+ otherwise) and $\sigma^+ : x \mapsto t_j, y \mapsto t_k$. We get $\mathsf{k}_{uv}(R_j \oplus R_k, t_N) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa_1)}(K_{\mathsf{solved}})$ with $\kappa_1 \leq \hat{\mathcal{S}}_l(\pi_k) + \hat{\mathcal{S}}_l(\pi_j)$. .

We apply our induction hypothesis on the set $\{K_i | i \notin \{k, j\}\}$ and obtain that $\mathsf{i}_w(\bigoplus_{i \notin \{k,j\}} R_i, 0) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa_2)}(K_{\mathsf{solved}})$ with $\kappa_2 \leq \sum_{i \notin \{k,j\}} \hat{\mathcal{S}}_l(\pi_i)$. Finally, using CONG, MOD-I and Lemma 15, we get $\mathsf{i}_{uv}(\sum_{i \notin \{k,j\}} R_i \oplus R_k \oplus R_j\downarrow, 0) \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K_{\mathsf{solved}})$ with $\kappa \leq \sum_i \hat{\mathcal{S}}_l(\pi_i)$ which allows to conclude. ☐

Corollary 1: Let K be a saturated knowledge base. If $\mathsf{k}_u(R, t)$ has a derivation tree $\pi \in \Pi'(K_{\mathsf{solved}})$ and $\mathsf{k}_{uv}(R', t)$ has a derivation tree $\pi' \in \Pi'(K_{\mathsf{solved}})$ then $\mathsf{i}_{uv}(R, R') \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa)}(K_{\mathsf{solved}})$ with $\kappa \leq \hat{\mathcal{S}}_l(\pi) + \hat{\mathcal{S}}_l(\pi')$.

Proof. If π and $\pi' \in \Pi'_{-}(K_{\mathsf{solved}})$, we apply Lemma 23 to get $\mathsf{i}_w(R, R') \in \mathcal{H}_{\mathsf{e}}^{(l,\kappa')}(K_{\mathsf{solved}})$ with $\kappa' \leq \hat{\mathcal{S}}_l(\pi) + \hat{\mathcal{S}}_l(\pi')$. Otherwise, it is a direct consequence of Lemma 24 applied on the set of conclusions of $\mathrm{extract}(\pi) \cup \mathrm{extract}(\pi')$ and Lemma 15. ☐

Lemma 25: If $\mathsf{k}_w(R_1, t) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$ and $\mathsf{k}_w(R_2, t) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$ then $\mathsf{i}_w(R_1, R_2) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$

Proof. By definition of $\mathcal{H}_{\mathsf{e}}$, there exist $\mathsf{k}_w(R'_1, t) \in \mathcal{H}(K_{\mathsf{solved}})$, $\mathsf{k}_w(R'_2, t) \in \mathcal{H}(K_{\mathsf{solved}})$ such that $\mathsf{i}_w(R_1, R'_1) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$ and $\mathsf{i}_w(R_2, R'_2) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. From Corollary 1, we got that $\mathsf{i}_w(R'_1, R'_2) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. Therefore, using Lemma 15 we derive that $\mathsf{i}_w(R_1, R_2) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. ☐

Proposition 5: If $\mathsf{k}_u(R, t) \in \mathcal{H}_{\mathsf{e}}(K)$ then $\mathsf{k}_{uv}(R, t) \in \mathcal{H}_{\mathsf{e}}(K)$.

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

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

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

$$\mathsf{k}_w(\bigoplus_{1 \leq i \leq m} R_i, \bigoplus_{1 \leq i \leq m} t_i\downarrow) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

Proof. We proceed by induction on $\sum_{1 \leq i \leq m} \mathbf{width}(t_i)$.

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 $\mathsf{k}_w(R_j \oplus R_k, t_j \oplus t_k)$ from f_0^+, π_j and π_k. We conclude by applying our induction hypothesis.

Inductive case. $(\bigoplus_{1 \leq i \leq m} t_i)\downarrow \neq \bigoplus_{1 \leq i \leq m} t_i$. If there exists j such that $\pi_j \in \Pi'_{+}(K_{\mathsf{solved}})$, then we consider π_a and π_b its immediate sub-trees which prove $\mathsf{k}_w(R_a, t_a)$ and $\mathsf{k}_w(R_b, t_b)$. We conclude by induction hypothesis on the set where $\mathsf{k}_w(R_j, t_j)$ has been replaced by $\mathsf{k}_w(R_a, t_a)$ and $\mathsf{k}_w(R_b, t_b)$. Otherwise there exist j, k, $\pi_j \in \Pi'_{-}(K_{\mathsf{solved}})$ and $\pi_k \in \Pi'_{-}(K_{\mathsf{solved}})$ such that $(t_j \oplus t_k)\downarrow \neq t_j \oplus t_k$.

