J Autom Reasoning
DOI 10.1007/s10817-010-9205-y

Unification Modulo Homomorphic Encryption

Siva Anantharaman · Hai Lin · Christopher Lynch · Paliath Narendran · Michael Rusinowitch

Received: 25 August 2010 / Accepted: 31 August 2010

Abstract Encryption 'distributing over pairs' is a technique employed in several cryptographic protocols. We show that unification is decidable for an equational theory HE specifying such an encryption. The method consists in transforming any given problem in such a way, that the resulting problem can be solved by combining a graph-based reasoning on its equations involving the homomorphisms, with a syntactic reasoning on its pairings. We show HE-unification to be NP-hard and in EXPTIME. We also indicate, briefly, how to extend HE-unification to Cap unification modulo HE, that can be used as a tool for modeling and analyzing cryptographic protocols where encryption follows the ECB mode, i.e., is done block-wise on messages.

Keywords Rewriting · Unification · Protocol analysis

Work supported by NSF Grants CNS-0831305 and CNS-0831209,
and partially supported by the FP7-ICT-2007-1 Project no. 216471, AVANTSSAR.

S. Anantharaman (✉)
LIFO, University of Orléans, Orléans, France
e-mail: siva@univ-orleans.fr

H. Lin · C. Lynch
Clarkson University, Potsdam, NY, USA

H. Lin
e-mail: linh@clarkson.edu

C. Lynch
e-mail: clynch@clarkson.edu

P. Narendran
University at Albany-SUNY, Albany, NY, USA
e-mail: dran@cs.albany.edu

M. Rusinowitch
Loria-INRIA Lorraine, Nancy, France
e-mail: rusi@loria.fr

Published online: 15 September 2010

1 Introduction

Several methods based on rewriting have been proposed with success, for the formal analysis of cryptographic protocols. The following Dolev–Yao system (DY) underlies many of them:

$$\begin{aligned} p_1(x.y) &\to x & dec(enc(x, y), y) &\to x \\ p_2(x.y) &\to y & enc(dec(x, y), y) &\to x \end{aligned} \qquad \text{(DY)}$$

The '.' here is the '*pairing*' operation on messages, p_1, p_2 are the respective projections from pairs, and $dec(x, y)$ (resp. $enc(x, y)$) stands for decryption (resp. encryption) of message x, using y as key; the second arguments of '*dec*' and '*enc*' are therefore referred to as keys.

The so-called *public collapsing* theories, used in some works (e.g., [10]), are presented by rewrite systems where the right-hand-side (rhs) of every rule is a ground term or a variable. Some other results assume that the rhs of any rule is a proper subterm of the lhs. A general procedure for protocol security analysis has been given in [5] for such systems, extensively using equational unification and narrowing. Rewrite systems with such a 'subterm property' have been called *dwindling* in [1], where a decision procedure was given for passive deduction—i.e., detecting secrecy attacks by an intruder *not* interacting actively with the protocol sessions. The technique used is one that combines unification and narrowing with the notion of *cap closure* modeling the evolution of the intruder knowledge. The algorithm presented in [1] was actually shown to be complete for passive deduction for a class of convergent rewrite systems called Δ-*strong*, that contains strictly the class of dwindling systems; this class contains in particular the following convergent, non-dwindling system, that we refer to as HE; it extends DY with the requirement that '*encryption distributes over pairs*':

$$\begin{aligned} p_1(x.y) &\to x & & \\ p_2(x.y) &\to y & enc(x.y, z) &\to enc(x, z).enc(y, z) \\ enc(dec(x, y), y) &\to x & dec(x.y, z) &\to dec(x, z).dec(y, z) \\ dec(enc(x, y), y) &\to x & & \end{aligned} \qquad \text{(HE)}$$

We shall refer to the equational theory defined by this system HE as *Homomorphic Encryption*, or again as HE. On protocols where encryption is based on the so-called Electronic Code Book mode (ECB)—i.e., is performed sequentially on a block decomposition of the message –, encryption can be modeled as an homomorphism on pairs. It seems worth mentioning here that, although ECB is known to be more prone to attacks than other modes such as Cipher Block Chaining (due to the fact that message blocks with the same content get encrypted exactly alike, irrespective of where these blocks are located), nevertheless, ECB still seems widely used in many commercial protocols; cf. e.g., http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesval.html

As we just mentioned, passive deduction is known to be decidable for protocols employing HE; but the problem of *active deduction* for such protocols—i.e., when the intruder is allowed to interact with the protocol steps, e.g., to forge the identity of some honest agent—has not been studied yet. For deciding active deduction modulo

any given intruder theory E, the decidability of E-unification is known to be a necessary condition (cf. e.g., [8]), and that gave us the motivation for studying HE-unification. Note that the homomorphism $enc(-, y)$ defined on terms, for any given y, admits an inverse homomorphism $dec(-, y)$ modulo HE; consequently, unification modulo HE cannot be reduced directly to unification modulo one-sided distributivity [18].

This paper is structured as follows: The needed preliminaries are given in Section 2. Unification modulo HE is shown to be decidable in Section 3. The main idea consists in reducing any given HE-unification problem into one of solving a set of 'simple' equations of the form $Z = enc(X, V)$ or $Z = dec(X, V)$, where none of the first arguments under *enc* gets split into pairs by the other equations. Solving such a set of 'simple' equations is essentially the unification problem modulo the two rules for encryption and decryption:

$$dec(enc(x, y), y) \to x$$

$$enc(dec(x, y), y) \to x$$

which form a confluent, dwindling system, so has a decidable unification problem, cf. [16]. The method we propose in this work actually combines a graph-based algorithm reasoning modulo the group structure on homomorphisms—that is specific to 'simple' HE-unification problems—with one that reasons modulo a theory for pairings. We show that *even solving 'simple' HE-unification problems* (i.e., without pairings) *is NP-complete*.

A couple of examples are given in Section 4 to illustrate our HE-unification algorithm. Section 5 presents briefly a couple of other convergent rewrite systems modeling Homomorphic Encryption; they show, in particular, that the approach we have presented here can handle asymmetric keys as well. In Section 6, we recall briefly the notion of Cap Unification, a technique that extends unification for solving a sequence of cap constraints—also called 'deducibility constraints' by some authors –, modulo any given intruder theory; such constraints model in a natural manner the steps of any given (cryptographic) protocol session that an 'intruder' can interact with, for gaining knowledge. We illustrate the technique on a small (Needham–Schroeder like) 'toy' protocol, assuming the encryption mode to be ECB-based.

This paper – which is an enhanced version of [2] – also contains an Appendix, whose aim is to show that the passive deduction problem and unification can have unrelated behaviors, in general—a fact which is in sharp contrast with the case of active deduction, as was already observed in the concluding section of [1], as well as in [12]. We prove this fact here, by showing that unification modulo general Δ-strong rewrite systems is undecidable, by reduction from a suitable version of the Modified Post Correspondence Problem (MPCP).

2 Notation and Preliminaries

As usual, Σ will stand for a ranked signature, and $\mathcal{X}$ a countably infinite set of variables. $\mathcal{T} = \mathcal{T}(\Sigma, \mathcal{X})$ is the algebra of terms over this signature; terms in $\mathcal{T}$ will be denoted as $s, t, \ldots$, and variables as $u, v, x, y, z, \ldots$, all with possible suffixes. The set of all positions on any term t is denoted as $Pos(t)$; if $q \in Pos(t)$, then $t|_q$ denotes the subterm of t at position q; and the term obtained from t by replacing the subterm $t|_q$

by any given term t' will be denoted as $t[q \leftarrow t']$; a similar notation is employed also for the substitution of variables of t by terms. We assume a simplification ordering $\succ$ on $\mathcal{T}$ that is total on ground terms (terms not containing variables). A rewrite rule is a pair of terms (l, r) such that $l \succ r$, and is represented as usual, as $l \rightarrow r$; a rewrite system is a finite set of rewrite rules. The notions of reduction and of normalization of a term by a rewrite system are assumed known, as well as those of termination and of confluence of the reduction relation defined by such a system on terms (cf. e.g., [4]). A rewrite system R is *convergent* iff the reduction relation it defines on the set of terms is terminating and confluent.

By an HE-Unification problem we mean, as usual, any given finite set $\mathcal{P}$ of equations between terms over Σ; and a solution to the problem $\mathcal{P}$ is a substitution σ such that $\sigma s = \sigma t$ mod HE, for every equation $s = t$ in $\mathcal{P}$. For proving that HE-Unification is decidable, we shall be applying several reductions to the given problem. To start with, we shall assume (via usual reasonings mod HE) that the given problem $\mathcal{P}$ is in a *standard form*, in the following sense: each of its equations to solve, modulo HE, is assumed to have one of the following forms:

$$Z = T, \quad Z = X.Y, \quad Z = enc(X, Y), \quad Z = const,$$

where the $T, X, Y, Z, \ldots$ stand for variables, and *const* is any ground constant. (If an equation in $\mathcal{P}$ is given in the form $U = dec(V, W)$, it is rewritten mod HE as $V = enc(U, W)$.) The equations in $\mathcal{P}$ of the *first* and *fourth* forms are said to be '*equalities*', equations of the second form are called '*pairings*' and those of the third form are said to be of the *enc* type. *We also assume, explicitly,* that if such a problem $\mathcal{P}$ contains two equations of the form $Z = t$, $Z' = t$ (with identical right hand sides), then $\mathcal{P}$ also contains the equality $Z = Z'$.

