Verification of Security Protocols with a Bounded Number of Sessions based on Resolution for Rigid Variables

Reynald Affeldt and Hubert Comon-Lundh

Research Center for Information Security (RCIS),
National Institute of Advanced Industrial Science and Technology (AIST)
{reynald.affeldt,h.comon-lundh}@aist.go.jp
http://www.rcis.aist.go.jp

Abstract. First-order logic resolution is a standard way to automate the verification of security protocols. However, it sometimes fails to produce security proofs for secure protocols because of the detection of false attacks. For the verification of a bounded number of sessions, false attacks can be avoided by introducing rigid variables. Unfortunately, this yields complicated resolution procedures. We show here that there is a simple translation of the security problem for a bounded number of sessions into first-order logic, that does not introduce false attacks. This is shown by translating clauses involving rigid variables into classical first-order clauses, while preserving satisfiability. We illustrate this approach by giving a complete and terminating strategy for a first-order logic fragment resulting from the above translation, that yields a decision procedure for a bounded number of sessions.

1 Introduction

It is convenient and simple to model the security of cryptographic protocols within first-order logic. It is indeed possible then to use general purpose theorem provers such as SPASS (see first experiments for security protocols in [16]). There are also successful verification tools such as ProVerif [4], which are based on first-order logic. However, such a formalization requires some approximations. First, global properties such as freshness require a heavy encoding to be faithfully represented in first-order logic (see e.g., [9]), which is not amenable to further automation.

Second, pieces of messages that can be replaced with any message (since they cannot be analyzed by the recipient) are abstracted by variables. Such variables are naturally universally quantified in first-order logic. However, if an attacker can indeed replace these messages with an arbitrary forged message (hence a universal quantification), he should be allowed to do it only once for every variable: the attacker can choose the substitution, but has to commit on this value. On the other hand, in a first-order logic formulation, since $\forall x.\phi(x)$ is equivalent to $(\forall x.\phi(x)) \wedge (\forall y.\phi(y))$, the attacker may use two distinct substitutions for the

same variable. Hence, in general, the attacker model in first-order logic corresponds to a stronger attacker than the real one. We will give concrete examples in Sect. 3.

It follows that a first-order logic formulation may yield false attacks. This has been well-known for a long time and it is the reason why several more accurate formalisms have been designed, for instance using MSR or linear logic [6].

Instead of proving security for an arbitrary number of sessions, much work focuses on finding an attack for a bounded number of sessions (e.g., [14]). In this setting, there is no need for approximation: the insecurity problem can be translated into deducibility constraints, after guessing an interleaving of actions. Translating this approach in first-order logic is not straightforward, for the same reason as above: we must express that a variable, though universally quantified, can be instantiated only once.

A simple way to fix the problems and remove the false attacks due to universal quantification is to use *rigid variables*: while universally quantified variables can be instantiated as many times as we wish, rigid variables get only one instance [2]. This is exactly what we need. We need however one set of rigid variables for each session. Hence this is only relevant to bounded number of sessions. This is exactly what is done in [12]: the authors introduce rigid clauses to model the protocol rules when the number of sessions is fixed. Then they design a proof calculus for such clauses and show a termination result.

To the best of our knowledge, there is no formulation of the security problem for a bounded number of sessions within first-order logic, that avoids false attacks. This is what we do in the present paper. Actually, we show that there is a simple translation of rigid variables into first-order logic that preserves the satisfiability of formulas. It follows that we can capture rigid-validity, hence security for a bounded number of sessions, within first-order logic. This result has many interesting applications. First, on the practical side, it makes it possible to use first-order theorem provers for finding attacks in a bounded number of sessions, without generating false attacks. This can be useful, when trying to reconstruct attacks from candidate attacks found by a theorem prover: this is an alternative to [1]. It makes it also possible to search, with the same tool, for attacks and proofs. This approach is also appealing when compared with the alternative constraint solving techniques, because we do not need to guess an interleaving of actions.

Using this simple translation of rigid variables in first-order logic, we may considerably simplify the proof rules for rigid clauses of [12]. We can also extend the calculus, allowing for clauses mixing rigid and non-rigid variables as well as equalities (however only flexible variables are allowed in equalities).

Finally, we can translate back and forth results from first-order logic to first-order logic with rigid variables. For instance, from the decision results on security protocols for a bounded number of sessions, we can derive decision results (in co-NP) for fragments of first-order logic. We illustrate this by giving a resolution strategy, which we prove to be complete and terminating for a certain class of clauses; as a corollary, we derive the decidability of the probem of security for

a bounded number of sessions in the classical Dolev-Yao model. As we anticipated, the simplicity of the corresponding decision procedure makes software implementation an easier task: it took us only little time to implement it to confirm our results by experiments.

We recall the basics about models of security protocols in first-order logic in Sect. 3, using examples. In Sect. 4, we show the simple translation from rigid clauses to first-order clauses. In Sect. 5, we sketch some possible applications, including a new class of clauses for which resolution is a decision procedure.

2 Notations

We use the notations of [5], some of which are recalled below. $\mathcal{X} = \{x, y, z, v, \ldots\}$ is a set of variables symbols. $\mathcal{F}$ is a set of function symbols, each with a given arity. $\mathcal{T}(\mathcal{F}, \mathcal{X}) = \{s, t, u, \ldots\}$ is the set of terms built on $\mathcal{F}$ and $\mathcal{X}$. $Var\,(t)$ is the set of variables occurring in t. t is ground when $Var\,(t) = \emptyset$. $t|_p$ is the subterm of t at position p. $\bar{t}$ denotes the vector $t_1, t_2, \ldots$ Set-theoretic notations $(\cap, \cup, \uplus, \subseteq)$ are also used for vectors.

A substitution σ is a mapping from variables to terms, which is the identity, except on a finite set called its domain, noted $Dom\,(\sigma)$. A substitution mapping $x_1, x_2, \ldots$ to $t_1, t_2, \ldots$ is written $\{x_1 \mapsto t_1; x_2 \mapsto t_2; \cdots\}$. Substitutions are homomorphically extended to $\mathcal{T}(\mathcal{F}, \mathcal{X})$ and used in postfix notation. The variable range of the substitution σ is defined as $VRange\,(\sigma) = \bigcup_{x \in Dom(\sigma)} Var\,(x\sigma)$ and its set of variables as $Var\,(\sigma) = Dom\,(\sigma) \cup VRange\,(\sigma)$. $Mgu(t =^? u)$ is a most general unifier of terms t, u (or most general solution of the equation $t =^? u$). We assume w.l.o.g. that $Var\left(Mgu(t =^? u)\right) \subseteq Var\,(t) \cup Var\,(u)$ and that $VRange\left(Mgu(t =^? u)\right) \cap Dom\left(Mgu(t =^? u)\right) = \emptyset$, i.e., we deal with idempotent unifiers.

$\mathcal{P} = \{P, Q, I, \ldots\}$ is a set of predicate symbols, each with a given arity. A predicate P of arity n applied to terms $t_1, \ldots, t_n$ is an atom; atoms are written $A, B, \ldots$ A literal L is an atom A or its negation $\neg A$. Clauses (ranged over by $C, \ldots$) are finite sets of literals and are noted $\bigvee_i L_i$. All variables in a clause are implicitly universally quantified (sometimes we make the quantifiers explicit). A Horn clause is a clause that contains at most one positive literal; it is noted $A_1, \ldots, A_n \rightarrow A$, or $A_1, \ldots, A_n \rightarrow$ when there is no positive literal.

A $\mathcal{F}$-algebra $\mathcal{A}$ is a set $D_\mathcal{A}$ with for each $f \in \mathcal{F}$ of arity n, a function $f_\mathcal{A} : D_\mathcal{A}^n \rightarrow D_\mathcal{A}$. $[\![t]\!]^{\sigma,\mathcal{A}}$ is the interpretation of a term t w.r.t. an assignment $\sigma : Dom\,(Var\,(t)) \rightarrow D_\mathcal{A}$. An $\mathcal{F}, \mathcal{P}$-structure $\mathcal{S}$ is a $\mathcal{F}$-algebra $\mathcal{A}$ together with an interpretation $[\![P]\!]^\mathcal{S} \subseteq D_\mathcal{A}^n$ of each n-ary predicate symbol. Given an $\mathcal{F}, \mathcal{P}$-structure $\mathcal{S}$ and an assignment σ of the free variables of ϕ into the interpretation domain of $\mathcal{S}$, $\mathcal{S}, \sigma \models \phi$ is the usual first-order satisfaction relation.

Binary resolution is an inference rule on clauses: $\dfrac{C \vee A \quad C' \vee \neg A'}{C\sigma \vee C'\sigma}$, where $\sigma = Mgu(A =^? A')$. We succinctly note $\cdot \dashv \cdot, \cdot$ for resolution steps. For any set of clauses S, we write $C \dashv^* S$ when C is derivable from clauses in S and we let S^* be $\{C \mid C \dashv^* S\}$.

3 Models of Security Protocols in First-order Logic

We give the main ideas about models of security protocols in first-order logic (see e.g., [4] for further details) and examples to illustrate false attacks.