- If $t_j = t_k$, from Lemma 23, we have $\mathsf{i}_w(R_j, R_k) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. From Lemma 15, we have $\mathsf{i}_w(R_j \oplus R_k, 0) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. By REFL we have $\mathsf{i}_w(\bigoplus_{i \notin \{j,k\}} R_i, \bigoplus_{i \notin \{j,k\}} R_i) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$ and using the CONG rule we obtain that

$$\mathsf{i}_w(\bigoplus_{1 \leq i \leq m} R_i, \bigoplus_{i \notin \{j,k\}} R_i) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

  By induction hypothesis, we get

$$k_w(\bigoplus_{i \notin \{j,k\}} R_i, \bigoplus_{1 \leq i \leq m} t_i\downarrow) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$$

  and we conclude by EQ. CONS.
- If $t_j \neq t_k$, there exist t', t'_j, t'_k such that $t_j =_{\mathsf{AC}} t'_j \oplus t'$ and $t_k =_{\mathsf{AC}} t'_j \oplus t'$. Applying Lemma 21 on the suitable $f \in f_1^+ \cup f_2^+$ and the derivation trees π_j and π_k we obtain $k_w(R_j \oplus R_k, t_j \oplus t_k\downarrow) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. From definition of $\mathcal{H}_{\mathsf{e}}$, there exists $k_w(R', t_j \oplus t_k\downarrow) \in \mathcal{H}'(K_{\mathsf{solved}})$ of derivation tree π' and $\mathsf{i}_w(R_j \oplus R_k, R') \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$. By REFL we have $\mathsf{i}_w(\bigoplus_{i \notin \{j,k\}} R_i, \bigoplus_{i \notin \{j,k\}} R_i) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$ and using the CONG rule we obtain that

$$\mathsf{i}_w(\bigoplus_{1 \leq i \leq m} R_i, \bigoplus_{i \notin \{j,k\}} R_i \oplus R') \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

  Since $\mathbf{width}((t_j \oplus t_k)\downarrow) < \mathbf{width}(t_j) + \mathbf{width}(t_k)$, by induction hypothesis, we get

$$k_w(\bigoplus_{i \notin \{j,k\}} R_i \oplus R', \bigoplus_{1 \leq i \leq m} t_i\downarrow) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$$

and using EQ. CONSEQ. we conclude that

$$\mathsf{k}_w(\bigoplus_{1\leq i\leq m} R_i, \bigoplus_{1\leq i\leq m} t_i\downarrow) \in \mathcal{H}_\mathsf{e}(K_\mathsf{solved}).$$

□

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

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

Case: EXTENDK. we have that $H = \mathsf{k}_w(R,t)$ and $\mathsf{k}_u(R,t) \in \mathcal{H}(S)$ for some prefix u of w, in which case by the induction hypothesis we have that $\mathsf{k}_u(R,t) \in \mathcal{H}_\mathsf{e}(K_\mathsf{solved})$ and we conclude by Proposition 5.

Case: SIMPLE CONSEQUENCE. There is a statement

$$f = \left(H' \Leftarrow B'_1, \ldots, B'_n\right) \in S$$

and a substitution σ grounding for f such that $\mathsf{skel}(f\sigma)$ is in normal form, $H = H'\sigma$ and $B'_i\sigma \in \mathcal{H}(S)$. By the induction hypothesis, we have that $B'_i\sigma \in \mathcal{H}_\mathsf{e}(K_\mathsf{solved})$. W.l.o.g. assume that $B'_i = \mathsf{k}_{w'_i}(X_i, t'_i)$. As $B'_i\sigma \in \mathcal{H}_\mathsf{e}(K_\mathsf{solved})$, we have by definition of $\mathcal{H}_\mathsf{e}$ that there exist R'_i such that

$$B''_i = \mathsf{k}_{w'_i\sigma}(R'_i, t'_i\sigma) \in \mathcal{H}(K_\mathsf{solved}), \tag{3}$$

$$\mathsf{i}_{w'_i\sigma}(R'_i, X_i\sigma) \in \mathcal{H}_\mathsf{e}(K_\mathsf{solved}) \tag{4}$$

for all $1 \leq i \leq n$ and let π''_i be a derivation tree in $\Pi'(K_\mathsf{solved})$ of B''_i.