The second arguments of *enc*, in the equations of $\mathcal{P}$, are referred to as the *keys* or *key variables* of $\mathcal{P}$. If Y is a key variable which is also the lhs of an equation of the fourth form, i.e., the rhs is a constant, then Y as well as the constant of that equation will both be said to be a key constant of $\mathcal{P}$. Given a problem $\mathcal{P}$ in standard form, we denote by $\mathcal{X}_\mathcal{P}$ the set of its variables, and by $\mathcal{K}_\mathcal{P}$ the set of key variables and key constants of $\mathcal{P}$.

The *conjugate* of any *enc* equation $Z = enc(X, Y)$ in $\mathcal{P}$, is defined as the equation $X = dec(Z, Y)$; it will be said to be of the '*dec*' type. For every key variable/constant Y occurring in $\mathcal{P}$, let h_Y (resp. $\overline{h}_Y$) denote the homomorphism $enc(-, Y)$ (resp. $dec(-, Y)$) defined on terms. An *enc* equation $Z = enc(X, Y)$ can thus be written as $Z = h_Y(X)$, and its conjugate as $X = \overline{h}_Y(Z)$. We denote by $\mathcal{H} = \mathcal{H}_\mathcal{P}$ the finite set of all such homomorphisms associated with the key variables/constants of the problem $\mathcal{P}$. All our unification problems in the sequel will be assumed to be in standard form (unless mentioned otherwise explicitly).

The Dependency Graph of $\mathcal{P}$ We construct a graph of dependency $G = G_\mathcal{P}$ between the variables of the given problem $\mathcal{P}$: Its nodes will be the variables (or constants) of $\mathcal{P}$. From a node Z on G, there is an oriented arc to a node X on G iff the following holds:

a) $\mathcal{P}$ has an equation of the form $Z = h_Y(X)$ (resp. $X = h_Y(Z)$), for some $Y \in \mathcal{K}_\mathcal{P}$; the arc is then labeled with the symbol h_Y (resp. with $\overline{h}_Y$);
b) $\mathcal{P}$ has an equation of the form $Z = X.V$ (resp. $Z = V.X$): the arc is then labelled with p_1 (resp. with p_2).

Semantics: If G contains an edge of the form $Z \rightarrow^h X$, with $h \in \mathcal{H}$, then Z can be evaluated by applying the homomorphism h to the evaluation of X. (Note that there are *no 'equality' arcs* on the graph $G_{\mathcal{P}}$.)

3 Unification Modulo HE

Theorem 1 *Unification modulo the theory HE is decidable.*

To facilitate understanding, we present briefly the outlines of the proof. An inference procedure will be applied to the unification problem $\mathcal{P}$, given in standard form. The transformation of $\mathcal{P}$ under the rules of this procedure (named *trimming* rules) will be based on the following guiding principles:

- Perfect Encryption: If $Z = enc(X, Y) \in \mathcal{P}$ and also $Z = enc(X, Y') \in \mathcal{P}$ (resp. $Z = enc(X', Y) \in \mathcal{P}$), then $Y = Y'$ (resp. $X = X'$) must be in $\mathcal{P}$.
- Pairing is free in HE: If $Z = X.Y$ and $Z = X'.Y'$ are both in $\mathcal{P}$, then the equalities $X = X'$, $Y = Y'$ must be in $\mathcal{P}$.
- Split on Pairs: $Z = enc(X, Y) \in \mathcal{P}$ and if one of Z, X is the lhs of a pairing equation in $\mathcal{P}$, then the other must be so too.
- Irredundancy of the Dependency Graph: If Z', Z'' are two distinct nodes of $G_{\mathcal{P}}$, then $\mathcal{P}$ must not contain the equality $Z' = Z''$.

(The first principle says that, from any given message two different keys cannot generate the same encrypted message; the third says that no encrypted message may split into a pair, if the original message itself is not a pair.) Our objective is to transform the given problem into a 'trimmed' problem composed of two sub-problems which can be treated 'almost' separately: one containing only the pairings and equalities, and the other containing only the *enc* equations whose lhs variables (resp. first arguments under *enc*) are *not* splittable into pairs; the latter sub-problem will be solved by 'combinatorial' means as we shall be seeing farther down; and their solutions extend naturally as solutions for the entire problem, via combination with its equalities and pairings.

Among the four guiding principles above, the third necessitates introducing fresh 'splitting' variables, in general. For defining a measure on how deep we may need to go, for introducing such fresh variables starting from any given variable, we need the following relations on the set of variables $\mathcal{X} = \mathcal{X}_{\mathcal{P}}$ appearing in $\mathcal{P}$:

Definition 1 $U \sim V$ is the finest equivalence relation on $\mathcal{X}$ such that:

- if $U = V \in \mathcal{P}$ then $U \sim V$;
- if $U = enc(V, T) \in \mathcal{P}$ or $V = enc(U, T) \in \mathcal{P}$, for some T, then $U \sim V$;
- if $\mathcal{P}$ contains two pairings of the form $W = U.X$ and $W' = V.X'$ (or of the form $W = X.U$ and $W' = X'.V$), where $W \sim W'$, then $U \sim V$.

- We write $U \gg V$ iff there is a loop-free chain from U to V formed of $\sim$- or p_1/p_2-steps, *at least one* of them being a p_1- or p_2- step.
- For any problem $\mathcal{P}$ and for any given $Z \in \mathcal{X} = \mathcal{X}_{\mathcal{P}}$, the *sp-depth of* Z (short for *splitting depth* of Z, and denoted as $spd(Z)$) is defined as *the maximum*

number of p_1- or p_2- steps from Z to all possible $X \in \mathcal{X}$, *along the loop-free chains formed of $\sim$- or p_1/p_2-steps from Z to X.*

A ground substitution on the set $\mathcal{X} = \mathcal{X}_\mathcal{P}$ is said to be *discriminating* iff distinct key variables of $\mathcal{P}$ are assigned distinct irreducible ground terms. A discriminating solution for $\mathcal{P}$ is such a substitution that also solves $\mathcal{P}$. A fifth guiding principle is that, in order that $\mathcal{P}$ admits such a solution, there can be no directed loop with a 'non-trivial label', and formed only of $h/\overline{h}$-arcs, from any node to itself on the graph $G_\mathcal{P}$. In formal terms: Let X, Y be any two nodes on G, and α any given word over the set $\mathcal{H}$ of homomorphisms. We shall write $X \rightarrowtail^\alpha Y$ iff there is a directed path from X to Y on G, the arcs of which are labeled respectively by the homomorphisms forming the word α. The following condition gives our fifth guiding principle:

(**SNF**): For any directed loop on $G = G_\mathcal{P}$ from any node Z on G to itself, such that the labels of its arcs form a word $\alpha \in \mathcal{H}^*$, the word α must simplify to the empty word under the following set of rules:

$$h_T \overline{h}_T \to \epsilon, \quad \overline{h}_T h_T \to \epsilon, \qquad T \in \mathcal{K}_\mathcal{P}. \qquad (\mathcal{D})$$

This condition **SNF** is necessary for $\mathcal{P}$ to admit a discriminating solution: indeed, if σ is such a solution for $\mathcal{P}$, and $Z \rightarrowtail^\alpha Z$ is a non-trivial loop on G formed only of $h/\overline{h}$-arcs, it means the ground term $\sigma(\alpha)(\sigma Z)$ must normalize to $\sigma(Z)$, and that can be done only by the two rewrite rules to the bottom-left of the rewrite system HE.

Note that any non-discriminating solution to $\mathcal{P}$, i.e., one that does not assign distinct values to distinct key variables, can be seen as a discriminating solution to a *variant* of $\mathcal{P}$, obtained by 'equating some keys' by adding some further equalities to $\mathcal{P}$ (inference rule 6 below). We are in a position now to formulate our inference rules.

The Inference Rules We denote by **Eq** (resp. **Pair**, **Enc**) the set of equalities (resp. pairings, the *enc*-equations) in $\mathcal{P}$, respectively.

Rule 1. (*Perfect Encryption*)

$$a)\ \frac{\mathbf{Eq};\ \mathbf{Pair};\ \mathbf{Enc} \sqcup \{Z = enc(X, Y),\ Z = enc(V, Y)\}}{\mathbf{Eq} \cup \{V = X\};\ \mathbf{Pair};\ \mathbf{Enc} \sqcup \{Z = enc(X, Y)\}}$$

$$b)\ \frac{\mathbf{Eq};\ \mathbf{Pair};\ \mathbf{Enc} \sqcup \{Z = enc(X, Y),\ Z = enc(X, T)\}}{\mathbf{Eq} \cup \{T = Y\};\ \mathbf{Pair};\ \mathbf{Enc} \sqcup \{Z = enc(X, Y)\}}$$

Rule 1'. (*Variable Elimination*)

$$\frac{\{U = V\} \sqcup \mathbf{Eq};\ \mathbf{Pair};\ \mathbf{Enc}}{\{U = V\} \cup [V/U](\mathbf{Eq});\ [V/U](\mathbf{Pair});\ [V/U](\mathbf{Enc})}$$

Rule 2. (*Pairing is free in HE*)

$$\frac{\mathbf{Eq};\ \mathbf{Pair} \sqcup \{Z = U_1.U_2,\ Z = V_1.V_2\};\ \mathbf{Enc}}{\mathbf{Eq} \cup \{V_1 = U_1,\ V_2 = U_2\};\ \mathbf{Pair} \sqcup \{Z = U_1.U_2\};\ \mathbf{Enc}}$$