3.1 A Standard Attacker Model

Typically, the set of function symbols contain $\{\cdot\}_\cdot$ and $[\cdot]_\cdot$, for respectively public-key and symmetric key encryption, the pairing function $\langle\cdot,\cdot\rangle$ and the function symbol pk, that builds a public key out of a private one. We will sometimes omit the pairing symbol or consider it as variadic to ease reading.

I is a predicate symbol that captures the intruder's knowledge. The attacker capabilities are typically described by the following set of Horn clauses:

$$
\begin{array}{llll}
(I0) & I(x), I(y) \to I(\langle x, y\rangle) & (I1) & I(\langle x, y\rangle) \to I(x) \\
(I2) & I(x), I(y) \to I([x]_y) & (I3) & I([x]_y), I(y) \to I(x) \\
(I4) & \to I(pk(x)) & (I5) & I(x), I(pk(y)) \to I\left(\{x\}_{pk(y)}\right) \\
(I6) & I\left(\{x\}_{pk(y)}\right), I(y) \to I(x) & &
\end{array}
$$

3.2 An Example of False Attack

Let us consider a protocol from [8]. We first write it using the (sometimes ambiguous) Alice-and-Bob notation:

$$
\begin{array}{lcl}
A \to B & : & \{pk_A, N\}_{pk_B} \\
B \to A & : & \{N, K\}_{pk_A} \\
A \to B & : & [S]_K \\
B \to A & : & N
\end{array}
$$

In the first phase, two agents A and B use their public keys pk_A and pk_B to exchange a new, symmetric key K (together with a nonce N). Later, A uses the key K to send a secret S to B. Eventually (and this is the peculiarity of this protocol), B reveals the nonce N. The security property states that any secret S generated by an honest agent A for an honest agent B is never disclosed.

Using first-order logic (as in ProVerif), the protocol is modeled as an oracle, that can be used by the intruder to get more information: for each rule, if the intruder can construct a message matching the expected pattern, then it gets the corresponding reply message:

$$
\begin{array}{ll}
(\Delta 1) & I(pk(x)), I(pk(y)) \to I\left(\{pk(x), N(x,y)\}_{pk(y)}\right) \\
(\Delta 2) & I\left(\{x, y\}_{pk(z)}\right) \to I\left(\{y, K(x,y,z)\}_x\right) \\
(\Delta 3) & I\left(\{N(x,y), z\}_{pk(x)}\right) \to I\left([S(x,y,z)]_z\right) \\
(\Delta 4) & I\left([v]_{K(x,y,z)}\right) \to I(y)
\end{array}
$$

For instance, the clause ($\Delta 2$) represents the first action of agent B: upon reception from A of $\{x, y\}_{pk(z)}$ (expected to be $\{pk(sk_A), N(sk_A, sk_B)\}_{pk(sk_B)}$) the reply of B is the message $\{y, K(x,y,z)\}_x$.

The initial intruder knowledge is modeled by (positive) atoms. For instance, if C is a corrupted agent, then there is a clause $I(sk_C)$. The security property can be modeled as $\neg I(S(sk_A, sk_B, z))$. If the protocol is insecure, then the set of clauses is unsatisfiable: there is a derivation of $I(S(sk_A, sk_B, t))$ for some t.

In the above clauses, the freshness of N, K, and S is approximated using a function symbol, which depends on the terms seen at this stage. This may be a cause of false attacks as, for instance, every session between A and B will use the same representation of N. For a bounded number of sessions, this problem does not occur as different symbols can be used for nonces occurring in different sessions.

There are however other sources of false attacks. In the above example, the protocol is (supposedly) secure, while there is a simple derivation of the empty clause: from a honest session of the protocol (i.e., using clauses ($\Delta 1$) to ($\Delta 4$) once), we derive $I(N(x, y))$. Now, for any z such that $I(z)$ we derive $I(\{N(x, y), z\}_{pk(x)})$ using the intruder capabilities. Next, using clause ($\Delta 3$) we get $I([S(x, y, z)]_z)$ and, from this clause and $I(z)$, we derive $I(S(x, y, z))$.

The problem here is that the nonce is first kept secret but eventually revealed. A first-order model leads to a false attack by wrongly inferring that the intruder could have the nonce at an early stage: when the nonce N is revealed, the rule ($\Delta 3$) is replayed and the intruder gets back $[S]_{K'}$ for a key K' of his choice, which he can decrypt. This would not occur in a more accurate model, where the agents would have moved forward their internal state, preventing the replay of rule ($\Delta 3$). This kind of problem occurs even for a single session, as shown by our example.

3.3 Trying to Refine the Model: there are still False Attacks

The false attack above comes from the ability (in the model) to play again a rule of the protocol after completing it. One may think that this can be fixed by adding some state information at each step of the protocol. While this is quite difficult for an unbounded number of sessions, there is an easy (though expensive) encoding for a bounded number of sessions.

First, we get rid of the freshness encoding by modeling nonces with distinct constants. Then, we guess an interleaving of actions (this is expensive and this is something that we can avoid) and use a different predicate symbol at each step: instead of a single I, we use $I_0, \ldots, I_n$ to represent the intruder knowledge after n steps. The protocol clauses increase the index of this predicate:

$$I_k(t) \rightarrow I_{k+1}(u) \qquad \text{for } k = 0, \ldots, n-1$$

We also add clauses $I_k(x) \rightarrow I_{k+1}(x)$ for $k = 0, \ldots, n-1$, that express the increasingness of the intruder knowledge. Finally, clauses reflecting intruder capabilities are replicated n times with the indices $0, \ldots, n$. As for the security property, it is stated as $\neg I_n(S)$ where S is the supposed secret.

With such an encoding, the above false attack can be prevented. However, this is not sufficient in general. Here is an (cook-up) example showing again a false attack in this new setting.

Example 1. Relying on the long-term shared secret K_{AB}, A wants to establish a short-term secret with B. B generates two nonces N_1, N_2 and sends them separately. A acknowledges both nonces by sending back one of them. The short-term secret is $N_1 \oplus N_2$.

$$\begin{array}{lcl} A \to B & : & [A, N_0]_{K_{AB}} \\ B \to A & : & [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\ A \to B & : & N_1 \end{array}$$

In a single-session model, there is no attack: the intruder can get either N_1 or N_2, but not both. However, in a clausal formulation we get the two clauses:

$$\begin{array}{rcl} I_1\left([A, x]_{K_{AB}}\right) & \to & I_2\left([B, x, N_1]_{K_{AB}}, [B, x, N_2]_{K_{AB}}\right) \\ I_2\left([B, N_0, x]_{K_{AB}}, [B, N_0, y]_{K_{AB}}\right) & \to & I_3\left(x\right) \end{array}$$

From $I_2\left([B, x, N_1]_{K_{AB}}, [B, x, N_2]_{K_{AB}}\right)$, by swapping pair projections, we infer $I_2\left([B, x, N_2]_{K_{AB}}, [B, x, N_1]_{K_{AB}}\right)$. Then using two instances of the second clause, we get immediately $I_3\left(N_1\right)$ and $I_3\left(N_2\right)$, hence the secret. This is a false attack: the last rule should not be played twice.

4 Rigid Clauses vs. Classical Clauses

The best way to prevent the last false attack is to use rigid variables and rigid clauses. We introduce these notions first, before showing how to get rid of them.

4.1 Rigid Clauses

Variables are either *rigid* (written in upper-case) or *flexible* (written in lower-case). Both types of variables are universally quantified, but rigid variables can only yet one instance. Before a formal definition, let us give some examples.

Example 2. Consider the following set of clauses (taken from [12]):

$$\left\{I\left(a\right),\ I\left(b\right),\ \neg I\left(X\right) \vee I\left(f(X)\right),\ \neg I\left(f(a)\right) \vee \neg I\left(f(b)\right)\right\}$$

If X was an ordinary first-order variable, this set of clauses would be unsatisfiable: from the three first clauses we can infer both $I\left(f(a)\right)$ and $I\left(f(b)\right)$. We need however two instances of the third clause, which is forbidden for rigid variables. We can choose the instance $X = a$ or the instance $X = b$, but not both.

The above set of clauses is satisfiable in our intended interpretation of rigid variables since the two following sets of ground clauses are satisfiable:

$$\left\{I\left(a\right), I\left(b\right), \neg I\left(a\right) \vee I\left(f(a)\right), \neg I\left(f(a)\right) \vee \neg I\left(f(b)\right)\right\} \quad \text{and}$$

$$\left\{I\left(a\right), I\left(b\right), \neg I\left(b\right) \vee I\left(f(b)\right), \neg I\left(f(a)\right) \vee \neg I\left(f(b)\right)\right\}$$

The next example shows that resolution procedures cannot be easily extended to clauses containing rigid variables.

Example 3. Consider the following set of clauses:

$$\{I(X),\ \neg I(f(x)) \vee I_0,\ \neg I(g(x))\}$$

It is unsatisfiable: the first and the third clauses resolve to the empty clause.

However, assume that we start by resolving the two first clauses. This yields the new set of clauses $\{I(f(Y)),\ I_0,\ \neg I(f(x)) \vee I_0,\ \neg I(g(x))\}$ where Y is a new rigid variable resulting from the unification $X =^? f(x)$. We can still choose Y, but we committed to an assignment of X to a term headed with f. Now the set of clauses is satisfiable. For a complete resolution procedure, we would have to restart from the beginning, with another choice of clauses to resolve.