But $w'_i\sigma$ is a prefix of w, where w is such that $H = \mathit{predicate}_w(\ldots)$ with $\mathit{predicate} \in \{\mathsf{r}, \mathsf{k}\}$. Note that as S is a set of *seed* statements, $\mathit{predicate} \notin \{\mathsf{i}, \mathsf{ri}\}$. By applying the EXTEND rule to Equation (4), we obtain

$$\mathsf{i}_w(R'_i, X_i\sigma) \in \mathcal{H}_\mathsf{e}(K_\mathsf{solved}). \tag{5}$$

Let σ' be the substitution defined to be σ except that it maps X_i to R'_i for all $1 \leq i \leq n$. We distinguish several cases depending on the statement f.

1) Case where f is f_0^+, f_1^+, f_2^+, f_3^+ or f_4^+. We conclude by applying Lemma 26 and rules of $\mathcal{H}_\mathsf{e}$.
2) Case where f is not f_1^+, f_2^+, f_3^+ or f_4^+.
   We will show that $H'\sigma' \in \mathcal{H}_\mathsf{e}(K_\mathsf{solved})$. As K was updated by f, there are three cases:
   a) if $f \in K$, we conclude by Lemma 21 or Lemma 17 (depending on the predicate).
   b) Otherwise, $f\Downarrow \in K_\mathsf{solved}$ or $f\Downarrow = \left(\mathsf{k}_w(R,t) \Leftarrow C_1, \ldots, C_m\right)$ and there exists R' such that
   $$\left(\mathsf{k}_{w_0}(R',t) \Leftarrow B_1, \ldots, B_n\right) \in \mathbf{conseq}(K_\mathsf{solved})$$
   and such that
   $$\left(\mathsf{i}_{w_0}(R\downarrow, R'\downarrow) \Leftarrow B_1, \ldots, B_n\right) \in K_\mathsf{solved}.$$
   In this case we conclude by Lemma 20.

We have shown that $H'\sigma' \in \mathcal{H}_\mathsf{e}(K_\mathsf{solved})$. We distinguish several cases depending on *predicate*:

- *predicate* = r: In such a case, we have that $H'\sigma' = H'\sigma = H$ and we easily conclude.
- *predicate* = k: In such a case, we have that $H'\sigma' = \mathsf{k}_w(R\sigma', t\sigma')$. Relying on Equation 5 and applying the rules CONG and REFL, we deduce that: $\mathsf{i}_w(R\sigma, R\sigma') \in \mathcal{H}_\mathsf{e}(K_\mathsf{solved})$. Since $\mathsf{k}_w(R\sigma', t\sigma') \in \mathcal{H}_\mathsf{e}(K_\mathsf{solved})$ and $t\sigma = t\sigma'$, using Proposition 4, we conclude that $\mathsf{k}_w(R\sigma, t\sigma) \in \mathcal{H}_\mathsf{e}(K_\mathsf{solved})$. □

APPENDIX
PROOF OF THE ALGORITHM

A. Proof of Lemma 1

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

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

Proof. First, we have that $K \uplus \mathsf{ext}(K)$ is a knowledge base. Second, we show that any rule triggered from $K \uplus \mathsf{ext}(K)$ 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 $\mathsf{ext}(K)$.

1) if the rule RESOLUTION is triggered, we have $f, g \in K$. First, no statement $\left(\mathsf{k}(m,m) \Leftarrow\right) \in \mathsf{ext}(K)$ can play the role of g in the RESOLUTION saturation rule since $t' = m$ must unify with $t \notin X$. Therefore t must be m, but $m \notin \mathit{names}(K)$ by hypothesis and therefore t cannot be m. Second, no statement in $\mathsf{ext}(K)$ 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 $\left(\mathsf{k}(m,m) \Leftarrow\right) \in \mathsf{ext}(K)$ plays the role of f 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 $m \notin \mathit{names}(K)$ by hypothesis it follows that $g \in \mathsf{ext}(K)$ and therefore $g = \mathsf{k}(m,m)$. Therefore the resulting statement is $\mathsf{i}(m,m) \in \mathsf{ext}(K)$.
   b) if a statement $\left(\mathsf{k}(m,m) \Leftarrow\right) \in \mathsf{ext}(K)$ 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 $\left(\mathsf{i}(m,m) \Leftarrow\right) \in \mathsf{ext}(K)$ plays the role of f, then $g = \mathsf{r}_u \Leftarrow B_1, \ldots, B_n \in K$ and the result is in $\mathsf{ext}(K)$.
   b) otherwise $f \in K$. The statement g must also be in K since g is a reachability statement and $\mathsf{ext}(K)$ does not contain reachability statements. Therefore the rule has already been applied. □

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, $\mathcal{N}^P_{\mathsf{pub}} \subseteq \mathcal{N}_{\mathsf{pub}}$ be the finite set of public names occurring in P. We have that:

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

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

- $\mathsf{k}_\epsilon(n, n) \Leftarrow$ for any $n \in \mathcal{N}_{\mathsf{pub}} \smallsetminus \mathcal{N}^P_{\mathsf{pub}}$;
- $\mathsf{i}_\epsilon(n, n) \Leftarrow$ for any $n \in \mathcal{N}_{\mathsf{pub}} \smallsetminus \mathcal{N}^P_{\mathsf{pub}}$;
- $\mathsf{ri}_u(n, n) \Leftarrow B_1, \ldots, B_n$ for any $\mathsf{r}_u \Leftarrow B_1, \ldots, B_n \in K$ in solved form, any $n \in \mathcal{N}_{\mathsf{pub}} \smallsetminus \mathcal{N}^P_{\mathsf{pub}}$.

Proof. Let $K_1 = \{\mathsf{k}_\epsilon(n, n) \Leftarrow \;\mid\; n \in \mathcal{N}_{\mathsf{pub}} \smallsetminus \mathcal{N}^P_{\mathsf{pub}}\}$. We have $K_{\mathsf{init}}(P) = K_{\mathsf{init}}(\mathsf{seed}(P, \mathcal{N}^P_{\mathsf{pub}})) \cup K_1$. We apply saturation and we trigger only saturation rules whose premisses are not in $\mathsf{ext}(K)$. We get:

$$K_{\mathsf{init}}(P) = K_{\mathsf{init}}(\mathsf{seed}(P, \mathcal{N}^P_{\mathsf{pub}})) \cup K_1 \Rightarrow^* K \cup K_1$$

where $K = \mathsf{sat}(K_{\mathsf{init}}(\mathsf{seed}(P, \mathcal{N}^P_{\mathsf{pub}})))$. Then we can show that $K \cup K_1 \Rightarrow^* K \cup \mathsf{ext}(K)$ (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 $K \cup \mathsf{ext}(K)$ is saturated. ☐

B. Correction of the algorithm

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

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

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

Proof. The seed knowledge base satisfies the above properties and they are preserved by 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 $c_i \notin names(P_0)$ for all $1 \le i \le k$. If

$$(P_0, \varphi_0) \xRightarrow{L_1} (P_1, \varphi_1) \xRightarrow{L_2} \ldots \xRightarrow{L_n} (P_n, \varphi_n)$$

and $\forall 1 \le i \le k$

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

then

$$(P_0, \varphi_0) \xRightarrow{L_1\theta'} (P_1\theta, \varphi_1\theta) \xRightarrow{L_2\theta'} \ldots \xRightarrow{L_n\theta'} (P_n\theta, \varphi_n\theta),$$

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

Proof. By induction on the length n of the derivation. ☐

Lemma 31: Let P be a ground process, $\{c_1, \ldots, c_k\}$ be public names not occurring in P, $\theta : \{c_1, \ldots, c_k\} \to \mathcal{T}(\Sigma, \mathcal{N})$, and $\theta' : \{c_1, \ldots, c_k\} \to \mathcal{T}(\Sigma, \mathcal{N}_{\mathsf{pub}} \cup \mathcal{W})$.

If $P \models \mathsf{r}_w$, $P \models \mathsf{k}_w(R, t)$, and $P \models \mathsf{k}_{w\theta}(c_i\theta', c_i\theta)$ then $P \models \mathsf{k}_{w\theta}(R\theta', t\theta)$.

Proof. Suppose that $P \models \mathsf{r}_{w\theta}$. Otherwise the conclusion trivially follows from the semantics of the k predicate. Let $w = \ell_1 \ldots \ell_n$.

As $P \models \mathsf{r}_w$, we have that $(P, \emptyset) \xRightarrow{L_1,\ldots,L_n} (Q, \varphi)$ such that for all $1 \le i \le n$ it holds that $L_i\varphi{\downarrow} =_{\mathsf{AC}} \ell_i{\downarrow}$.

As $P \models \mathsf{r}_{w\theta}$, we have that $(P, \emptyset) \xRightarrow{L'_1,\ldots,L'_n} (Q', \varphi')$ such that for all $1 \le i \le n$ it holds that $L'_i\varphi'{\downarrow} =_{\mathsf{AC}} \ell_i\theta{\downarrow}$.

By induction on n we can show that $\varphi' =_{\mathsf{AC}} \varphi\theta{\downarrow}$. Finally, we show by induction on R that $\varphi \vdash_R t$ and $\varphi\theta \vdash_{c_i\theta'} c_i\theta$ (for $1 \le i \le k$) implies that $\varphi\theta \vdash_{R\theta'} t\theta$. ☐

Lemma 32: Let P be a ground process and φ a frame such that $(P, \varphi) \xRightarrow{\mathbf{in}(d,R)} (P', \varphi')$. Let R' be such that $(R = R')\varphi$. Then we have that $(P, \varphi) \xRightarrow{\mathbf{in}(d,R')} (P', \varphi')$.