Rule 3. (*Split on Pairs*)

$$a)\ \frac{\mathbf{Eq};\ Z = Z_1.Z_2 \in \mathbf{Pair};\ \mathbf{Enc} \sqcup \{Z = enc(X, Y)\}}{\mathbf{Eq};\ \mathbf{Pair} \cup \{X = X_1.X_2\};\ \mathbf{Enc} \sqcup \{Z_1 = enc(X_1, Y),\ Z_2 = enc(X_2, Y)\}}$$

$$b)\ \frac{\mathbf{Eq};\ X = X_1.X_2 \in \mathbf{Pair};\ \mathbf{Enc} \sqcup \{Z = enc(X, Y)\}}{\mathbf{Eq};\ \mathbf{Pair} \cup \{Z = Z_1.Z_2\};\ \mathbf{Enc} \sqcup \{Z_1 = enc(X_1, Y),\ Z_2 = enc(X_2, Y)\}}$$

Rule 4. (*Occur check*)

$$\frac{\mathbf{Eq};\ \mathbf{Pair};\ \mathbf{Enc};\ \ Z \sim Z' \ and\ Z \gtrdot Z'}{FAIL}$$

Rule 4'. (*Clash with Pair*)

$$\frac{\mathbf{Eq} \sqcup \{Z = a\};\ \mathbf{Pair} \sqcup \{Z = U_1.U_2\};\ \mathbf{Enc}}{FAIL}$$

Rule 4". (*Clash with Constant*)

$$\frac{\mathbf{Eq} \sqcup \{Z = a,\ Z \sim b\};\ \mathbf{Pair};\ \mathbf{Enc}}{FAIL}$$

Rule 5. (**SNF** *Fails*)

$$\frac{\mathbf{Eq};\ \mathbf{Pair};\ \mathbf{Enc};\ \ Z \in G,\ \alpha \in \mathcal{H}^*,\ Z \rightarrowtail^{\alpha} Z,\ \alpha \not\rightarrow^*_{\mathcal{D}} \epsilon}{FAIL}$$

Rule 6. (*Equate Some Keys*)

$$\frac{\mathbf{Eq};\ \mathbf{Pair};\ \mathbf{Enc};\ \ U,\ V \text{ are keys of } \mathcal{P}}{\mathbf{Eq} \cup \{U = V\};\ \mathbf{Pair};\ \mathbf{Enc}}$$

The notation $\sqcup$ in rules 3a, b signifies *disjoint union*; it means here that the *enc*-equation in the numerator is *replaced* by the two *enc*-equations in the denominator; this *might* need the variables X_1, X_2 in rule 3a (resp. Z_1, Z_2 in rule 3b) to be fresh, but this is not mandatory. Rules 1, 1′ and 2 are referred to as "*Simplification Rules*", and rules 3*a* and 3*b* as "*Splitting Rules*". Rules 4, 4′, 4″ and 5 are called "*Failure Rules*". All these rules 1, 1′, 2, 3, 4, 4′, 4″, 5, constitute our "*Trimming Rules*".

A problem $\mathcal{P}$ in standard form is said to be *trimmed* iff none of the trimming rules is applicable. The 'Occur-Check' rule 4 is meant to eliminate easy cases of unsolvability, such as when $\mathcal{P}$ contains two equations of the form $Z = enc(X, T)$, $Z = X.Y$; similarly, rule 4′ (resp. rule 4″) eliminates unsolvable cases such as $Z = a,\ Z = X.Y$ (resp. $Z = a, X = b,\ Z = enc(X, Y)$). The graph of the current problem is kept irredundant by rule 1′. The rules 3—which might create fresh splitting variables— are to be applied only if none of the other rules 1 through 5 are applicable; more precisely:

the Failure rules are to be applied with the highest priority, followed by rules $1a, 1b, 1'$ and 2, and then by rules 3. Rules 1, 2, 3 are *don't-care* nondeterministic. Rule 6 is *don't-know* nondeterministic; it is to be applied outside the scope of the trimming rules (in particular rule $1b$), and its role is to produce variants of $\mathcal{P}$. It is easy to check that the set of solutions of $\mathcal{P}$ is equal to the union of the solution sets of the trimmed problems derived by applying *all* the inference rules to $\mathcal{P}$.

We will show shortly that such an inference procedure terminates on any problem given in standard form. Hence only finitely many trimmed problems can be derived from any problem $\mathcal{P}$ given in standard form. For proving termination, we shall be needing the following notions.

(1) Let $\mathcal{P}$ be any such given problem. We introduce a binary, infix operator '$\circ$' representing pairs (but denoted differently, to avoid confusion); and define $\mathcal{T}_p(\mathcal{P}) = \mathcal{T}_p$ as the set of all terms formed over $\mathcal{X}$, the symbol '$\circ$', and the set of all homomorphisms h_T – where T runs over all the keys of $\mathcal{P}$.

- Any pairing $X = X_1.X_2$ in $\mathcal{P}$, is seen as a rewrite rule: $X \to X_1 \circ X_2$;
- Any equation $Z = enc(X, T)$ in $\mathcal{P}$ gives rise to two rewrite rules: $Z \to^{h_T} X$, and $X \to^{\overline{h}_T} Z$.
  Rules of the former type will be called pairing rules; those of the latter type will be respectively called h-rules or $\overline{h}$-rules, with key T, and with target X for the first among them, and Z for the second. We define $\mathcal{R}_\mathcal{P}$ to be the rewrite system formed of all such rules. By a *critical configuration* in $\mathcal{R}_\mathcal{P}$, we mean any given pair of distinct rewrite rules of $\mathcal{R}_\mathcal{P}$ such that:
- both rules have the same variable $X \in \mathcal{X}_\mathcal{P}$ to their left;
- if one of them is a h-rule (resp. $\overline{h}$-rule), then the other rule must be a pairing rule or a h-rule (resp. pairing rule or a $\overline{h}$-rule);
- if both are h-rules (or $\overline{h}$-rules), then they have the same key or the same target.

The common lhs variable of a critical configuration is referred to as its *peak*.

(2) For any such given problem $\mathcal{P}$, and any given critical configuration wrt $\mathcal{R}_\mathcal{P}$ with $X \in \mathcal{X}$ as its peak, let n_X stand for the number of distinct nodes on $G_\mathcal{P}$ to which there is a loop-free, non-empty chain from X *formed only of* h- *or* $\overline{h}$- *arcs*. The *weight* of the critical configuration is then defined as the (lexicographically) ordered pair of integers $(spd(X), n_X)$.

Lemma 1 *Trimming terminates on problems given in standard form.*

Proof Given $\mathcal{P}$ in standard form, we only need to consider the inferences other than $4, 4', 4'', 5$, which—to be applied whenever applicable—would yield 'FAIL'. We define the *measure* $m(\mathcal{P})$ of $\mathcal{P}$ as the lexicographic combination of 2 components: $m_1 = m_1(\mathcal{P})$, $m_2 = m_2(\mathcal{P})$, where:

- m_1 is the number of distinct key variables appearing in $\mathcal{P}$;
- m_2 is the *multiset of weights* of all the critical configurations over $\mathcal{R}_\mathcal{P}$.

Consider now any inference on $\mathcal{P}$, by a rule other than 4, 4′, 4″ and 5. It is not hard to check that none of the inference rules will increase the *sp*-depth of a variable. We then have the following:

- Inference rule 1*b* will lower m_1.
- Inference rules 1*a*, 2—followed by applying 1′ *en bloc*—will either lower m_1, or will leave m_1 unchanged but lower m_2.
- The Splitting inference rules 3*a*, 3*b*—followed by applying 1′ *en bloc* — will leave m_1 unchanged but lower m_2.

A few words by way of justifying the last two claims: If some nodes 'become equal' under the inferences, but if the number of keys is not lower for the new problem derived, then we have the following situation:

- either some of the critical configurations have been eliminated, while the others remain unchanged;
- or some 'current' critical configurations are replaced by one or more new ones.

In the latter case, for each of the new critical configurations with Y as a peak, replacing an old one with X as a peak, we have: either $spd(Y) < spd(X)$, or $spd(Y) = spd(X)$ and $n_Y < n_X$. This follows from the definition of *spd* and of n_X, n_Y. The strict inequality on the n's holds for Splitting, because the *enc*/*dec* arc between Y and X gets cut under such an inference. The contribution of all the replacing peaks Y to the measure of the modified problem is thus strictly smaller than that of X to the measure of the initial problem. □

Example 1

(i) The following problem is not in standard form:

$$T = Z,\ Z = enc(X, Y),\ X = dec(T, Y),\ X = U.V,\ Y = Y_1.Y_2,\ Y_2 = a.$$

We first put it in standard form:

$$\mathcal{T} = Z,\ Z = enc(X, Y),\ T = enc(X, Y),\ X = U.V,\ Y = Y_1.Y_2,\ Y_2 = a.$$

Under variable elimination (rule 1′), we first get:

$$T = Z,\ Z = enc(X, Y),\ X = U.V,\ Y = Y_1.a,\ Y_2 = a;$$

which has one critical configuration: $Z \leftarrow^{\overline{h}_Y} X \rightarrow U \circ V$. Only a splitting inference is applicable (on Z); the trimmed equivalent that we get is:

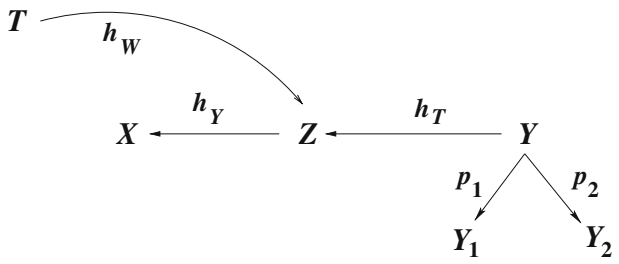
$$T = Z,\ Z = Z_1.Z_2,\ X = U.V,\ Y = Y_1.a,\ Y_2 = a,\ Z_1 = enc(U, Y),\ Z_2 = enc(V, Y).$$

(ii) The following problem:

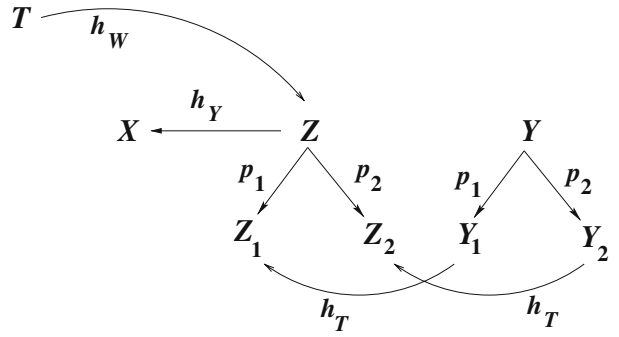
$$Z = enc(X, Y),\ Y = enc(Z, T),\ T = enc(Z, W),\quad Y = Y_1.Y_2.$$

is in standard form, but not trimmed: we have one critical configuration, namely: $Z \leftarrow^{h_T} Y \rightarrow Y_1 \circ Y_2$, with peak at Y. Now $spd(Y) = 1$, but $n_Y = 3$ (we

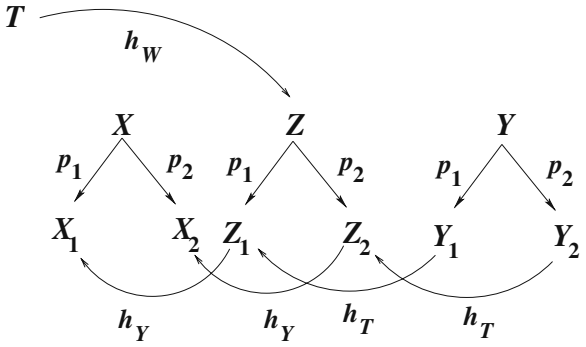
can go from Y to T, X, Z using only *enc*/*dec* arcs); so m_2 here is $\{(1, 3)\}$, and the measure $m(\mathcal{P})$ of the problem is $(3, \{(1, 3)\})$.



Trimming needs here several splitting steps. We first write $Z = Z_1.Z_2$, and replace the second *enc* equation by the 2 equations: $Y_1 = enc(Z_1, T)$, $Y_2 = enc(Z_2, T)$; we get a problem with two critical configurations, both with peak at Z, $spd(Z) = 1$ and $n_Z = 2$; so the measure is lowered to $(3, \{(1, 2), (1, 2)\})$: The evolution of the dependency graph of the problem under splitting is illustrated below (where, for readability, we have not put in the $\overline{h}$-arcs).



Next, we write $X = X_1.X_2$ and replace the first *enc* equation by $Z_1 = enc(X_1, Y)$, $Z_2 = enc(X_2, Y)$, and get a problem with measure $(3, \{(1, 1)\})$:



Finally, we write $T = T_1.T_2$ and replace the last *enc* equation by: $T_1 = enc(Z_1, W)$, $T_2 = enc(Z_2, W)$. We get the following trimmed equivalent, with measure (3, {(0, 0)}:

$$\begin{aligned} &Z_1 = enc(X_1, Y), \qquad Z_2 = enc(X_2, Y),\\ &Y_1 = enc(Z_1, T), \qquad Y_2 = enc(Z_2, T),\\ &T_1 = enc(Z_1, W), \qquad T_2 = enc(Z_2, W),\\ &Y = Y_1.Y_2,\ Z = Z_1.Z_2, \qquad X = X_1.X_2,\ T = T_1.T_2. \end{aligned}$$

Remark 1

(i) The number of equations in a trimmed equivalent of a problem $\mathcal{P}$ given in standard form—derived at the end of the inference procedure, when it does not FAIL—can be exponential wrt the number of initial equations in $\mathcal{P}$; a typical illustrative example is the following:

$$\begin{aligned} &X_1 = enc(X_2, U1) \qquad X_{11} = enc(X_{12}, U2) \qquad X_{111} = enc(X_{112}, U3)\\ &X_1 = X_{11}.X_{12} \qquad X_{11} = X_{111}.X_{112} \qquad X_{111} = X_{1111}.X_{1112} \end{aligned}$$

(Intuitively: Splitting here fills out a full binary tree with the fresh variables created.)

(ii) A key variable of a problem $\mathcal{P}$ (in standard form), can also be a 'message variable', as the Y, T of Example 1.(ii) above. Note that *as a key* it will remain unaffected under trimming, even when it gets split as a 'message'. So, the number of keys of a problem remains unaffected under trimming.

A problem $\mathcal{P}$ is said to be *admissible* iff it is a trimmed equivalent of itself. Such a problem $\mathcal{P}$ is divided into two sub-problems: one containing only the pairings and equalities of $\mathcal{P}$, and the other containing only its *enc* equations; this latter subproblem will be referred to as the *kernel* of $\mathcal{P}$. *A problem $\mathcal{P}$ will be said to be* simple *iff it is its own kernel*, i.e., if $\mathcal{P}$ is admissible and has no pairings.

The graph of an Admissible Problem The dependency graph $G_{\mathcal{P}}$ of such a problem $\mathcal{P}$ is irredundant: if X, V are two distinct nodes on $G_{\mathcal{P}}$, then $X = V$ is *not* an equality in $\mathcal{P}$. An admissible problem $\mathcal{P}$ will actually be seen as a combination of its kernel $\mathcal{P}'$, and its 'other' subproblem $\mathcal{P}''$ consisting only of the pairings and equalities; and its graph $G = G_{\mathcal{P}}$ as a 'join' of the dependency graph $G' = G_{\mathcal{P}'}$ of its kernel $\mathcal{P}'$—each arc of which is labeled with an h or an $\overline{h}$—and the dependency graph G'' of the subproblem $\mathcal{P}''$, each arc of which is labeled with either p_1 or p_2.

Lemma 2

(i) *From any given node Z on G' there is* at most one loop-free path *on G' to any other given node X on G'.*

(ii) *For any given word α over $\{p_1, p_2\}$, and any given node Z on G'', there is* at most one directed path *outgoing from Z that is labeled by α.*

Proof Assertion (i) follows from **SNF**, and assertion (ii) from Inference rule 2, and the fact that the dependency graph is irredundant (rule 1b). □

Definition 2 Between the variables of an admissible problem $\mathcal{P}$, we define a relation called *key-dependency* and denoted as $\succ_k$:

- $Z \succ_k X$ iff $Z \neq X$, and there is a directed path from Z to X on the graph $G_{\mathcal{P}}$ that contains an arc labeled with $h_{X'}$ or $\overline{h}_{X'}$, where $X' = X$ or $X' \gg X$.

An admissible problem $\mathcal{P}$ and its graph $G_{\mathcal{P}}$ are said to satisfy the condition **NKDC**—short for '*No-Key-Dependency-Cycle*', *iff the following two conditions hold*:

(i) The graph $G = G_{\mathcal{P}}$ does not contain any node X such that $X \succ_k^+ X$, where $\succ_k^+$ is the transitive closure of the relation $\succ_k$.
(ii) If the graph G' of the kernel $\mathcal{P}'$ of $\mathcal{P}$ contains a constant node a, then a is $\succ_k$-minimal on its connected component.

These relations play a key role in our method for solving a trimmed problem.

3.1 Discriminating Solutions for Admissible Problems

Let $\mathcal{P}$ be any given admissible problem. The case where the kernel of $\mathcal{P}$ is empty—i.e., when $\mathcal{P}$ contains only equalities and pairings—is trivial, as is easily checked; we shall therefore assume henceforth that $\mathcal{P}$ has a non-empty kernel $\mathcal{P}'$. Our objective in this section is to prove the following proposition:

Proposition 1 *An admissible problem admits a discriminating solution if and only if it satisfies NKDC.*

NKDC is Necessary In this paragraph, θ stands for a discriminating substitution on the set $\mathcal{X}$. For any $X \in \mathcal{K}_{\mathcal{P}}$, $\tilde{h}_X$ stands for either h_X or its conjugate $\overline{h}_X$; and $C, C', \ldots$, referred to as contexts, stand for words over the $\tilde{h}_X$. For any term t, $|t|$ stands as usual for its size. If θ solves $\mathcal{P}$, then it must also solve $\mathcal{P}'$. Now, we know that solving $\mathcal{P}'$ amounts to solving the unification problem modulo the convergent system R formed of the following two rules:

$$(R): \quad enc(dec(x, y), y) \to x, \qquad dec(enc(x, y), y) \to x.$$

Lemma 3 *Assume that $Y = \tilde{h}_X(C(t))$ for some context C, and term t. Then, for any R-normalized ground substitution θ, we have that $\theta(Y)$ is either a subterm of $\theta(C(t))$ or $\theta(X)$ is the outermost key of $\theta(Y)$.*

Proof

Case i) Where the encryption (or decryption) key $\theta(X)$ gets cancelled by the outermost decryption (or encryption) key of the term $\theta(C(t))$: in this case, $\theta(Y)$ must be a subterm of the term $\theta(C(t))$.
Case ii) Where the key $\theta(X)$ does not get cancelled, by the key just below in the term $\theta(C(t))$: in this case $\theta(X)$ will remain the outermost key of $\theta(Y)$. □

Lemma 4 *Let X, Y be two different nodes on the graph of the kernel $\mathcal{P}'$ of some admissible problem $\mathcal{P}$; and suppose the (unique) loop-free, directed path on $G' = G_{\mathcal{P}'}$ from Y to X contains an arc labeled with an $\tilde{h}_{X'}$, where $X' = X$ or $X' \gtrdot X$. Then, for any discriminating R-normalized ground substitution θ, we have: $|\theta(Y)| > |\theta(X)|$.*

Proof We observe to start with, that $\mathcal{P}'$ being itself admissible, remains unmodified under inference rule 1; so the following holds: Suppose $\tilde{h}_T, \tilde{h}_{T'}$ label two successive arcs on any loop-free path on G', then:

- either $T = T'$: in which case, the two successive arcs have to be both *enc*-arcs, or both *dec*-arcs;
- or $T \neq T'$: in which case θ assigns different ground normal terms to T and T'.