This example shows that, unlike classical first-order clauses, resolution does not yield a logically equivalent set of clauses. Therefore, resolution theorem proving has to be reconsidered; this is the reason for complications in [12].

Let us now formalize the model theory of clauses with rigid variables.

Definition 1. *Let $\mathcal{C}$ be a set of clauses whose variables are split into $\overline{X}$ (rigid variables) and $\overline{y}$ (flexible variables).*

$\mathcal{C}$ is satisfiable *if there is an $\mathcal{F}$-algebra $\mathcal{A}$ such that, for any $\mathcal{A}$-assignment σ of $\overline{X}$, there is a structure $\mathcal{S}$ whose underlying algebra is $\mathcal{A}$ such that $\mathcal{S}, \sigma \models \forall \overline{y}.\mathcal{C}$.*

In other words, models of formulas with rigid variables are collections of structures, one for each assignment of the rigid variables.

Example 4. In Example 2, for any of the two assignments of X, there is a model: for the assignment $\{X \mapsto a\}$, $\{I(a), I(b), I(f(a)), \neg I(f(b))\}$, and for the assignment $\{X \mapsto b\}$, $\{I(a), I(b), I(f(b)), \neg I(f(a))\}$.

Example 5. The one session case of Example 1 can be translated into the following rigid clauses (keeping the intruder rules with flexible variables)

$$\begin{array}{rcl} & \rightarrow & I([A, N_0]_{K_{AB}}) \\ I([A, X]_{K_{AB}}) & \rightarrow & I(\langle [B, X, N_1]_{K_{AB}}, [B, X, N_2]_{K_{AB}} \rangle) \\ I(\langle [B, N_0, Y]_{K_{AB}}, [B, N_0, Z]_{K_{AB}} \rangle) & \rightarrow & I(Y) \end{array}$$

which, together with $\neg I(\langle N_1, N_2 \rangle)$ is satisfiable. In contrast, if the above variables are considered as flexible, it is unsatisfiable (yielding a false attack).

Example 6. There are also some traps. For instance, $\forall x.\phi(x) \wedge \psi(x) \models\!\models \forall x, y.\phi(x) \wedge \psi(y)$ while $\forall X.\phi(X) \wedge \psi(X) \not\models \forall X, Y.\phi(X) \wedge \psi(Y)$. Indeed, consider $\phi(X) = \psi(X) = P(X) \wedge (\neg P(a) \vee \neg P(b))$. $\forall X.\phi(X) \wedge \psi(X)$ is satisfiable: consider the algebra with two constants a and b. For the assignment $\{X \mapsto a\}$ (resp. $\{X \mapsto b\}$), the structure $\mathcal{S}$ such that $P(a)$ holds (resp. $P(b)$ holds) satisfies $\phi(a) \wedge \psi(a)$ (resp. $\phi(b) \wedge \psi(b)$). On the other hand, $\forall X, Y.\phi(X) \wedge \psi(Y)$ is not satisfiable, since, for the assignment $\{X \mapsto a; Y \mapsto b\}$, there is no structure that satisfies $\phi(a) \wedge \psi(b)$.

So, as we illustrated, rigid variables model exactly the intruder ability to use a protocol rule: (s)he may replace the variables by any value of his (her) choice, but (s)he has to commit to this value.This is the reason for studying rigid clauses and their satisfiability in [12]. However, as shown in the above examples, the resolution procedure involves a lot of complications and cannot be implemented easily. We now show how to circumvent these problems.

4.2 Translation of Rigid Clauses into First-order Clauses

As can be seen from the definition of satisfiability, the interpretation of predicates depends on the assignment of rigid variables. Then, a simple Skolemization argument suffices to eliminate this dependence and brings back first-order clauses:

Theorem 1. *There is an algorithm that, given a finite set of clauses $\mathcal{C}$ computes a finite set of clauses $\mathcal{C}'$, which does not contain any rigid variable, and such that $\mathcal{C}$ is satisfiable iff $\mathcal{C}'$ is satisfiable.*

Proof. $\mathcal{C}'$ is constructed from $\mathcal{C}$ as follows. Let $X_1, \ldots, X_n$ be the rigid variables of $\mathcal{C}$. For each $P \in \mathcal{P}$ of arity m, let P' be a predicate symbol of arity $n+m$. If $\neg P_1(\overline{s_1}) \vee \cdots \vee \neg P_{n_1}(\overline{s_{n_1}}) \vee Q_1(\overline{t_1}) \vee \cdots \vee Q_{n_2}(\overline{t_{n_2}})$ is a clause $C \in \mathcal{C}$, let

$$\neg P'_1(x_1, \ldots, x_n, \overline{s'_1}) \vee \cdots \vee \neg P'_{n_1}(x_1, \ldots, x_n, \overline{s'_{n_1}}) \vee Q'_1(x_1, \ldots, x_n, \overline{t'_1}) \vee \cdots \vee Q'_{n_2}(x_1, \ldots, x_n, \overline{t'_{n_2}})$$

be a clause $C' \in \mathcal{C}'$ where $x_1, \ldots, x_n$ are distinct variables, which do not occur free in the clause C and $s'_1, \ldots, s'_{n_1}, t'_1, \ldots, t'_{n_2}$ are the terms obtained from their unprimed version by replacing each X_i with the corresponding x_i.

If the set of clauses $\mathcal{C}$ is satisfiable, then there is an $\mathcal{F}$-algebra $\mathcal{A}$ such that, for any $\mathcal{A}$-assignment σ of $X_1, \ldots, X_n$ there is a structure $\mathcal{S}_\sigma$ such that, for every clause $C \in \mathcal{C}$, we have $\mathcal{S}_\sigma, \sigma \models \forall \overline{y}.C$. Consider then the structure $\mathcal{S}'$ (whose underlying algebra is $\mathcal{A}$) such that

$$(a_1, \ldots, a_n, b_1, \ldots, b_m) \in [\![P']\!]^{\mathcal{S}'} \text{ iff } (b_1, \ldots, b_m) \in [\![P]\!]^{\mathcal{S}_{\{X_1 \mapsto a_1; \ldots; X_n \mapsto a_n\}}}.$$

For any clause $C \in \mathcal{C}$, we claim that $\mathcal{S}' \models \forall \overline{x}, \overline{y}.C'$. For any assignment σ' of the variables $x_1, \ldots, x_n$ and for any assignment θ of the other variables $\overline{y}$ of the clause, we let σ be the assignment of the rigid variables defined by $\sigma(X_i) = \sigma'(x_i)$ for every i. By hypothesis, $\mathcal{S}_\sigma, \sigma, \theta \models C$. It follows that, for some literal $L \in C$, $\mathcal{S}_\sigma, \sigma, \theta \models L$. Assume for instance that L is a positive literal (the other case is similar): $L = P(u_1, \ldots, u_m)$ and $([\![u_1]\!]^{\sigma,\theta,\mathcal{A}}, \ldots, [\![u_m]\!]^{\sigma,\theta,\mathcal{A}}) \in [\![P]\!]^{\mathcal{S}_\sigma}$. This is equivalent, by definition, to

$$(a_1, \ldots, a_n, [\![u_1]\!]^{\sigma,\theta,\mathcal{A}}, \ldots, [\![u_m]\!]^{\sigma,\theta,\mathcal{A}}) \in [\![P']\!]^{\mathcal{S}'}$$

which, again by construction, yields $\mathcal{S}', \sigma', \theta \models C'$.

Conversely, if $\mathcal{C}'$ is satisfiable, then let $\mathcal{S}'$ be a structure which satisfies all clauses of $\mathcal{C}'$. Consider an arbitrary assignment σ of rigid variables occurring in $\mathcal{C}$. Let $\mathcal{S}_\sigma$ be the structure defined by

$$[\![P]\!]^{\mathcal{S}_\sigma} = \left\{ (b_1, \ldots, b_m) \mid (X_1\sigma, \ldots, X_n\sigma, b_1, \ldots, b_m) \in [\![P']\!]^{\mathcal{S}'} \right\}.$$

As before, $\mathcal{S}_\sigma, \sigma \models \forall \overline{y}.C$ iff $\mathcal{S}' \models \forall \overline{x}, \overline{y}.C'$. □

This extends to clauses with equality, provided that every equality clause does not contain any rigid variable.

5 Examples of Possible Applications

5.1 Automatic Proofs for a Bounded Number of Sessions

Thanks to the effective procedure given in the proof of Theorem 1, we can use resolution for mechanizing proofs in a bounded number of sessions. It works as well for clauses mixing rigid and flexible variables and also if we have (flexible) equations (though, in the latter case, there is no guarantee for termination).