Proof. R and R' are recipes for the same term in φ and therefore the transition still holds. ☐

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

- if $P \sqsubseteq \mathcal{P}$ then REACH-IDENTITY$(K^0_{\mathsf{solved}}, \mathcal{P})$ holds;
- if $\mathcal{P}$ is determinate and REACH-IDENTITY$(K^0_{\mathsf{solved}}, \mathcal{P})$ holds then $P \sqsubseteq \mathcal{P}$.

Proof. Let $K \in \mathsf{sat}(K_{\mathsf{init}}(P))$. By Lemma 1, we have that $K_{\mathsf{solved}} = K^0_{\mathsf{solved}} \cup K_R$ for some K_R as defined in Lemma 1. We first prove that if REACH-IDENTITY fails then $P \not\sqsubseteq \mathcal{P}$. In such a case, we have that

$$\left(\mathsf{ri}_{l_1,\ldots,l_n}(R, R') \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{i\in\{1,\ldots,m\}}\right) \in K^0_{\mathsf{solved}}$$

1) either $(\mathcal{P}, \emptyset) \not\xRightarrow{M_1,\ldots,M_n} (P', \varphi)$ for any (P', φ). However, by Theorem 2 (soundness of K_{solved} and thus of K^0_{solved}), we have that there exists (P'', φ'') such that $(P, \emptyset) \xrightarrow{M_1,\ldots,M_n} (P'', \varphi'')$. Hence, we have that $P \not\sqsubseteq \mathcal{P}$.
2) or for any (P', φ) such that $(\mathcal{P}, \emptyset) \xRightarrow{M_1,\ldots,M_n} (P', \varphi)$ and any grounding substitution ω we have that $R\omega\varphi{\downarrow} \neq_{\mathsf{AC}} R'\omega\varphi{\downarrow}$. By Theorem 2, we have that there exists (P'', φ'') such that $(P, \emptyset) \xrightarrow{M_1,\ldots,M_n} (P'', \varphi'')$ and $R\omega\varphi''{\downarrow} =_{\mathsf{AC}} R'\omega\varphi''{\downarrow}$. Hence, we have that $P \not\sqsubseteq \mathcal{P}$.

Next, we prove that if $P \not\sqsubseteq \mathcal{P}$ and $\mathcal{P}$ determinate, then REACH-IDENTITY fails. We assume that $P \not\sqsubseteq \mathcal{P}$, and that REACH-IDENTITY holds, and we derive a contradiction. As $P \not\sqsubseteq \mathcal{P}$, it follows that there exist $L_1, \ldots, L_n$, and φ such that $(P, \emptyset) \xrightarrow{L_1,\ldots,L_n} (P', \varphi)$ and $(R =_{\mathsf{E}} R')\varphi$, and

1) either $(\mathcal{P}, \emptyset) \not\xRightarrow{L_1,\ldots,L_n} (Q', \psi)$ for any (Q', ψ);
2) or $(R \neq_{\mathsf{E}} R')\psi$ where ψ is the frame reached by $\mathcal{P}$ after the execution of $L_1, \ldots. L_n$.

We consider such a witness of minimal length. Note that, since $\mathcal{P}$ is determinate, even if such a frame ψ 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 $(P, \emptyset) \xrightarrow{L_1,\ldots,L_n} (P_n, \varphi_n) = (P', \varphi)$ and $(R =_{\mathsf{E}} R')\varphi_n$, by completeness, we have that:

$$\mathsf{i}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow}(R, R') \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

In the second case, we also have that $(R \neq_{\mathsf{E}} R')\psi$. From the fact that $\mathsf{i}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow}(R, R') \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$ and $(R \neq_{\mathsf{E}} R')\psi$, we can show that there exist recipes R_A, R'_A and $k \leq n$ such that $\mathsf{i}_{L_1\varphi_n\downarrow,\ldots,L_k\varphi_n\downarrow}(R_A, R'_A) \in \mathcal{H}(K_{\mathsf{solved}})$ but $(R_A \neq R'_A)\psi$. 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 $\hat{\mathcal{S}}_n(\pi)$ is minimal too. More formally, we consider a witness of non-inclusion such that $(n, \hat{\mathcal{S}}_n(\pi))$ is minimal (with a lexical order) where n is the length of the witness and π is the derivation tree in $\Pi'(K_{\mathsf{solved}})$ of $\mathsf{i}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow}(R_A, R'_A)$.