In either case, it follows that the two keys $\tilde{h}_{\theta(T)}, \tilde{h}_{\theta(T')}$—images under θ of the two successive labels—cannot cancel each other.

Now, by assumption, between the two nodes X, Y on G', we have a relation of the form $Y = C'(\tilde{h}_{X'}(C(X)))$, for some contexts C' and C, where we may assume wlog that the context C contains no homomorphism of the form $\tilde{h}_{X'}$ such that $X' = X$ or $X' \gtrdot X$. So we have $\theta(Y) = \theta(C')(\tilde{h}_{\theta(X')}(\theta(C)(\theta(X)))$. From what we observed above (namely: the successive keys in the term to the right cannot cancel each other), and from Lemma 3, it follows that the assertion of the lemma needs to be checked only in the case where the contexts C, C' are empty; i.e., when $\theta(Y) = \tilde{h}_{\theta(X')}(\theta(X))$. But then the term to the right is R-irreducible, due to our assumption on X'. We are done. □

Corollary 1 *Let $\mathcal{P}$ be any admissible problem on which our inference procedure does* not *fail; and suppose the graph $G' = G_{\mathcal{P}'}$ of its kernel $\mathcal{P}'$ contains a* constant *node, for some ground constant a. Then a is minimal for the relation $\succ_k$ on G'.*

Proof Immediate from the preceding Lemma. □

Lemma 5 *Let $\mathcal{P}$ be any admissible problem, that is solvable; and suppose the graph $G_{\mathcal{P}}$ of $\mathcal{P}$ does* not *satisfy the criterion NKDC; then there is* no *discriminating solution for the kernel $\mathcal{P}'$ of $\mathcal{P}$, so* no *discriminating solution for the problem $\mathcal{P}$.*

Proof Suppose $\mathcal{P}$ admits a discriminating solution θ, but $G_{\mathcal{P}}$ does *not* satisfy NKDC; constants are $\succ_k$-minimal by Corollary 1, so this means that there are two nodes Y, Z on the graph $G_{\mathcal{P}'}$ of its kernel $\mathcal{P}'$ such that:

- some arc on the (unique) path on $G_{\mathcal{P}'}$ from Y to Z uses a Y' as key, where $Y' = Y$ or $Y' \gtrdot Y$;
- and also that some arc on the (unique) path from Z to Y (which has to be the reverse of the path from Y to Z, cf. Lemma 2), uses a Z' as key, where $Z' = Z$ or $Z' \gtrdot Z$.

But then, by Lemma 4 above, we would have $|\theta(Y)| > |\theta(Z)| > |\theta(Y)|$—absurd. □

It follows from Lemma 5, and Corollary 1, that the two conditions for NKDC are indeed necessary for an admissible problem to have a discriminating solution.

Example 2 Consider the following (simple) problems:

(i) $\mathcal{P}_1$: $Y = enc(Z, X), X = enc(Z, Y)$
(ii) $\mathcal{P}_2$: $Z = enc(X, X), Z = dec(T, T)$
(iii) $\mathcal{P}_3$: $U = enc(X, Z), Z = enc(U, Y), Y = enc(U, X)$

(i) $\mathcal{P}_1$ does not admit any discriminating solution: Indeed we have $Y \succ_k X \succ_k Y$, so, if there is a solution, it must assign the same value to X, Y; thus, if the keys are to be unequal, then $\mathcal{P}_1$ would be unsolvable; or else, we could have guessed the key equality $X = Y$ (inference rule 6), and reduced the problem to one single equation $X = enc(Z, X)$, which is solvable as $Z = dec(X, X)$.
(ii) Problem $\mathcal{P}_2$ is unsolvable: First, we have $X \succ_k T \succ_k X$, so $\mathcal{P}_2$ does not admit any discriminating solution; if the keys are to be unequal, one deduces then that there is no solution. On the other hand, if we had guessed $X = T$, the problem to solve would reduce to: $Z = enc(X, X),\ Z = dec(X, X)$, for which there can be no solution at all modulo the 2-rule system R; inference rule 5 ('SNF fails') applies, leading to failure.
(iii) No discriminating solution is possible for $\mathcal{P}_3$, since $X \succ_k Z \succ_k Y \succ_k X$. And guessing an equality on the keys, such as e.g., $Y = Z$, would transform the problem into one of the two problems just studied. □

NKDC is Sufficient Let $\mathcal{P}$ be an admissible problem (with a non-empty kernel), satisfying NKDC. We propose then a method for constructing a discriminating solution, assuming that the graph $G_{\mathcal{P}}$ is connected (the general case follows). The idea is easily understood on an example. First a definition.

Definition 3 Let Γ be any connected component on the graph $G_{\mathcal{P}}$ of $\mathcal{P}$.

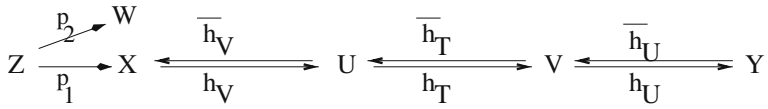
i) A *base-node* on Γ, for $\mathcal{P}$ and its kernel $\mathcal{P}'$, is any node $V_0 \in \Gamma$ that is minimal for the key-dependency relation $\succ_k^+$.
ii) An *end-node* for $\mathcal{P}$ on Γ is any node X such that:
   – there is an incoming path at X each arc of which is labeled with either p_1 or p_2;
   – there is no outgoing arc from the node X.

On any given component Γ, there may be several base-nodes and end-nodes; note also that *there may not be any end-node*.

Example 3 Consider the following trimmed problem:

$$(\mathcal{P}) : Z = X.W,\ X = enc(U, V),\ U = enc(V, T),\ V = enc(Y, U)$$

whose kernel $\mathcal{P}'$ is formed of the three *enc*-equations. Its graph is connected, and there is a single maximal (loop-free) path between the nodes X and Y:



This graph satisfies NKDC: indeed, we only have two key-dependencies $X \succ_k V$ and $Y \succ_k U$; there are no key-dependency cycles, and both U and V are *minimal* for the relation $\succ_k$. So, either U or V can be chosen as a 'base-node'; W is the only 'end-node'. Thus, if we take U to be the base-node, the following substitution is a discriminating solution for $\mathcal{P}$:

$$V = \overline{h}_T(U),\ Y = \overline{h}_U(V) = \overline{h}_U\overline{h}_T(U),\ X = h_V(U),\ Z = X.W$$

with U, T and W arbitrary. The value assigned by this solution to any given node (or variable) is obtained by 'propagating' the values assigned to the chosen base-node and end-node, along the unique paths from the given node to these latter nodes; propagation is done by using the homomorphisms (resp. pairings) which label the arcs of these unique paths.

Before we proceed to establish the principal result of this paragraph (Lemma 7 below), we first prove an auxiliary result whose role is technical.

Lemma 6 *If $\mathcal{P}$ is admissible and solvable, then no connected component of the graph G' of its kernel $\mathcal{P}'$ can contain two distinct constant nodes.*

Proof Suppose a, b are two distinct constant nodes on some connected component of G'. Then we have a path from a to b formed only of *enc*- and/or *dec*- arcs. So, we would have $a \sim b$ under the relation '$\sim$' (cf. Definition 1); and inference rule $4''$ (Clash with Constant) would have led to failure. □

Lemma 7 *If $\mathcal{P}$ is admissible and satisfies NKDC, then $\mathcal{P}$ admits a discriminating solution.*

Proof We show that the kernel of $\mathcal{P}$ admits a discriminating solution. (Such a substitution, under a condition of minimality, can be extended as a solution to the entire problem, as illustrated in the Example above, via propagation on every connected component of $G_{\mathcal{P}}$. For a formal algorithm, see Section 3.2 below.) So we assume $\mathcal{P}$ itself to be simple, and $G_{\mathcal{P}}$ to be connected.

On every given connected component Γ of the graph $G = G_{\mathcal{P}}$ of $\mathcal{P}$, choose some base-node V; if Γ contains a constant node, choose that as the base-node. (Note that a constant node has to be $\succ_k$-minimal on Γ by Corollary 1; also, by Lemma 6, Γ cannot contain more than one constant node.) Then, for any given node X on Γ, we solve for X by propagating to X any value v that is assignable to the chosen base-node V: i.e., we set $X = \alpha_{XV}(v)$, where α_{XV} is the word over $\mathcal{H}$ labeling the (unique) loop-free path from X to the base-node V. Such an assignment is sound; i.e., the term assigned to X is well-defined: indeed, none of the arcs along this path can be labeled with a h_X or $\overline{h}_X$: otherwise we would have $V \succ_k X$, and V wouldn't be $\succ_k$-minimal on Γ.