Example 7. Let us come back to Example 5. We translate now the rigid clauses into first-order clauses:

$$\begin{aligned} &\rightarrow I(x,y,z,[A,N_0]_{K_{AB}}) \\ I(x,y,z,[A,x]_{K_{AB}}) &\rightarrow I(x,y,z,\langle [B,x,N_1]_{K_{AB}}, [B,x,N_2]_{K_{AB}}\rangle) \\ I(x,y,z,\langle [B,N_0,y]_{K_{AB}}, [B,N_0,z]_{K_{AB}}\rangle) &\rightarrow I(x,y,z,y) \\ I(x,y,z,\langle N_1,N_2\rangle) &\rightarrow \end{aligned}$$

Using an appropriate strategy (see next section), resolution terminates in a few steps, yielding in particular the literals $I(N_0,N_1,N_2,N_1)$ and $I(N_0,N_2,N_1,N_2)$ (which, without the three first components, were used to mount a false attack). On the other side, the goal is decomposed into $\neg I(x,y,z,N_1) \vee \neg I(x,y,z,N_2)$ and leads, using the two inferred literals, to clauses $\neg I(N_0,N_2,N_1,N_1)$ and $\neg I(N_0,N_1,N_2,N_2)$. But the empty clause cannot be derived: there is no one-session attack.

5.2 Decidable Fragments of First-order Logic

If we translate back in terms of strategies the constraint solving techniques used for the decidability and complexity proofs for a bounded number of sessions [10], we get a decision result for formulas in the following clausal form. In this theorem, the part of I's arguments that model ordering of protocol rules is put in subscript position to ease reading.

Theorem 2. *Assume that all clauses are of one of the following forms:*

1. $I_{\overline{z}}(\overline{x},y_1),\ldots,I_{\overline{z}}(\overline{x},y_n) \rightarrow I_{\overline{z}}(\overline{x},f(y_1,\ldots,y_n))$ *with* $\overline{x}$, $\overline{y}$, $\overline{z}$ *pairwise disjoint and distinct, and* $f \in \mathcal{F}$
2. $I_{\overline{z}_{[i\leftarrow k]}}(\overline{x},y) \rightarrow I_{\overline{z}_{[i\leftarrow k+1]}}(\overline{x},y)$ *with* $\{y\}$, $\overline{x}$, $\overline{z}$ *pairwize disjoint*
3. $I_{\overline{z}_{[i\leftarrow k]}}(\overline{x},s) \rightarrow I_{\overline{z}_{[i\leftarrow k+1]}}(\overline{x},t)$ *with* $Var(t) \subseteq Var(s) \subseteq \overline{x}$ *and* $\overline{x} \cap \overline{z} = \emptyset$
4. $I_{\overline{z}}(\overline{x},t)$ *with* $Var(t) = \emptyset$ *and* $\overline{x} \cap \overline{z} = \emptyset$
5. $\neg I_{\overline{z}}(\overline{x},s)$ *with* $Var(s) \subseteq \overline{x}$ *and* $\overline{x} \cap \overline{z} = \emptyset$

where $\overline{z}_{[i\leftarrow k]}$ *represents the variable-vector* $\overline{z}$ *whose* i^{th} *element is replaced by* k. *Then the satisfiability modulo the axioms of encryption/decryption (resp. satisfiability modulo exclusive-or [7, 11], resp. satisfiability modulo Abelian groups [15]) is co-NP-complete.*

This shows a new decidable fragment of first-order logic. It is related to both the extended Skolem class and the $\mathcal{E}^+$ class [13], but it is not subsumed by any of the two classes. This shows that it should be possible to design strategies in such a way that resolution becomes a decision procedure for the above class. This is exactly what we do in the next section for the Dolev-Yao intruder.

5.3 A Decision Procedure For the Security Problem

We provide a decision procedure for a class of clauses, which model security protocols with a Dolev-Yao intruder. It consists of a resolution strategy that we prove complete and terminating.

For the sake of simplicity, we explain our decision procedure without taking ordering of protocol rules into account. The latter can be added without compromising decidability as explained at the end of this section.

Here is the class in question. Note that clauses $(I0)$–$(I6)$ of Sect. 3.1 are intruder clauses.

Definition 2 ($\mathcal{C}^m$). *For any $m \in \mathbb{N}$, let $\mathcal{C}^m$ be the set of clauses C such that, for some vector of variables $\overline{x}$ of length m, C has one of the following forms:*

1. $I(\overline{x}, y_1), \ldots, I(\overline{x}, y_n) \rightarrow I(\overline{x}, f(y_1, \ldots, y_n))$ *with* $\overline{x} \cap \{y_1, \ldots, y_n\} = \emptyset$ *(the set $\mathcal{C}^m_C$ of* composition clauses*)*
2. $I(\overline{x}, f(u_1, \ldots, u_n)), I(\overline{x}, y_1), \ldots, I(\overline{x}, y_k) \rightarrow I(\overline{x}, y)$ *with* $\{y, y_1, \ldots, y_k\} \subseteq Var(u_1, \ldots, u_n) \subseteq \{u_1, \ldots, u_n\}$ *and* $\overline{x} \cap \{u_1, \ldots, u_n\} = \emptyset$ *(the set $\mathcal{C}^m_D$ of* decomposition clauses*)*
3. $I(\overline{x}, s) \rightarrow I(\overline{x}, t)$ *with* $Var(t) \cup Var(s) \subseteq \overline{x}$ *(the set $\mathcal{C}^m_P$ of* protocol clauses*)*
4. $\rightarrow I(\overline{x}, t)$ *with t ground (the set $\mathcal{C}^m_O$ of* initialization clauses*)*
5. $I(\overline{x}, t) \rightarrow$ *with t ground (the set $\mathcal{C}^m_G$ of* goal clauses*)*

The set of intruder clauses *is $\mathcal{C}^m_I = \mathcal{C}^m_C \cup \mathcal{C}^m_D$.*

Remarks:

- We assume in the following that our set of clauses always contains at least one element of $\mathcal{C}^m_O$ and at least one element of $\mathcal{C}^m_G$. Otherwise the set of clauses is trivially satisfiable.
- The condition that t is ground in 4 and 5 can be weaken to $Var(t) \subseteq \overline{x}$. Indeed, if t is not ground in some clause $\rightarrow I(\overline{x}, t)$, we can meet condition 4 by replacing it with the clauses $\rightarrow I(\overline{x}, a)$ and $I(\overline{x}, a) \rightarrow I(\overline{x}, t)$, where a is a fresh constant, provided $Var(t) \subseteq \overline{x}$. Similarly, clauses $I(\overline{x}, t) \rightarrow$ such that $Var(t) \subseteq \overline{x}$ can be replaced with the clauses $I(\overline{x}, b) \rightarrow$ and $I(\overline{x}, t) \rightarrow I(\overline{x}, b)$.
- Note that the protocol clauses do not require that $Var(t) \subseteq Var(s)$. Neither do we assume that variables or the terms u_i are distinct in the above definition. In these respects, the conditions are more general than those of Theorem 2: we may cover some cases that do not correspond to protocols.

Our strategy is based on binary resolution with free selection. To define this selection function, we consider a well-founded ordering $\preccurlyeq$, compatible with substitution, containing the subterm ordering and such that there are only finitely many terms smaller than a given term. An example of such an ordering is the subterm ordering itself. $\preccurlyeq$ is extended to atoms as follows: $A(t_1, \ldots, t_n) \preccurlyeq A'(t'_1, \ldots, t'_m)$ when $A = A'$, $m = n$ and $t_1 \preccurlyeq t'_1, \ldots, t_n \preccurlyeq t'_n$.

Definition 3. *Let Sel be the* selection function *such that for any Horn clause $C = A_1, \ldots, A_n \rightarrow B$ whose set of maximal atoms is MAX, then*

1. *if MAX is a singleton then Sel(C) is the only literal in MAX,*
2. *otherwise, if there is a maximal atom $A_i = I(\overline{s}, t)$ where t is not a variable, then return such an A_i,*
3. *otherwise, if $B = I(\overline{s}, t)$ is maximal and t is not a variable, then return B,*
4. *otherwise, return any literal.*

Definition 4. *We consider the following rule of* binary resolution with free selection *for Horn clauses:*

$$\frac{C \vee A \quad C' \vee \neg A'}{C\sigma \vee C'\sigma}$$

where $\sigma = Mgu(A =^? A')$, $Sel(C \vee A) = A$, and $Sel(C' \vee \neg A') = \neg A'$.

Remark 1. Let C be any clause derivable from $\mathcal{C}^m$ using the resolution rule of Definition 4. Then for any two atoms $A, A' \in C$, there exist $\overline{s}, t, t'$ such that $A = I(\overline{s}, t)$ and $A' = I(\overline{s}, t')$.

Definition 5. *A clause $I(\overline{s}, x_1), \ldots, I(\overline{s}, x_n) \rightarrow I(\overline{s}, x)$ where $x_1, \ldots, x_n, x$ are distinct variables is* contradictory.

Contradictory clauses yield unsatisfiability as soon as the sets of clauses in $\mathcal{C}_O^m$ and $\mathcal{C}_G^m$ are both non empty, which we assumed.