We denote by π_m the derivation tree of such a minimal witness $\mathsf{i}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow}(R_A, R'_A) \in \mathcal{H}(K_{\mathsf{solved}})$.

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 π_m has size $\hat{\mathcal{S}}_n(\pi_m) = 0$), and of course, we do not have that $(R_A \neq R'_A)\psi$. Now, considering such a minimal witness, we have that:

$$(P, \emptyset) \xrightarrow{L_1} (P_1, \varphi_1) \xrightarrow{L_2} \ldots (P_{n-1}, \varphi_{n-1}) \xrightarrow{L_n} (P_n, \varphi_n)$$

By Theorem 2 (completeness), we know that:

$$\mathsf{r}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow} \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}}).$$

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

- $\mathsf{r}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow} \in \mathcal{H}(K_{\mathsf{solved}})$, and
- $\mathsf{i}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow}(R_A, R'_A) \in \mathcal{H}(K_{\mathsf{solved}})$.

Thanks to Lemma 22, we deduce that:

$$\mathsf{ri}_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R_A, R'_A) \in \mathcal{H}_{\mathsf{e}}^{(n,\hat{\mathcal{S}}_n(\pi_m))}(K_{\mathsf{solved}}).$$

Therefore there exists a statement

$$f = \left(\mathsf{ri}_{l_1,\ldots,l_n}(R_B, R'_B) \Leftarrow \{\mathsf{k}_{w_i}(X_i, x_i)\}_{i\in\{1,\ldots,m\}}\right) \in K_{\mathsf{solved}}$$

and a substitution τ grounding for f such that:

- $\mathsf{k}_{w_i\tau}(X_i\tau, x_i\tau) \in \mathcal{H}(K_{\mathsf{solved}})$ of derivation tree π_i (for all $1 \leq i \leq m$),
- $l_1\tau, \ldots, l_n\tau = L_1\varphi_n\downarrow, \ldots, L_n\varphi_n\downarrow$, and
- $R_B\tau\downarrow = R_A\downarrow$, and $R'_B\tau\downarrow = R'_A\downarrow$.

Due to the shape of f (note that $(R_B, R'_B) \neq (n, n)$ for any public name n), we have that $f \in K^0_{\mathsf{solved}}$. We suppose w.l.o.g. that $i \leq j$ implies $w_i \sqsubseteq w_j$ for f.

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

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

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

be a bijection of inverse σ^{-1}. We have that:

- $\mathsf{k}(c_j, c_j) \Leftarrow\ \in\ K_{\mathsf{init}}(P)$ for all $1 \leq j \leq k$;
- $\mathsf{k}_{w_i\sigma}(X_i\sigma', x_i\sigma) \in \mathcal{H}(K_{\mathsf{solved}})$ for all $1 \leq i \leq m$;

where $dom(\sigma') = \{X_1, \ldots, X_m\}$ and $\sigma'(X_i) = x_i\sigma$ for all $1 \leq i \leq m$. We also define the mapping σ'^{-1} to be such that $dom(\sigma'^{-1}) = \{c_1, \ldots, c_k\}$ 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

$$\mathsf{ri}_{l_1\sigma,\ldots,l_n\sigma}(R_B\sigma', R'_B\sigma') \in \mathcal{H}(K_{\mathsf{solved}}).$$

By Theorem 2 (soundness), $P \models \mathsf{ri}_{l_1\sigma,\ldots,l_n\sigma}(R_B\sigma', R'_B\sigma')$. Therefore, there exist recipes R'_i (for all $1 \leq i \leq n$ such that $l_i = \mathbf{in}(d_i, t_i)$) such that $P \models \mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R'_i, t_i\sigma)$. By Theorem 2 (completeness) and definition of $\mathcal{H}_{\mathsf{e}}$ there exist recipes R_i such that $\mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(K_{\mathsf{solved}})$, and $\mathsf{i}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, R'_i) \in \mathcal{H}_{\mathsf{e}}(K_{\mathsf{solved}})$.

Let $M_i = l_i$ if $l_i \in \{\mathbf{test}, \mathbf{out}(c) \mid c \in \mathcal{C}h\}$ and let $M_i = \mathbf{in}(d_i, R''_i)$ if $l_i = \mathbf{in}(d_i, t_i)$ for all $1 \leq i \leq n$ where the recipes R''_i correspond to those computed during the algorithm. As REACH-IDENTITY$(K_{\mathsf{solved}}, \mathcal{P})$ holds there exists $Q'_0 \in \mathcal{P}$ such that

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

where $\psi'_0 = \emptyset$.