Note that if T is any key of $\mathcal{P}$ which is *not* also a node on $G_{\mathcal{P}}$, then T is 'unaffected' by such a substitution, i.e., it keeps its symbolic (or constant) value. Note also that if X, Y are any two distinct variables not 'made equal by any equalities' of $\mathcal{P}$, then they must correspond to distinct nodes of $G_{\mathcal{P}'}$, then (by admissibility), the words labeling the paths from X, Y to the base-node V must be distinct, so $\alpha_{XV}(v) \neq \alpha_{YV}(v)$ by

Perfect Encryption. We deduce that the substitution constructed is indeed a discriminating solution. □

To conclude this subsection, we observe that if σ is a non-discriminating solution for $\mathcal{P}$, then there is a variant $\mathcal{P}_1$ of $\mathcal{P}$, obtained by applying Inference Rule 6 to equate some further keys (possibly followed by further trimming rules) as appropriate, such that σ defines a discriminating solution for $\mathcal{P}_1$.

3.2 Solving a Problem in Standard Form

We can formulate now a non-deterministic decision procedure for solving any HE-unification problem, given in standard form. For that, we need the notion of a minimal discriminating solution for the kernel of a trimmed problem (condition needed for propagation as solution to the entire problem):

Definition 4 Let $\mathcal{P}$ be a HE-unification problem in trimmed form, $\mathcal{P}'$ its kernel, and σ, τ two discriminating solutions for $\mathcal{P}'$. We define $\sigma \gg \tau$ iff:

- $Dom(\sigma) \supsetneq Dom(\tau)$, and
- $\exists$ variables Z, X of $\mathcal{P}$ such that $Z \gtrdot X$ and $Z \in Dom(\sigma) \setminus Dom(\tau)$.

σ is said to be a *minimal discriminating solution* for the kernel $\mathcal{P}'$ iff it is minimal for the strict relation $\gg$.

A minimal discriminating solution for the kernel $\mathcal{P}'$, of a trimmed problem $\mathcal{P}$, thus does not instantiate any key variable that is not also a node on $G_{\mathcal{P}'}$.

Example 4 The following problem is in trimmed form :

$$V = V_1.V_2,\ V_1 = enc(W_1, V),\ V_2 = enc(W_2, V).$$

Its admissible kernel is formed of the two *enc*-equations above; its graph has two connected components, respectively with V_1 and V_2 as base-nodes (there are no end-nodes). Among the following two *discriminating solutions for this kernel*, constructed as described in the previous lemma:

$$\alpha : W_1 = dec(V_1, V),\ W_2 = dec(V_2, V),\ V_1 = a,\ V_2 = b\,,$$
$$\tau : W_1 = dec(V_1, V),\ W_2 = dec(V_2, V),\ V_1 = a,\ V_2 = b\,,\ V = c$$

only α is minimal: we have $\tau \gg \alpha$.

The Algorithm $\mathcal{A}$ *Given:* An HE-unification problem $\mathcal{P}$, in standard form;
$G =$ the dependency graph for $\mathcal{P}$.

1a. Non-deterministically generate a non-Failing trimmed equivalent of $\mathcal{P}$.
1b. Replace $\mathcal{P}$ by the trimmed equivalent thus obtained; set
$\mathcal{P}' =$ the kernel of $\mathcal{P}$; $G' =$ the sub-graph of $G_{\mathcal{P}}$ for $\mathcal{P}'$.
2a. Check for the criterion NKDC on every connected component of G';
2b. If NKDC is unsatisfied on some component, exit with 'Fail';

3a. On each connected component Γ of G', choose a *base-node* V_Γ for $\mathcal{P}'$: if there is a constant node choose that as base-node, otherwise choose any node that is minimal for the relation $\succ_k^+$;
3b. Build a *minimal* discriminating substitution for the variables on each component, derived via propagation from V_Γ to each of them (as described in the previous lemma). Let σ' be the substitution, solution for $\mathcal{P}'$, thus obtained.
4a. On each connected component of G, choose all the end-nodes (if any).
4b. To the variables of the equalities and pairings of G that are not in G', assign the values deduced (via propagation) from σ', and the values assigned to the end-nodes (if any); return $\sigma =$ substitution thus obtained, as solution to $\mathcal{P}$.

Note that the Step 4b of the algorithm assigns a *unique, well-defined, term modulo HE*, to the variables of G which are not in G': indeed $\mathcal{P}$ being trimmed, we have the following:

- No two distinct pairing equations of $\mathcal{P}$ can have the same variable to the left; and the variable to the left of any pairing equation is *not* a node on the (sub)graph $G_{\mathcal{P}'}$.
- Variables to the left or to the right of the equalities of $\mathcal{P}$ have a unique representative node on the graph.

Proposition 2 *The algorithm $\mathcal{A}$ is sound and complete.*

Proof Given $\mathcal{P}$ in standard form, and $\mathcal{P}_1$ a problem derived from $\mathcal{P}$ by applying $\mathcal{A}$, let σ_1 be a solution for $\mathcal{P}_1$; one can then deduce from σ_1 a solution σ for $\mathcal{P}$; so $\mathcal{A}$ is sound. The algorithm $\mathcal{A}$ is complete as well: this follows from the fact that NKDC is a necessary condition for any simple problem to admit a discriminating solution; and on the other hand, if there is a non-discriminating solution σ for $\mathcal{P}$, then, following $\mathcal{A}$, we can derive a trimmed variant of $\mathcal{P}$ (after an inference based on Rule 6), for which σ gives a discriminating solution. □

Proposition 3 *Unification modulo HE, based on algorithm $\mathcal{A}$, is NP-hard and is in EXPTIME wrt the number of initial equations, for any problem $\mathcal{P}$ given in standard form.*

Proof We first observe that an upper bound $N(\mathcal{P})$ for the number of equations generated by trimming $\mathcal{P}$ can be given as follows: Let Variant($\mathcal{P}$) be the set of all 'variants' of $\mathcal{P}$ obtained by adding some further equalities between the keys; let m be the sup of the number of equations in all these variants, and let d stand for the sup of the sp-depths of the variables in these variants. Then $N(\mathcal{P}) \leq m\, 2^d$.

Once $\mathcal{P}$ is trimmed, the algorithm $\mathcal{A}$ runs on its kernel $\mathcal{P}'$ in polynomial time wrt the number of equations in $\mathcal{P}'$: indeed, checking for the NKDC criteria on any component of $G_{\mathcal{P}'}$ can be done in polynomial time wrt the number of nodes on the graph $G_{\mathcal{P}'}$; and solving for $\mathcal{P}$ in terms of the solutions for $\mathcal{P}'$ is also done in polynomial time. We therefore get an EXPTIME upper bound for the algorithm $\mathcal{A}$.

The NP lower bound follows from our next Proposition, where we actually prove a more precise statement. □

Proposition 4 *Solving simple HE-unification problems is NP-complete.*

Proof We just saw that solving simple problems is in NP. So, we need only to prove the NP lower bound; that is done by reduction from the following so-called *Monotone 1-in-3 SAT* problem:

- Given a propositional formula without negation, in CNF over 3 variables, check for its satisfiability under the assumption that *exactly* one literal in each clause evaluates to true.

This problem is known to be NP-complete [17]. Now consider the simple problem (without pairings) derived from the following unification problem over the 2-rule system R, involving 3 variables x_1, x_2, x_3:

$$dec(enc(dec(enc(dec(enc(a, b), x_1), b), x_2), b), x_3) =^? dec(enc(a, b), c).$$

Obviously, solving this problem amounts to saying that exactly one of the three variables x_1, x_2, x_3 is assigned the term c. □

Remark 2 It turns out that the Tiden-Arnborg algorithm [18] for unification modulo one-sided distributivity takes exponential time in the worst case [15]. (Note: This is in contradiction with an assertion of [18], where a polynomial complexity estimate is given for this unification problem.)

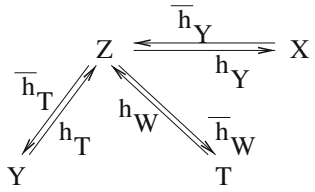
4 Illustrative Examples

The solutions for a problem $\mathcal{P}$ that the algorithm $\mathcal{A}$ returns, are substitutions that are built "lazily" (in its steps 3a through 4b); i.e., the variables of $\mathcal{P}$ get instantiated only if and when needed, as is shown in Example 5 below.

Example 5 Consider the following problem:

$$(\mathcal{P}') : Z = enc(X, Y), Y = enc(Z, T), T = enc(Z, W).$$

The problem is simple, and its dependency graph is connected:



The graph does satisfy NKDC: the key-dependency relations are $X \succ_k Y \succ_k T$; so, T is the only base-node here. We solve for Z and Y, along the path from Y to T: namely $Y \rightarrow^{h_T} Z \rightarrow^{\overline{h}_W} T$; choosing arbitrarily T, W we get $Z = \overline{h}_W(T), Y = h_T\overline{h}_W(T)$ as solutions for Z, Y; and for the variable X, connected to this path at Z, we deduce get $X = \overline{h}_Y(Z) = \overline{h}_Y\overline{h}_W(T)$. (Note: the base-node T has not been assigned any specific term here.)