Definition 6. *A clause $A_1, \ldots, A_n \rightarrow B$ is* redundant *if $A_i = B$ for some i.*

Our strategy consists in applying the rule of Definition 4 to clauses that are not contradictory and deleting redundant clauses. Completeness is a consequence of known results (see e.g., Sect. 7.2 of [3]):

Theorem 3. *Our resolution strategy is refutationally complete for $\mathcal{C}^m$.*

The delicate problem is termination. One can easily see that an inappropriate strategy could cause non-termination. For example, standard binary resolution for the following two clauses of $\mathcal{C}^m$

$$I(x, y_1), I(x, y_2) \rightarrow I(x, \langle y_1, y_2 \rangle) \quad \text{and} \quad I(x, x) \rightarrow I(x, a)$$

yields the infinite set of clauses:

$I(\langle y_1, y_2 \rangle, y_1), I(\langle y_1, y_2 \rangle, y_2) \rightarrow I(\langle y_1, y_2 \rangle, a)$,
$I(\langle \langle y_1, y_3 \rangle, y_2 \rangle, y_1), I(\langle \langle y_1, y_3 \rangle, y_2 \rangle, y_3), I(\langle \langle y_1, y_3 \rangle, y_2 \rangle, y_2) \rightarrow I(\langle \langle y_1, y_3 \rangle, y_2 \rangle, a)$,
$\ldots$

This example explains why our selection function avoids resolution when the last argument of I is a variable (cases 2 and 3 of Definition 3).

Proving termination amounts to find some measure, for which resolvent clauses are smaller than their premises. We define such an ordering on clauses, comparing first the number N of variables occurring in the first arguments of I (corresponding to rigid variables) and next the size of their atoms with respect to the ordering $\preccurlyeq$.

As a first step, we prove an invariant showing, in particular, that N does not increase by resolution. More formally, any atom in any clause derivable from any subset of $\mathcal{C}^m$ is of the form $I(\overline{s}, t)$ with $Var\,(\overline{s}) \subseteq \overline{s}$. This is the invariant 2 in the following lemma:

Lemma 1. *Let $\mathcal{C} \subseteq \mathcal{C}^m$. For any clause C derivable from $\mathcal{C}$ by our resolution strategy, the following invariant holds:*

1. *(a) There is a vector of terms $\overline{s}$ such that every atom of C is of the form $I(\overline{s}, t)$ with $Var\,(t) \subseteq Var\,(\overline{s})$, or*
   (b) C is an intruder clause (and in particular every atom of C is of the form $I(\overline{x}, t)$ with $Var\,(t) \cap Var\,(\overline{x}) = \emptyset$).
2. *Every atom of C is of the form $I(\overline{s}, t)$ with $Var\,(\overline{s}) \subseteq \overline{s}$.*

Proof. (Sketch) The proof goes by induction on the length of the derivation of C, and by case analysis on the possible premises of the resolution rule. In most cases, invariants are easily shown to be preserved, except in a few cases where the proof of invariant 2 requires the following lemma:

Lemma 2. *Let $\overline{s}$, $\overline{s'}$ such that $|\overline{s}| = |\overline{s'}|$, $Var\,(\overline{s}) \subseteq \overline{s}$, and $Var\left(\overline{s'}\right) \subseteq \overline{s'}$. If $\sigma = Mgu(\overline{s} =^? \overline{s'})$, then $Var\,(\overline{s}\sigma) \subseteq \overline{s}\sigma$ and $Var\left(\overline{s'}\sigma\right) \subseteq \overline{s'}\sigma$.*

The detailed proofs of these lemmas are given in Appendix A.

We are half-way of proving termination. Using Lemma 1, we show roughly that, for any resolution step $C \dashv C_1, C_2$, either (1) the number of variables that encode rigid variables in C is stricly smaller than the number of such variables in C_1 or C_2, or (2) the number of such variables is unchanged and the atoms of the resolvent are smaller (w.r.t. $\preccurlyeq$) than those of the premises (Lemma 6 of Appendix B). Then we can show:

Lemma 3. *Any derivation sequence using our resolution strategy and starting with a finite subset of $\mathcal{C}^m$ is finite.*

Proof. (Sketch) Let $\mathcal{C}$ be a finite subset of $\mathcal{C}^m$, and let $R(C)$ be the vector $\overline{s}$, as defined in Lemma 1 (by Remark 1, this vector is independent of the chosen atom in C). We show by induction that, for any $n \leq m$, there are only finitely many clauses C derivable from the clauses in $\mathcal{C}$ such that $\phi(C) = |\,Var\,(R(C))\,| = m-n$.

If $C \dashv C_1, C_2$ and $\phi(C) = m-n$, then either $\phi(C_1), \phi(C_2) > m-n$, which can only occur finitely many times by induction hypothesis. Or else $\phi(C_1) = \phi(C)$ and $\phi(C_2) \geq \phi(C_1)$, in which case $R(C) = R(C_1)$ (up to renaming). Hence the set of vectors $R(C_1)$ such that $\phi(C_1) = m - n$ is finite, up to renaming. Next, once $R(C_1)$ is fixed, there are only finitely many possible atoms in C_1, since new clauses C' such that $R(C_1) = R(C')$ can only be obtained when unification is a renaming. The detailed proof is given in Appendix B.

Corollary 1. *Our resolution strategy is a decision procedure for the class $\mathcal{C}^m$.*

Including an Ordering on Stages. Faithfully representing the protocol instances requires to record state information, as explained in Sect. 3.2. For this purpose, we add another component to the predicate I, to record at which stage of each session messages are known.

If there are n sessions, we represent the stages by a vector of n local states. Several data structures can be used for this; we do not commit to any of them and simply write $f(q_1, \ldots, q_n)$ when each session i has reached the stage q_i. To restrict protocol clauses to the appropriate stages, instead of a clause $I(\overline{x}, s) \rightarrow I(\overline{x}, t)$, we consider a clause

$$I(f(z_1, \ldots, z_{i-1}, q_j, z_{i+1}, \ldots, z_n), \overline{x}, s) \rightarrow I(f(z_1, \ldots, z_{i-1}, q_{j+1}, z_{i+1}, \ldots, z_n), \overline{x}, t)$$

for the jth rule of session i. We also need clauses $I(z, \overline{x}, y), z' > z \rightarrow I(z', \overline{x}, y)$ to express the increasingness of the intruder knowledge; how "$>$" is implemented is not relevant here.

Our resolution procedure can be extended to such clauses: we simply ignore the first component of I in the resolution strategy. Since there are only finitely many possible instances of the first component of I, our termination result can be applied and we get a complete and terminating procedure.

5.4 Enhancing First-order Provers for Security Protocols

Another possible use of Theorem 1 is to combine in a single first-order theorem prover the advantages of the approximations and of the bounded number of sessions: using the same engine and specification it is possible to look first for attacks/security in an exact way for a given number of sessions and then use an approximation for more sessions. Alternatively, in case a candidate attack is found, we can check the falsity of the attack using additional clauses.

6 Conclusion

We showed a simple encoding of rigid variables by translation to first-order logic. This encoding can be applied to the verification of security protocols for a bounded number of sessions, without introducing false attacks.

It opens some perspectives in automated deduction: decidability results in the verification of security protocols correspond to non-trivial decidable fragments of first-order logic. We illustrated this, showing a resolution-based decision procedure for the verification of security protocols in a standard Dolev-Yao model.

Our first-order formalisation and the decision procedure thereof are easy to implement (we have a prototype implementation, but we could also rely on any first-order theorem prover). It is also flexible, compared to other techniques such as constraint solving: we can easily change the intruder theory, consider other security properties, etc. the procedure would still work, without generating false attacks. Of course, there will be no guarantee of termination until the selection strategy is tuned according to the new theory. In this respect, it remains to design more selection strategies, for other intruder theories, including for instance algebraic properties of security primitives.

Acknowledgment. We are grateful to the anonymous reviewers for their very helpful remarks.

References

1. Allamigeon, X., Blanchet, B.: Reconstruction of Attacks against Cryptographic Protocols. In *18th IEEE Work. on Computer Security Foundations*, pages 140–154, 2005.
2. Andrews, P. B.: Theorem proving via general matings. *Journal of the ACM*, 28(2):193–214, 1981.
3. Bachmair, L., Ganzinger, H.: Resolution Theorem Proving. In *Handbook of Automated Reasoning,*, chapter 2, pages 19–99. Elsevier and MIT Press, 2001.
4. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Work. on Computer Security Foundations*, pages 82–96, 2001.
5. Dershowitz, N., Jouannaud, J.-P.: Rewrite Systems. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B)*, pages 243–320. Elsevier and MIT Press, 1990.
6. Cervesato, I., Durgin, N. A., Lincoln, P. D., Mitchell, J. C., Scedrov, A.: A meta-notation for protocol analysis. In *12th IEEE Work. on Computer Security Foundations*, pages 55–69, 1999.
7. Chevalier, Y., Küsters, R., Rusinowitch, M., Turuani, M.: An NP decision procedure for protocol insecurity with XOR. In *18th IEEE Symp. on Logic in Computer Science (LICS 2003)*, pages 261–270.
8. Cohen, A.: Combined CPV-TLV Security Protocol Verifier. Master's thesis, New York University, 2004.
9. Comon-Lundh, H., Cortier, V.: Security properties: two agents are sufficient. *Science of Computer Programming*, 50(1–3):51–71, 2004.
10. Comon-Lundh, H., Cortier, V., Zalinescu, E.: Deciding security properties for cryptographic protocols. Application to key cycles. To appear in ACM Transactions on Computational Logic.
11. Comon-Lundh, H., Shmatikov, V.: Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *18th IEEE Symp. on Logic in Computer Science (LICS 2003)*, pages 271–280.
12. Delaune, S., Lin, H., Lynch, C.: Protocol verification via rigid/flexible resolution. In *14th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2007)*, volume 4790 of *Lecture Notes in Artificial Intelligence*, pages 242–256. Springer.
13. Fermüller, C. G., Leitsch, A., Hustadt, U., Tamet, T.: Resolution decision procedure. In *Handbook of Automated Reasoning*, chapter 25. Elsevier and MIT Press, 2001.
14. Rusinowitch, M., Turuani, M.: Protocol insecurity with a finite number of sessions, composed keys is NP-complete. *Theoretical Computer Science*, 1-3(299):451–475, 2003.
15. Shmatikov, V.: Decidable analysis of cryptographic protocols with products and modular exponentiation. In *13th European Symp. on Programming (ESOP 2004)*, volume 2986 of *Lecture Notes in Computer Science*, pages 355–369. Springer.
16. Weidenbach, C.: Towards an automatic analysis of security protocols in first-order logic. In *16th Int. Conf. on Automated Deduction (CADE 1999)*, volume 1632 of *Lecture Notes in Computer Science*, pages 314–328. Springer.