Let i be such that $l_i = \mathbf{in}(d_i, t_i)$. Applying Lemma 29 to f we have that for all $x \in vars(t_i)$ there exists w_j such that $|w_j| < i$ and $x = x_j$. We have that $\mathsf{k}_{w_j\tau}(X_j\tau, x_j\tau) \in \mathcal{H}(K_{\mathsf{solved}})$, and $\mathsf{k}_{w_j\tau}(X_j\beta\tau, x_j\tau) \in \mathcal{H}(K_{\mathsf{solved}})$ by choice of f, τ and β. By Theorem 2 (soundness), we obtain that $P \models \mathsf{k}_{w_j\tau}(X_j\beta\tau, x_j\tau)$. Hence, as $|w_j| < i$, $P \models \mathsf{k}_{l_1\tau,\ldots,l_{i-1}\tau}(X_j\beta\tau, x_j\tau)$. We rewrite it as $P \models \mathsf{k}_{l_1\sigma\sigma^{-1}\tau,\ldots,l_{i-1}\sigma(\sigma^{-1}\tau)}(X_j\beta\sigma'(\sigma'^{-1}\tau), x_j\sigma(\sigma^{-1}\tau))$. We already established that $P \models \mathsf{r}_{l_1\sigma,\ldots,l_n\sigma}$, and we know that $\mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i\sigma) \in \mathcal{H}(K_{\mathsf{solved}})$ and therefore by Theorem 2 (soundness) we have that $P \models \mathsf{k}_{l_1\sigma,\ldots,l_{i-1}\sigma}(R_i, t_i\sigma)$. We apply Lemma 31 to obtain that

$$P \models \mathsf{k}_{l_1\tau,\ldots,l_{i-1}\tau}(R_i\sigma'^{-1}\tau, t_i\tau). \tag{6}$$

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

- θ^0 is the identity function; and
- $\theta^j(c_i) = \sigma'^{-1}\tau(c_i)\psi'_{idx(c_i)-1}\theta^{j-1}$ where $1 \leq j \leq n$, $dom(\theta^j) = \{c_i \mid idx(c_i) \leq j\}$ and where $idx(c_i) = min\{k \mid c_i \in names(M_k)\}$.

By applying Lemma 30 inductively on n we have that

$$(Q'_0, \psi'_0) = (Q'_0\theta, \psi_0\theta) \xRightarrow{M_1\sigma'^{-1}\tau} (Q'_1\theta, \psi'_1\theta) \xRightarrow{M_2\sigma'^{-1}\tau} \ldots \xRightarrow{M_n\sigma'^{-1}\tau} (Q'_n\theta, \psi'_n\theta).$$

Now we show by induction on n that

$$(Q'_0\theta, \psi_0\theta) \xRightarrow{L_1} (Q'_1\theta, \psi'_1\theta) \xRightarrow{L_2} \ldots \xRightarrow{L_n} (Q_n\theta, \psi'_n\theta).$$

We assume by the induction hypothesis that

$$(Q'_0\theta, \psi_0\theta) \overset{L_1}{\Longrightarrow} (Q'_1\theta, \psi'_1\theta) \overset{L_2}{\Longrightarrow} \ldots \overset{L_{i-1}}{\Longrightarrow} (Q'_{i-1}\theta, \psi'_{i-1}\theta)$$

and we show that

$$(Q'_{i-1}\theta, \psi'_{i-1}\theta) \overset{L_i}{\Longrightarrow} (Q'_i\theta, \psi'_i\theta).$$

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 $L_i = \mathbf{in}(d_i, R''_i)$ and $M_i\sigma'^{-1}\tau = \mathbf{in}(d_i, R_i\sigma'^{-1}\tau)$ (in the case of a **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 $\models$, we have that $P \models \mathsf{k}_{l_1\tau,\ldots,l_{i-1}\tau}(R''_i, t_i\tau)$. We have previously shown in (6) that $P \models \mathsf{k}_{l_1\tau,\ldots,l_{i-1}\tau}(R_i\sigma'^{-1}\tau, t_i\tau)$ and therefore $P \models \mathsf{i}_{l_1\tau,\ldots,l_{i-1}\tau}(R_i\sigma'^{-1}\tau, R''_i)$, or, equivalently, $(R_i\sigma'^{-1}\tau = R''_i)\varphi_{i-1}$. By the hypothesis, we have that there exists $Q \in \mathcal{P}$ such that

$$(Q, \emptyset) \overset{L_1}{\Longrightarrow} (Q_1, \psi_1) \overset{L_2}{\Longrightarrow} \ldots \overset{L_{i-1}}{\Longrightarrow} (Q_{i-1}, \psi_{i-1}),$$

and $(R_i\sigma'^{-1}\tau = R''_i)\psi_{i-1}$. By determinacy of Q it follows that $\psi_{i-1} \approx_s \psi'_{i-1}\theta$ 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 $(\mathcal{P}, \emptyset) \overset{L_1,\ldots,L_n}{\Longrightarrow} (Q'_n\theta, \psi'_n\theta)$, 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 $(R_B\sigma' =_{\mathsf{E}} R'_B\sigma')\psi'_n$, we deduce that $[(R_B\sigma'\sigma'^{-1}\tau =_{\mathsf{E}} R'_B\sigma'\sigma'^{-1}\tau)\psi'_n]\theta$.