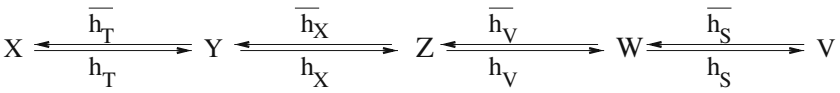
Suppose now, the problem $(\mathcal{P}')$ is the kernel of a non-simple problem, e.g., $(\mathcal{P}) : Z = enc(X, Y), Y = enc(Z, T), T = enc(Z, W), X = a$. Then, for the above

solution for its kernel to be valid, we need to check if $a = \overline{h}_Y \overline{h}_W(T)$ holds; this can be done by instantiating T, now, as $T = h_W h_Y(a)$.

Example 6 The following simple problem is unsolvable :

$$X = enc(Y, T), Y = enc(Z, X), Z = enc(W, V), W = enc(V, S)$$

Indeed, its graph (is connected and) fails to satisfy NKDC:



Indeed we have $X \succ_k V \succ_k X$. So, no discriminating solution can exist; on the other hand, it is easy to check that, no matter which keys are 'made equal' (via inference rule 6), NKDC will continue to fail for the variant obtained.

5 Other Systems for Modeling Homomorphic Encryption

(a) HE_1 = A convergent system for homomorphic encryption, where decryption is modeled as "encryption with the inverse key".

$$p_1(x.y) \to x \qquad enc(enc(x, y), g(y)) \to x$$
$$p_2(x.y) \to y \qquad enc(enc(x, g(y)), y) \to x$$
$$enc(x.y, z) \to enc(x, z).enc(y, z)$$

The method we presented above, for HE, works unchanged for this system HE_1: indeed, all we need is the fact that every homomorphism arc on the dependency graph of a problem admits an inverse homomorphism arc. It follows that the method presented in this paper can be easily adapted to handle asymmetric keys.

(b) The following rewrite system HE_2 assumes that *dec* is *only a left-inverse for enc*, and not a right-inverse as well.

$$p_1(x.y) \to x \qquad dec(enc(x, y), y) \to x$$
$$p_2(x.y) \to y \qquad enc(x.y, z) \to enc(x, z).enc(y, z)$$

HE_2 is non-convergent, but can be made convergent by adding a single *schematized* rewrite rule.[1] The method we gave above for the system HE cannot be made to work for the system HE_2. However, *Active Deduction* modulo HE_2 can be shown to be decidable, via an Inference procedure for solving a set of constraints known as Cap constraints, based on a notion of *Cap Unification*, cf. [3] (the definition of this notion is briefly recalled in the next section, along with a small illustrative

[1] Alternatively we can show that the other three rules are confluent and terminating modulo the distributivity rule.

example). It follows that unification modulo HE_2 is decidable too, as was observed in the Introduction.

6 Extending HE-Unification for Analyzing Protocols Under ECB

Our goal here is to briefly illustrate a technique for *extending* unification modulo any given intruder theory $\mathcal{E}$ into one known as *cap unification* modulo $\mathcal{E}$, for modeling and analyzing cryptographic protocols. We shall be doing this for the theory HE, so assuming that the encryption mode is ECB. (Our presentation here is an adaptation from a more complete and detailed development, given in [3].)

Typically, any given protocol step is seen as a pair $(\{t_1, \ldots, t_n\}, t)$ referred to as a *deduction rule*, where the t_i's and t are all terms, over some suitable signature; we shall denote such a rule as $\{t_1, \ldots, t_n\} \blacktriangleright t$, with the following intended semantics: if σ is a substitution such that the terms $t_i\sigma$, $1 \leq i \leq n$, are already "part of the intruder knowledge", then (s)he can deduce the term $t\sigma$.

Protocol rules are used to simulate a protocol step in a protocol session. It suffices to consider the analysis of *one protocol session,* since the case of several sessions can be reduced to that of a single session, via standard techniques [10]. Thus, every protocol rule is used only once; and when the variables of a rule are instantiated, their values are propagated to all the other rules; therefore, the variables of a protocol rule are often said to be 'rigid' variables.

The sequence of steps in any given protocol session is then transformed into an ordered set of constraints, called Cap Constraints. (Note: these have also been called "Deducibility constraints" in many related works; cf. e.g. [6, 14].). Cap Unification modulo $\mathcal{E}$ is a technique for solving the set of all Cap Constraints thus derived, from the steps of the protocol session that the 'intruder' can interact with (or exploit), for gaining further knowledge. A few definitions are appropriate before we proceed.

Definition 5

(i) Let S be a given set of terms over the signature of HE. Then the Cap Closure of S is the set of terms denoted as $Cap(S)$, and defined as follows:

- $S \subseteq Cap(S)$
- If $t_i \in Cap(S)$, for all $1 \leq i \leq n$, and f is a symbol in HE of arity n, then $f(t_1, t_2, \ldots, t_n) \in Cap(S)$.
  (It is assumed here, that if f is p_1 or p_2, then its argument t must be a pair.)

(ii) A *cap constraint* over HE is a constraint written in the form $S \rhd_{HE} t$, where S is a set of terms, and t is a term. It is *solvable* iff there exists $s \in Cap(S)$, and a substitution σ such that $s\sigma = t\sigma$ mod HE. We call σ a solution of $S \rhd_{HE} t$.

Let $\Gamma = \{S_i \rhd_{HE} t_i,\ 1 \leq i \leq n\}$ be any given set of cap constraints. A substitution σ is said to be *solution* for Γ iff σ is a solution for *every* cap constraint in Γ.

An Illustrative Example We present here a small (Needham–Schroeder like) toy protocol, and an attack on it, to illustrate how Cap Unification captures attacks. (In [3], it is also shown how to transform a sequence of randomly chosen steps of a given

protocol session, into a sequence of cap constraints satisfying the property that any variable that appears in the LHS of some constraint, has already been 'deduced'; i.e., has already appeared in the RHS of an earlier constraint in the sequence).

A, B, I	principals
na, nb	(fresh) nonces
K'	(symmetric) longterm key shared between A and B.
k	a session key that A sends to B, paired with na and encrypted with K'.

1. $A \rightarrow B : enc(k.na, K')$
2. $B \rightarrow A : enc(na.nb, K')$
3. $A \rightarrow B : nb$

We observe that:

B's point of view on Step 2 is	If Receive: $enc(x.y, K')$, then Send: $enc(y.nb, K')$
A's point of view on Step 3 is	If Receive: $enc(na.z, K')$, then Send: z

The following is an attack by intruder I, impersonating B, and intercepting the message sent at step 1:

1'. $A \rightarrow I(B) : enc(k.na, K')$
2'. $I(B) \rightarrow A : enc(na.k, K')$
3'. $A \rightarrow I(B) : k$

This attack is based only on homomorphism, and is deducible by cap unification: (Knowing how the protocol is specified) Intruder I needs only to solve the following cap constraints modulo HE:

$$\{enc(k.na, K')\} \rhd_{HE} enc(na.z, K')$$
$$\{enc(k.na, K'), z\} \rhd_{HE} k$$

$\sigma : z = k$ is a solution. Indeed, let $s = p_2(enc(k.na, K')) \,.\, p_1(enc(k.na, K'))$; then the term s is in the cap closure of intruder's initial knowledge, namely the message (s)he intercepted in step 1′; and we have $\sigma(s) =_{HE} \sigma(enc(na.z, K'))$.

7 Conclusion

We have presented a unification procedure for the theory HE modeling homomorphic encryption, with the hope that it can be used for analyzing cryptographic protocols employing the ECB encryption mode. (As observed in the Introduction, despite being vulnerability-prone, ECB is still being used in several commercial protocols.) On the other hand, many protocols employ other encryption modes, such as Cipher Block Chaining using the AC-operator XOR (cf. [9, 11]), to overcome the vulnerability inherent in ECB; it is possible to model such a CBC in terms of a finite, convergent, AC-rewrite system, cf. the concluding section of [3]. It would be of interest to propose a procedure based on unification and cap unification modulo such a rewrite system, that would be complete for active (or at least passive) deduction.

We also believe that the techniques we have employed in this work are general enough to be customizable for some other algebraic properties; and that this work would be a first step towards deriving a unification-based algorithm for protocol security, when the crypto-operators satisfy additional algebraic properties besides

DY (cf. e.g., [7, 13]). It seems in particular possible to adapt the inference procedure given here into one that is complete for unification modulo a simple rewrite system modeling a cipher block chaining technique using *no* AC-operators.

Acknowledgements Our thanks to the anonymous referees for their comments, which have helped us clarify some of the technical details.

Appendix

The decision procedure given in [1], based on forming the cap closure of terms, was shown to be complete for passive deduction modulo certain classes of convergent intruder systems. This holds in particular for the so-called Δ-strong systems, of which the system HE studied above is a special case. Our objective here is to show that unification modulo general Δ-strong systems is undecidable; we get as a corollary that passive deduction and unification may be unrelated. We first recall some notational preliminaries from [1].

We assume given a proper subset $\mathcal{P}$ of symbols of the given signature Σ—referred to as the set of *public* symbols – such that $\Sigma \setminus \mathcal{P}$ contains at least one ground constant; the symbols of $\Sigma \setminus \mathcal{P}$ will be said to be *private*. Any convergent rewrite system R, such that the top-symbol of the lhs of every rule in R is a public symbol, will be said to be an intruder theory.