A Proof of Lemma 1

By induction on the length of the derivation.

Base case. The intruder clauses verify invariant 1b by definition and invariant 2 because $Var(\overline{x}) = \overline{x} \subseteq \overline{x}$. The protocol clauses verify invariant 1a because $Var(t) \subseteq Var(\overline{x})$ by definition. They verify invariant 2 because $Var(\overline{x}) = \overline{x} \subseteq \overline{x}$. The initialization clauses trivially verify invariant 1a, because $Var(t) = \emptyset$, and invariant 2 because $Var(\overline{x}) = \overline{x} \subseteq \overline{x}$. The goal clauses trivially verify invariant 1a, because $Var(t) = \emptyset$, and invariant 2, because $Var(\overline{x}) = \overline{x} \subseteq \overline{x}$.

Inductive case. There are several cases.

1. Resolution between two clauses $I(\overline{s}, t_1), \ldots, I(\overline{s}, t_n) \rightarrow I(\overline{s}, t)$ $(n \geq 1)$ and $I(\overline{s'}, t'_1), \ldots, I(\overline{s'}, t'_k) \rightarrow I(\overline{s'}, t')$ $(k \geq 1)$, verifying invariants 1a and 2, with $\sigma = Mgu(I(\overline{s}, t) =^? I(\overline{s'}, t'_j))$.

   The resolvent verifies invariant 1a, i.e.,

   $$Var(t_i\sigma) \subseteq Var(\overline{s}\sigma), \quad Var(t'_i\sigma) \subseteq Var(\overline{s'}\sigma) \ (i \neq j), \text{ and } Var(t'\sigma) \subseteq Var(\overline{s'}\sigma).$$

   Let us prove the inclusion $Var(t_i\sigma) \subseteq Var(\overline{s}\sigma)$ (other inclusions are similar). Consider some $x \in Var(t_i\sigma)$. If $x \notin Var(\sigma)$, then $x \in Var(t_i)$. By the induction hypothesis, $x \in Var(\overline{s})$. Thus $x \in Var(\overline{s}\sigma)$. If $x \in VRange(\sigma)$, then there is some $x' \in Dom(\sigma) \cap Var(t_i)$ such that $x \in Var(x'\sigma)$. By the induction hypothesis, $x' \in Var(\overline{s})$. Thus $x \in Var(\overline{s}\sigma)$. (Since we assume that mgus are idempotent, we do not need to check the case where $x \in Dom(\sigma)$.)

   The resolvent verifies invariant 2, i.e., $Var(\overline{s}\sigma) \subseteq \overline{s}\sigma$.

   We apply Lemma 2 to $\overline{s} \uplus t$, $\overline{s'} \uplus t'_j$, and σ. This leads to $Var((\overline{s} \uplus t)\sigma) \subseteq (\overline{s} \uplus t)\sigma$, i.e., $Var(\overline{s}\sigma \cup t\sigma) \subseteq \overline{s}\sigma \uplus t\sigma$. We can show that $Var(t\sigma) \subseteq Var(\overline{s}\sigma)$ (proof similar to the one for preservation of invariant 1a just above).

   If $t\sigma$ is a variable, then t is a variable and, by invariant 2, $t \in \overline{s}$. Then $t\sigma \in \overline{s}\sigma$. Therefore $Var(\overline{s}\sigma) \subseteq \overline{s}\sigma \uplus t\sigma$ implies $Var(\overline{s}\sigma) \subseteq \overline{s}\sigma$.

   If t is a functional term, then $Var(\overline{s}\sigma) \subseteq \overline{s}\sigma \uplus t\sigma$ implies $Var(\overline{s}\sigma) \subseteq \overline{s}\sigma$.

2. Resolution between a clause $I(\overline{s}, t_1), \ldots, I(\overline{s}, t_k) \rightarrow I(\overline{s}, t)$ $(k \geq 1)$ verifying invariants 1a and 2 and a composition clause $I(\overline{x}, y_1), \ldots, I(\overline{x}, y_n) \rightarrow I(\overline{x}, f(y_1, \ldots, y_n))$ $(n \geq 1)$, with $\sigma = Mgu(I(\overline{x}, f(y_1, \ldots, y_n)) =^? I(\overline{s}, t_j))$.

   The resolvent verifies invariant 1a, i.e.,

   $$Var(y_i\sigma) \subseteq Var(\overline{x}\sigma), \quad Var(t_i\sigma) \subseteq Var(\overline{s}\sigma) \ (i \neq j), \text{ and } Var(t\sigma) \subseteq Var(\overline{s}\sigma).$$

   Let us prove $Var(y_i\sigma) \subseteq Var(\overline{x}\sigma)$ (the proofs of other inclusions are similar to the corresponding proof in case 1). Consider some $x \in Var(y_i\sigma)$. t_j is a functional term. Indeed, if t_j were a variable, then by definition of our selection function, all t_i's and t would be variables. Then the clause would be either redundant or contradictory, two cases that are discarded.

By hypothesis, $y_i\sigma = t_j|_i\sigma$. Since $Var(t_j) \subseteq Var(\overline{s})$, then $Var(t_j|_i) \subseteq Var(\overline{s})$, and $Var(t_j|_i\sigma) \subseteq Var(\overline{s}\sigma)$. Thus, $x \in Var(\overline{s}\sigma)$. We can conclude because $\overline{x}\sigma = \overline{s}\sigma$.

The resolvent verifies invariant 2, i.e., $Var(\overline{x}\sigma) \subseteq \overline{x}\sigma$.

t'_j is not a variable (this has already been shown in the preservation of invariant 1a just above). It is therefore a functional term of the form $t'_j = f(t'_j|_1, \ldots, t'_j|_n)$ such that $Var\left(t'_j|_i\right) \subseteq Var(\overline{s})$ for all i. We apply Lemma 2 to $\overline{x} \uplus y_1 \uplus \cdots \uplus y_n$, $\overline{s} \uplus t'_j|_1 \uplus \cdots \uplus t'_j|_n$, and σ. This shows that $Var\left((\overline{s} \uplus t'_j|_1 \uplus \cdots \uplus t'_j|_n)\sigma\right) \subseteq (\overline{s} \uplus t'_j|_1 \uplus \cdots \uplus t'_j|_n)\sigma$.

We can show that $Var\left(t'_j|_i\sigma\right) \subseteq Var(\overline{s}\sigma)$ for all i (proof similar to the proofs of preservation of invariant 1a in case 1). We also know that no $t'_j|_i$ is a variable that does not appear in $\overline{s}$. Thus, $Var(\overline{s}\sigma) \subseteq \overline{s}\sigma$, which implies $Var(\overline{x}\sigma) \subseteq \overline{x}\sigma$ since $x\sigma = \overline{s}\sigma$.

3. Resolution between a clause $I(\overline{s}, t_1), \ldots, I(\overline{s}, t_l) \rightarrow I(\overline{s}, t)$ $(l \geq 1)$ verifying invariants 1a and 2, and a decomposition clause $I(\overline{x}, f(u_1, \ldots, u_n)), I(\overline{x}, y_1), \ldots, I(\overline{x}, y_k) \rightarrow I(\overline{x}, y)$ $(n, k \geq 1)$ with $\sigma = Mgu(I(\overline{s}, t) =^? I(\overline{x}, f(u_1, \ldots, u_n)))$. Let us note $u = f(u_1, \ldots, u_n)$.

   The resolvent verifies invariant 1a, i.e.,

$$Var(t_p\sigma) \subseteq Var(\overline{s}\sigma) \text{ for all } p, \quad Var(y_j\sigma) \subseteq Var(\overline{x}\sigma) \text{ for all } j, \quad Var(y) \subseteq Var(\overline{x}\sigma).$$

   We have $\overline{x}\sigma = \overline{s}\sigma$ and $u\sigma = t\sigma$. By induction hypothesis $Var(t_p) \subseteq Var(\overline{s})$, hence $Var(t_p\sigma) \subseteq Var(\overline{s}\sigma)$. By definition of decomposition clauses $\mathcal{C}_D^m$, $Var(y_j\sigma) \subseteq Var(u\sigma)$. Using again the induction hypothesis, $Var(t\sigma) \subseteq Var(\overline{s}\sigma)$. Thus, $Var(y_j\sigma) \subseteq Var(u\sigma) = Var(t\sigma) \subseteq Var(\overline{s}\sigma) = Var(\overline{x}\sigma)$.