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 is 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 $c_j = \sigma(x_i)$.

By construction of θ, $c_j\theta = \tau \circ \sigma'^{-1}(c_j)(\psi'_n\theta)$ 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,

$$(R_B\beta\tau)(\psi'_n\theta) =_{\mathsf{E}} (R'_B\beta\tau)(\psi'_n\theta).$$

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

We now show that:

$$(R_B\beta\tau =_{\mathsf{E}} R'_B\beta\tau)\psi \text{ implies } (R_B\tau =_{\mathsf{E}} R'_B\tau)\psi.$$

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

1) Case: $\mathcal{W}(\pi_i) < n$ and $\mathcal{W}(\pi_j) < n$. In such a case, from Corollary 1, we have $\mathsf{i}_{L_1\varphi_n\downarrow,\ldots,L_k\varphi_n\downarrow}(X_i\tau, X_j\tau) \in \mathcal{H}_{\mathsf{e}}^{(k,\kappa)}(K_{\mathsf{solved}})$ for some $k < n$ and κ. Hence we have $(X_i\tau =_{\mathsf{E}} X_j\tau)\varphi_k$ By minimality of n we also have that $(X_i\tau =_{\mathsf{E}} X_j\tau)\psi$.
2) Case: $\mathcal{W}(\pi_i) = n$ and $\mathcal{W}(\pi_j) = n$. In such a case, from Corollary 1, we have $\mathsf{i}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow}(X_i\tau, X_j\tau) \in \mathcal{H}_{\mathsf{e}}^{(n,\kappa')}(K_{\mathsf{solved}})$ with $\kappa' \leq \hat{\mathcal{S}}_n(\pi_i) + \hat{\mathcal{S}}_n(\pi_j)$. Hence we have $(X_i\tau =_{\mathsf{E}} X_j\tau)\varphi$. Since $\hat{\mathcal{S}}_n(\pi_i) \leq \mathcal{S}(\pi_i) - 1$ (the root node is not counted) and $\mathcal{S}(\pi_i) + \mathcal{S}(\pi_j) \leq \hat{\mathcal{S}}_n(\pi_m)$, we have $\kappa' < \hat{\mathcal{S}}_n(\pi_m)$. By minimality of our witness, we also have that $(X_i\tau =_{\mathsf{E}} X_j\tau)\psi$.
3) Case $\mathcal{W}(\pi_i) = n$ and $\mathcal{W}(\pi_j) < n$. In such a case, from Corollary 1, we have $\mathsf{i}_{L_1\varphi_n\downarrow,\ldots,L_n\varphi_n\downarrow}(X_i\tau, X_j\tau) \in \mathcal{H}_{\mathsf{e}}^{(n,\kappa')}(K_{\mathsf{solved}})$ with $\kappa' \leq \hat{\mathcal{S}}_n(\pi_i)$ since $\hat{\mathcal{S}}_n(\pi_j) = 0$. Hence we have $(X_i\tau =_{\mathsf{E}} X_j\tau)\varphi$. Since $\hat{\mathcal{S}}_n(\pi_i) \leq \mathcal{S}(\pi_i) - 1$ (the root node is not counted) and $\mathcal{S}(\pi_i) \leq \hat{\mathcal{S}}_n(\pi_m)$, we have $\kappa' < \hat{\mathcal{S}}_n(\pi_m)$. By minimality of our witness, we also have that $(X_i\tau =_{\mathsf{E}} X_j\tau)\psi$.

From the above equalities, we have that

- $(R_B\tau =_{\mathsf{E}} R_B\beta\tau)\psi$,
- $(R'_B\tau =_{\mathsf{E}} R'_B\beta\tau)\psi$,

Therefore, we get $(R_B\tau =_{\mathsf{E}} R'_B\tau)\psi$ and hence $(R_A =_{\mathsf{E}} R'_A)\psi$, thus obtaining a contradiction.
As both cases yield a contradiction, it follows that if $P \not\sqsubseteq \mathcal{P}$ then REACH-IDENTITY($K^0_{\mathsf{solved}}, \mathcal{P}$) fails. $\square$