Suppose R_0 is any given convergent intruder system. An n-ary public symbol f is said to be *transparent* in/for R_0, or R_0-*transparent*, if and only if, for all $x_1, \ldots, x_n$, there exist 'context-terms' (with a single hole) $t_1(\diamond), \ldots, t_n(\diamond)$ such that the following holds: $t_i[\diamond \leftarrow f(x_1, \ldots, x_n)] \rightarrow^*_{R_0} x_i$, for every $1 \leq i \leq n$. For instance, the public function '.' ("pair") is transparent for the system: $p_1(x.y) \rightarrow x$, $p_2(x.y) \rightarrow y$, where p_1 and p_2 are both public. We shall consider public constants as transparent for any intruder system R_0. A public function symbol is R_0-*resistant* (or simply *resistant* if R_0 is clear from the context) iff it is not R_0-transparent. Private functions will be considered as resistant for any intruder system R_0. By definition, an R_0-resistant term is one whose top-symbol is R_0-resistant.

Let now R be any given convergent intruder theory, containing a dwindling subsystem. We assume given a simplification ordering $\succ$ on terms containing R, which is precedence based (such as *rpo* or *lpo*), and is such that every private symbol is greater than any public symbol, under $\succ$. Let Δ be a subsystem consisting of (some of the) dwindling rules of R. A rewrite rule $l \rightarrow r \in R$ is said to be Δ-strong, wrt the given simplification ordering $\succ$, if and only if every Δ-resistant subterm of l is greater than r wrt $\succ$. The intruder theory R is said to be Δ-strong wrt $\succ$ if and only if every rule in $R \setminus \Delta$ is Δ-strong wrt $\succ$.

The rewrite system HE studied above is Δ-strong, if Δ is taken to be the subsystem formed of the four dwindling rules of HE, and every function symbol is considered public: indeed the lpo ordering built over the precedence $enc > dec > . > p_1 > p_2$ is ground total and contains HE; the symbols '*dec*' and '*enc*' are both Δ-resistant.

Proposition 5 *Unification modulo general, convergent, Δ-strong theories is undecidable.*

Proof The proof is by reduction from a restricted version of the modified Post Correspondence Problem (MPCP).

Let $\Sigma = \{a, b\}$, and let $P = \{(\phi_i, \psi_i) \mid i = 1, \ldots, n\} \subseteq \Sigma^+ \times \Sigma^+$ be a finite sequence of non-empty strings over Σ for which the following restricted version of the Modified Post Correspondence Problem (MPCP) is undecidable:

Instance A non-empty string $\alpha \in \Sigma^+$.
Question Do there exist indices $i_1, \ldots, i_k \in \{1, \ldots, n\}$ such that

$$\alpha\phi_{i_1}\phi_{i_2}\ldots\phi_{i_k} = \psi_{i_1}\psi_{i_2}\ldots\psi_{i_k}?$$

For any string w over Σ, let $\widetilde{w}(x)$ denote the term formed by treating a and b as unary function symbols and the concatenation operator as function composition; more precisely, we set:

$$\widetilde{\lambda}(x) = x, \qquad \widetilde{au}(x) = a(\widetilde{u}(x)), \qquad \widetilde{bu}(x) = b(\widetilde{u}(x)).$$

Let f be a ternary function and $g_1, \ldots, g_n$ be distinct unary function symbols. Consider then the system $\mathcal{S}$ formed of the following rewrite rules:

$$f(\widetilde{\phi_i}(x),\ g_i(y),\ \widetilde{\psi_i}(z)) \to f(x,\ y,\ z)$$

for every pair (ϕ_i, ψ_i) of the MPCP. We also add a new unary function symbol h and the following set Δ of dwindling rules:

$$h(a(x)) \to x, \quad h(b(x)) \to x, \qquad h(g_i(x)) \to x, \ \ i \in \{1, \ldots, n\}.$$

The effect of this addition is that the monadic functions $a, b, g_1, \ldots, g_n$ are all Δ-transparent, whereas f is Δ-resistant. The role played by the g_i is to ensure that the rewrite system has no critical pairs. The system $\mathcal{R}$ formed of all these rewrite rules (i.e., $\mathcal{S} \cup \Delta$) is therefore convergent and Δ-strong (all function symbols being considered public). It is not hard then to see that the following unification problem

$$f(X,\ Y,\ \widetilde{\alpha}(X)) \ =^?_{\mathcal{R}} \ f(c,\ c,\ c)$$

has a solution if and only if the instance of the restricted MPCP above has a solution.

The "if" part is fairly straightforward: If $\alpha\phi_{i_1}\phi_{i_2}\ldots\phi_{i_k} = \psi_{i_1}\psi_{i_2}\ldots\psi_{i_k}$ for some indices $i_1, \ldots, i_k \in \{1, \ldots, n\}$, then the substitution

$$\tau = \{X \leftarrow \widetilde{\phi_{i_1}}\widetilde{\phi_{i_2}}\ldots\widetilde{\phi_{i_k}}(c),\ Y \leftarrow g_{i_1}g_{i_2}\ldots g_{i_k}(c)\}$$

is a solution for the unification problem: indeed, we have

$$\widetilde{\alpha}(\tau(X)) = \widetilde{\alpha}\,\widetilde{\phi_{i_1}}\widetilde{\phi_{i_2}}\ldots\widetilde{\phi_{i_k}}(c) = \widetilde{\psi_{i_1}}\widetilde{\psi_{i_2}}\ldots\widetilde{\psi_{i_k}}(c).$$

On the other hand, suppose θ is a solution for the above equation. Without loss of generality it can be assumed that θ is $\mathcal{R}$-*normalized*. Then the following necessarilty holds: $f(\theta(X),\ \theta(Y),\ \widetilde{\alpha}(\theta(X))) \longrightarrow^!_{\mathcal{S}} f(c,\ c,\ c)$. Now a solution for the MPCP instance can be obtained from $\theta(Y)$. □

References

1. Anantharaman, S., Narendran, P., Rusinowitch, M.: Intruders with Caps. In: Proc. of the Int. Conference RTA'07. LNCS, vol. 4533, pp. 20–35. Springer-Verlag (2007)
2. Anantharaman, S., Lin, H., Lynch, C., Narendran, P., Rusinowitch, M.: Unification modulo homomorphic encryption. In: Proc. of Int. Conf. FROCOS 2009, TRENTO-Italy. LNAI, vol. 5749, pp. 100–116. Springer-Verlag (2009)
3. Anantharaman, S., Lin, H., Lynch, C., Narendran, P., Rusinowitch, M.: Cap Unification: application to protocol security modulo homomorphic encryption. In: Proc. of the 5th ACM Symp. on Information, Computer and Communications Security, ASIACCS'10, pp. 192–203. ACM, New York (2010)
4. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press, New York (1998)
5. Baudet, M.: Deciding security of protocols against off-line guessing attacks. In: Proc. of the 12th ACM Conf. on Computer and Comm. Security, CCS'05, pp. 16–25. ACM, New York (2005)
6. Comon-Lundh, H., Treinen, R.: Easy intruder deductions. Verification: theory and practice In: LNCS 2772, pp. 225–242. Springer-Verlag (2004)
7. Comon-Lundh, H., Shmatikov, V.: Intruder deductions, constraint solving and insecurity decision in presence of exclusive-or. In: Proc. of the Logic in Computer Science Conference, LICS'03, pp. 271–280. IEEE Computer Society Press (2003)
8. Cortier, V., Delaune, S., Lafourcade, P.: A survey of algebraic properties used in cryptographic protocols. J. Comput. Secur. **14**(1), 1–43 (2006)
9. Cortier, V., Rusinowitch, M., Zalinescu, E.: A resolution strategy for verifying cryptographic protocols with CBC encryption and blind signatures. In: Proc. of the 7th ACM SIGPLAN Conf., PPDP'05, pp. 12–22 (2005)
10. Delaune, S., Jacquemard, F.: A decision procedure for the verification of security protocols with explicit destructors. In: Proc. of the 11th ACM Conference on Computer and Communications Security (CCS'04), pp. 278–287. ACM, Washington (2004)
11. Kremer, S., Ryan, M.D.: Analysing the vulnerability of protocols to produce known-pair and chosen-text attacks. In: Proc. of the 2nd Int. Workshop on Security Issues in Coordination Models, Languages, and Systems (SecCo 2004), ENTCS, vol. 128, Issue 5, pp. 87–104 (2004)
12. Lafourcade, P.: Relation between unification problem and intruder deduction problem. In: Proc. of the 3rd Int. Workshop on Security and Rewriting Techniques (SecReT'08). Pittsburgh, USA (2008)
13. Lafourcade, P., Lugiez, D., Treinen, R.: Intruder deduction for the equational theory of Abelian groups with distributive encryption. Inf. Comput. **205**(4), 581–623 (2007)
14. Millen, J., Shmatikov, V.: Constraint solving for bounded-process cryptographic protocol analysis. In: Proc. of the 8th ACM Conference on Computer and Communications Security, pp. 166–175. ACM, New York (2001)
15. Narendran, P., Marshall, A., Mahapatra, B.: On the Complexity of the Tiden-Arnborg Algorithm for Unification modulo One-Sided Distributivity. Int. Workshop on Unification UNIF'2010, Edinburgh (2010)
16. Narendran, P., Pfenning, F., Statman, R.: On the unification problem for cartesian closed categories. In: Proc. of the Logic in Computer Science Conference LICS'93, pp. 57–63 (1993)
17. Schaefer, T.J.: The complexity of satisfiability problems. In: Proc. of the 10th Annual ACM Symposium on Theory of Computing, pp. 216–226 (1978)
18. Tiden, E., Arnborg, S.: Unification problems with one-sided distributivity. J. Symb. Comput. **3**(1–2), 183–202 (1987)