   The resolvent verifies invariant 2, i.e., $Var(\overline{s}\sigma) \subseteq \overline{s}\sigma$. As before, since the premises of a resolution step are neither redundant nor contradictory, t is a functional term of the form $f(t_1, \ldots, t_n)$. We apply Lemma 2 to $\overline{s} \uplus t_1 \uplus \cdots \uplus t_n$, $\overline{x} \uplus u_1 \uplus \cdots \uplus u_n$, and σ. (This is possible, thanks to the hypotheses on the decomposition clauses $\mathcal{C}_D^m$.) This shows that

$$Var((\overline{s} \uplus t_1 \uplus \cdots \uplus t_n)\sigma) \subseteq (\overline{s} \uplus t_1 \uplus \cdots \uplus t_n)\sigma.$$

   We can show that $Var(t_i\sigma) \subseteq Var(\overline{s}\sigma)$ for all i. We also know that if t_i is a variable then it appears in $\overline{s}$. Thus, $Var(\overline{s}\sigma) \subseteq \overline{s}\sigma$.

4. Resolution between two intruder clauses: a composition clause

$$I(\overline{x}, y_1), \ldots, I(\overline{x}, y_n) \rightarrow I(\overline{x}, f(y_1, \ldots, y_n))$$

   and a decomposition clause

$$I(\overline{x'}, f'(u_1, \ldots, u_{n'})), I(\overline{x'}, y'_1), \ldots, I(\overline{x'}, y'_k) \rightarrow I(\overline{x'}, y')$$

   By definition of our selection function,

$$\sigma = Mgu(f(y_1, \ldots, y_n) =^? f'(u_1, \ldots, u_{n'})).$$

The resolvent is a redundant clause. Indeed, $\overline{x}\sigma = \overline{x'}\sigma$ (σ is a renaming on $\overline{x}, \overline{x'}$) and, for every i, $y_i\sigma = u_i\sigma$. By definition of decomposition clauses $\mathcal{C}_D^m$, $y' \in \{u_1, \ldots, u_n\}$. Thus, $y'\sigma = y_i\sigma$ for some i.

5. Other cases. Resolution between an initialization clause and a clause verifying invariants 1a and 2 is similar to case 1 (take $n = 0$). Resolution between a goal clause and a clause verifying invariants 1a and 2, is covered by case 1 (take $k = 1$ and no right-hand side for the second clause). Resolution between an initialization clause and a goal clause is similar to case 1 (take $n = 0$, $k = 1$, and no right-hand side for the second clause). Resolution between a composition clause and a goal clause is covered by case 2 (take $k = 1$ and no right-hand side for the first clause). Resolution between a decomposition clause and an initialization clause is similar to case 3 (take $l = 0$).

Lemma 4. *Let t, t' be terms such that σ unifies t and t'. For all $x \notin Var(\sigma)$, $x \in Var(t)$ iff $x \in Var(t')$.*

Proof. $t\sigma = t'\sigma$. Since $x \notin Var(\sigma)$, $x \in Var(t)$ iff $x \in Var(t\sigma)$ iff $x \in Var(t'\sigma)$ iff $x \in Var(t')$.

Lemma 5. *Let t, t' be terms such that $\sigma = Mgu(t =^? t')$. For all $x \in Dom(\sigma)$, if $x \in Var(t)$ then $Var(x\sigma) \subseteq Var(t')$.*

Proof. $t\sigma = t'\sigma$. By induction on the structure of t'.

Base case. Suppose that t' is some variable z'. We do a case analysis on t. If t is a variable, then it is the variable x, in which case $\sigma = \{x \mapsto z'\}$ and we indeed have $Var(x\sigma) = \{z'\} \subseteq Var(t') = \{z'\}$. If $t = f(t_1, \ldots, t_p)$, then $x \in VRange(\sigma)$, which contradicts the hypotheses.

Inductive case. Suppose that $t' = f'(t'_1, \ldots, t'_n)$. Then we also have $t = f'(t_1, \ldots, t_n)$ such that for some i, $x \in Var(t_i)$. $\sigma|_{Var(t_i) \cup Var(t'_i)} = Mgu(t_i =^? t'_i)$. By the inductive hypothesis, $Var\left(x\sigma|_{Var(t_i) \cup Var(t'_i)}\right) \subseteq Var(t'_i)$, from which we derive $Var(x\sigma) \subseteq Var(t')$.

A.1 Proof of Lemma 2

By induction on $|Var(\overline{s})| + |Var(\overline{s'})|$.

Base case. There is no variable. All s_i's and s'_i's are ground terms. Thus σ is empty and the inclusions are trivially verified.

Inductive case. We have $|Var(\overline{s})| + |Var(\overline{s'})| > 0$. If σ is a renaming, then the goal is trivially verified. Let us assume that σ is not a renaming; for the sake of simplicity, we assume that σ is idempotent. Then there is at least one variable $x \in Var(\overline{s})$ (or $Var(\overline{s'})$, but this is symmetric) such that the equation set contains an equation $s_i =^? s'_i$ such that, for some p, $s_i|_p = x$ and $s'_i|_p$ is defined and different from x.

Suppose for the sake of simplicity that $s_i = x$. (Otherwise, decompose the equation $s_i =^? s'_i$ and transform the equation set so as to obtain such an equation.)

We can reorder the equation set as follows:

$$\overline{s} =^? \overline{s'} \Leftrightarrow \begin{cases} x =^? s'_0 \\ s_1 =^? s'_1 \\ \vdots \\ s_m =^? s'_m \end{cases}$$

We do a case analysis on s'_0.

Suppose that s'_0 is some variable x'. We replace x with x' in all s_i's and s'_i's and obtain a new equation set $\overline{s_1} =^? \overline{s'_1}$. The inclusions $Var(\overline{s_1}) \subseteq \overline{s_1}$ and $Var\left(\overline{s'_1}\right) \subseteq \overline{s'_1}$ hold and there is one variable less. By construction, this new equation set has a mgu σ_1 and, by inductive hypothesis, we have $Var(\overline{s_1}\sigma_1) \subseteq \overline{s_1}\sigma_1$ and $Var\left(\overline{s'_1}\sigma_1\right) \subseteq \overline{s'_1}\sigma_1$. Using σ_1, we build the mgu σ for the original equation set as follows:

- If $x' \in Dom(\sigma_1)$, then $\sigma = \sigma_1 \uplus \{x \mapsto x'\sigma_1\}$.
- If $x' \in VRange(\sigma_1)$ or $x' \notin Var(\sigma_1)$, then $\sigma = \sigma_1 \uplus \{x \mapsto x'\}$.

In each case, we have $Var(\overline{s}\sigma) = Var(\overline{s_1}\sigma_1) \subseteq \overline{s_1}\sigma_1 = \overline{s}\sigma$ and similarly for $\overline{s'}$.

Suppose that s'_0 is a functional term $f(t_1, \ldots, t_p)$. We replace x with s'_0 in all s_i's and s'_i's and obtain a new equation set $\overline{s_1} =^? \overline{s'_1}$. $Var(\overline{s_1}) \subseteq \overline{s_1}$ does not necessarily hold because the previous replacement may introduce new variables $x'_j \in Var(s'_0)$ in $\overline{s}$. However, since $Var\left(\overline{s'}\right) \subseteq \overline{s'}$, for any such x'_j, there is an equation $s'_{i_j} =^? x'_j$. We replace each x'_j with s'_{i_j} in $\overline{s_1}$ and obtain a new equation set $\overline{s_2} =^? \overline{s'_1}$. In this new system, the desired inclusions hold and there is one variable less. By construction, this new equation set has an mgu σ_1 and, by the inductive hypothesis, we have $Var(\overline{s_2}\sigma_1) \subseteq \overline{s_2}\sigma_1$ and $Var\left(\overline{s'_1}\sigma_1\right) \subseteq \overline{s'_1}\sigma_1$. Using σ_1, we build the mgu $\sigma = \sigma_1 \uplus \{x \mapsto s'_0\sigma_1\}$ for the original equation set.

B Proof of Lemma 3

We define two functions R and ϕ as follows. For any clause $C \in \mathcal{C}^*$, any atom in C can be written $I(\overline{s}, t)$; then $R(I(\overline{s}, t)) = \overline{s}$ and $\phi(I(\overline{s}, t)) = |Var(\overline{s})|$. By Remark 1, we can overload the notations and extend R and ϕ to clauses.

Our goal is to show that $\mathcal{C}^*$ is finite. We will show more generally that for any n such that $0 \leq n \leq m$, there are only finitely many clauses $C \in \mathcal{C}^*$ such that $\phi(C) = m - n$, i.e., for all $n \leq m$, $\{C \in \mathcal{C}^* \mid \phi(C) = m - n\}$ is finite. The proof goes by induction on n.

Base case: We prove that there are finitely many clauses $C \in \mathcal{C}^*$ such that $\phi(C) = m$. This follows from Lemma 7 below.

Inductive case: By the inductive hypothesis, $S_0 = \{C \in \mathcal{C}^* \mid \phi(C) > m - n\}$ is finite. Our goal is to show that $\{C \in \mathcal{C}^* \mid \phi(C) = m - n\}$ is finite. We show that this set is included in another finite set, that we now define.

Let $S_1 = \{C \in \mathcal{C} \mid \phi(C) = m-n\} \cup \{C \mid C \dashv C_1, C_2 \in S_0 \text{ and } \phi(C) = m-n\}$. By construction, S_1 is finite. Let $F = \{R(C) \mid C \in S_1\}$. Since S_1 is finite, F is also finite. Let $S_2 = \{C\sigma \mid C \in S_0 \text{ and } R(C\sigma) \in F \text{ for some substitution } \sigma\}$. S_2 is also finite because S_0 is finite and there is only a finite number of ways to build the substitutions. We show that $\{C \in \mathcal{C}^* \mid \phi(C) = m - n\}$ is included in

$$\{C \mid C \dashv^* S_1 \cup S_2 \cup \mathcal{C}_I^m \text{ and } R(C)\rho \in F \text{ for some renaming } \rho\}.$$

By Lemma 7, this set is indeed finite. The inclusion proof goes by induction on the length of the derivation of C.

Base case: $C \in \mathcal{C}$ and $\phi(C) = m - n$, thus $C \in S_1$ and $R(C) \in F$.

Inductive case: let $C \in \mathcal{C}^*$ be such that the derivation length is strictly positive and $\phi(C) = m-n$. To fix notation, assume w.l.o.g. that $C \dashv C_1, C_2$ and $\phi(C_1) \leq \phi(C_2)$. According to Lemma 6, there are four cases:

1. $\phi(C) < \phi(C_1)$. Since $\phi(C) = m-n$, then $\phi(C_1) > m-n$ and $\phi(C_2) > m-n$. Thus $C_1, C_2 \in S_0$, which implies $C \in S_1$ and $R(C) \in F$.
2. $\phi(C) = \phi(C_1)$, there is a renaming ρ on $Var\,(C_1)$ such that $R(C) = R(C_1)\rho$ and C_2 is an intruder clause. Since $\phi(C_1) = m - n$, by the inductive hypothesis, $C_1 \dashv^* S_1 \cup S_2 \cup \mathcal{C}_I^m$ and $R(C_1)\rho' \in F$ for some renaming ρ'. Thus, $C \dashv^* S_1 \cup S_2 \cup \mathcal{C}_I^m$ and $R(C)\theta \in F$ for some renaming θ.
3. $\phi(C) = \phi(C_1) < \phi(C_2)$, there is a mgu ρ for the resolution step $C \dashv C_1, C_2$ that is a renaming on $Var\,(C_1)$ such that $R(C) = R(C_1)\rho = R(C_2)\rho$, and C_2 is not an intruder clause. By the inductive hypothesis, $C_1 \dashv^* S_1 \cup S_2 \cup \mathcal{C}_I^m$ and there is a renaming ρ' such that $R(C_1)\rho' \in F$.

   $\phi(C_2) > m - n$, hence $C_2 \in S_0$. Moreover, there is a renaming θ such that $R(C_2)\theta \in F$, thus $C_2' = C_2\theta \in S_2$. Then $C \dashv C_1, C_2'$, because θ is a renaming. Thus $C \dashv^* S_1 \cup S_2 \cup \mathcal{C}_I^m$ and $R(C)\theta \in F$
4. $\phi(C) = \phi(C_1) = \phi(C_2)$. There are renamings ρ_1, ρ_2 such that $R(C) = R(C_1)\rho_1 = R(C_2)\rho_2$. We apply the inductive hypothesis to both C_1 and C_2. We deduce that $C \dashv^* S_1 \cup S_2 \cup \mathcal{C}_I^m$ and that there is a renaming ρ' such that $R(C_1)\rho' \in F$ and a renaming ρ'' such that $R(C_2)\rho'' \in F$. There is therefore a renaming θ such that $R(C)\theta \in F$.

Lemma 6. *Consider $C_1, C_2 \in \mathcal{C}^*$ such that $\phi(C_1) \leq \phi(C_2)$. For any $C \dashv C_1, C_2$ one of the following holds:*

1. *$\phi(C) < \phi(C_1)$*
2. *$\phi(C) = \phi(C_1)$, there is a renaming ρ on $Var\,(C_1)$ such that $R(C) = R(C_1)\rho$, C_1 is not an intruder clause, C_2 is an intruder clause, and for any atom A occurring in C there is an atom A_1 occurring in C_1 such that $A \preccurlyeq A_1\rho$.*
3. *$\phi(C) = \phi(C_1) < \phi(C_2)$, there is a mgu ρ for the resolution step that is a renaming on $Var\,(C_1)$ such that $R(C) = R(C_1)\rho = R(C_2)\rho$, and neither C_1 nor C_2 are intruder clauses.*
4. *$\phi(C) = \phi(C_1) = \phi(C_2)$, there is a renaming ρ_1 on $Var\,(R(C_1))$ and a renaming ρ_2 on $Var\,(C_2)$ such that $R(C) = R(C_1)\rho_1 = R(C_2)\rho_2$, and every atom occurring in C is a renaming of an atom occurring in C_1 or C_2.*

Proof. Let σ be a mgu for the resolution step $C \dashv C_1, C_2$. We have $R(C) = R(C_1)\sigma = R(C_2)\sigma$. By definition of $\mathcal{C}_m$ and invariant 2 of Lemma 1, we also have $Var\,(R(C_1)) \subseteq R(C_1)$ and $Var\,(R(C_1)\sigma) \subseteq R(C_1)\sigma$. Thus:

$$\phi(C) = |\,Var\,(R(C_1)\sigma)\,| = |R(C_1)\sigma \cap \mathcal{X}| \text{ and } \phi(C_1) = |\,Var\,(R(C_1))\,| = |R(C_1) \cap \mathcal{X}|$$

In general $|R(C_1)\sigma \cap \mathcal{X}| \leq |R(C_1) \cap \mathcal{X}|$, and the equality is achieved iff σ is a renaming on $Var\,(R(C_1))$. Suppose that we do not have equality, then we have $\phi(C) < \phi(C_1)$ and we fall in case 1. Henceforth, suppose that we have the equality $\phi(C) = \phi(C_1)$. Since $\phi(C) = \phi(C_1)$, then σ is a renaming on $Var\,(R(C_1))$. Observe that C_1 and C_2 cannot be both intruder clauses simultaneously because of our resolution strategy. We do a case analysis on C_1, C_2.

Suppose that neither C_1 nor C_2 are intruder clauses. Then we can show that σ is not only a renaming on $Var\,(R(C_1))$ but also a renaming on $Var\,(C_1)$. By hypothesis, $\phi(C) \leq \phi(C_2)$. If $\phi(C) < \phi(C_2)$, then we fall in case 3. If $\phi(C) = \phi(C_2)$, then we can show as above that σ is also a renaming on $Var\,(C_2)$ and we fall in case 4.

Suppose that C_1 is not an intruder clause and that C_2 is an intruder clause. We show that we fall in case 2. First, observe that σ is a renaming on $Var\,(C_1)$. Assume that the resolution step unifies the atom A_2 of C_2 and some atom A_1 of C_1. A_2 is the only maximal literal in C_2 because it is an intruder clause. Every atom A of C is either a renaming of an atom of C_1 or an atom $A'_2\sigma$ for some $A'_2 \prec A_2$ in C_2; then $A'_2\sigma \prec A_2\sigma = A_1\sigma$.

Suppose that C_1 is an intruder clause but C_2 is not an intruder clause. Then we have $\phi(C_1) \geq \phi(C_2)$ because ϕ is maximal for intruder clauses. Thus $\phi(C_1) = \phi(C_2) = \phi(C)$. We can show that σ is a renaming on $Var\,(C_2)$. We fall in case 4.

Suppose that C_1 is not an intruder clause and that C_2 is an intruder clause. Then σ is a renaming on $Var\,(C_1)$. By hypothesis, $\phi(C_1) \leq \phi(C_2)$. If $\phi(C_1) < \phi(C_2)$, we fall in case 3. If $\phi(C_1) = \phi(C_2)$, then we can show that σ is also a renaming on $Var\,(C_2)$ and we fall in case 4.

Lemma 7. *Let $n \in \mathbb{N}$ and $S \subseteq \mathcal{C}^*$ be a finite set of clauses such that, for every $C \in S$, $\phi(C) = n$. Let S^* be the set of clauses C that are derivable using our resolution strategy from clauses in $S \cup \mathcal{C}_I^m$ and such that $\phi(C) = n$. Then S^* is finite.*

Proof. Lemma 6 shows that clauses in S^* are derivable from S using some resolution strategy which further restricts the resolvent C to be such that $\phi(C) = n$.

Furthermore, again by Lemma 6 and by induction on the derivation length, any atom A occurring in a clause of S^* is such that there is some atom A' occurring in some clause of S and some renaming ρ such that $A \preccurlyeq A'\rho$ (only cases 2 and 4 of Lemma 6 can occur).

By hypothesis on our ordering and by finiteness of S, it follows that there are only finitely many atoms in S^*, and therefore only finitely many clauses, up to renaming.