*Journal of Automated Reasoning* **33:** 1–28, 2004.



# Unification Modulo *ACUI* Plus Distributivity Axioms

SIVA ANANTHARAMAN, PALIATH NARENDRAN[1] and MICHAEL RUSINOWITCH[2]
*LIFO, Orléans, France. e-mail: siva@lifo.univ-orleans.fr*
[1]*University at Albany, SUNY, USA. e-mail: dran@cs.albany.edu*
[2]*LORIA, Nancy, France. e-mail: rusi@loria.fr*

(Accepted: 15 May 2004)

**Abstract.** $E$-unification problems are central in automated deduction. In this work, we consider unification modulo theories that extend the well-known *ACI* or *ACUI* by adding a binary symbol "$*$" that distributes over the $AC(U)I$-symbol "$+$." If this distributivity is one-sided (say, to the left), we get the theory denoted $AC(U)ID_l$; we show that $AC(U)ID_l$-unification is DEXPTIME-complete. If "$*$" is assumed two-sided distributive over "$+$," we get the theory denoted $AC(U)ID$; we show unification modulo $AC(U)ID$ to be NEXPTIME-decidable and DEXPTIME-hard. Both $AC(U)ID_l$ and $AC(U)ID$ seem to be of practical interest, for example, in the analysis of programs modeled in terms of process algebras. Our results, for the two theories considered, are obtained through two entirely different lines of reasoning. A consequence of our methods of proof is that, modulo the theory that adds to $AC(U)ID$ the assumption that "$*$" is associative-commutative, or just associative, unification is undecidable.

**Key words:** rewriting, equational unification, counter machines, Post correspondence problem, set constraints, decidability, complexity.

## 1. Introduction

The unification problem for the theories *AC* ("associativity-commutativity"), *ACI* ("AC plus idempotence"), and *ACUI* ("ACI with unit element") have been studied in great detail. Natural extensions are the theories that one obtains by adjoining a binary operator "$*$," which is assumed two-sided, or just left-, distributive over an *ACUI*-symbol "$+$"; we denote these theories respectively by *ACUID* and $ACUID_l$. From a practical point of view, the theory *ACUID* can be used in program specifications based on set constraints, whereas the theory $ACUID_l$ can be used in analyzing processes for *noninterference* through trace equivalence, since the concatenation of action symbols is left-distributive over the choice operator "$+$" on processes. To our knowledge the unification problems over *ACUID* or $ACUID_l$ have not been studied so far. (The theory $ACUID_r$ – where only right-distributivity of "$*$" over "$+$" is assumed – is similar to $ACUID_l$.) On the other hand, if we drop the idempo-

tence assumption on "+," all these unification problems get close to Hilbert's tenth problem and therefore are undecidable (see the Conclusion).

Because of the distributivity assumption, any ground term over an $ACUID_l$ (or *ACUID*) signature defines a homomorphism w.r.t. the *ACUI*-operator "+." In a sense, therefore, the theory *ACUIH* obtained by adjoining a set $H$ of homomorphisms over the *ACUI* symbol "+" can be viewed as intermediary between *ACUI* and *ACUID* or $ACUID_l$. It is only natural, then, to consider first the unification problem over such theories. Now, in [6] the case where $H$ is a set of *noncommuting* homomorphisms over the *ACUI*-symbol has been studied in detail, and its unification problem has been proved DEXPTIME-complete. So we begin by considering the unification problem modulo "*ACUI* plus a set of *commuting* homomorphisms," a theory that we denote $ACUIH^C$. We prove that $ACUIH^C$-unification is undecidable. The proof involves reducing a reachability problem for commutative word rewriting (with constraints) to $ACUIH^C$-unification; we then show that the halting problem of Minsky counter machines can be reduced to this reachability problem for commutative word rewriting (with constraints). These results constitute Section 2.

Unification modulo $ACUID_l$ is actually reducible to unification modulo *ACUIH* for a *noncommuting* set of homomorphisms $H$; unification modulo $ACUID_l$ thus turns out to be decidable in EXPTIME. On the other hand, we also deduce a DEXPTIME lower bound for this problem using reduction from the results of [6]; the reasoning employed – and the DEXPTIME lower bound obtained – hold for *ACUID*-unification as well. Section 3 is devoted to these proofs.

In Section 4 we turn our attention to the *ACUID*-unification problem. We first show, as an immediate consequence of the results of Section 2, that *ACUID*-unification is undecidable if the symbol "$*$" (assumed two-sided distributive over the *ACUI*-symbol "+") is assumed *AC* in addition. We show next that, if "$*$" is assumed associative besides being two-sided distributive over "+," *ACUID*-unification still remains undecidable: the proof is by reduction from the modified Post correspondence problem.

When no further laws other than two-sided distributivity over "+" are assumed on "$*$," the *ACUID*-unification problem can be formulated as a problem of solvability, *in terms of finite sets*, of a particular class of set constraints. Under such a vision *ACID*-unification is the problem of solving the same set constraints in terms of finite *nonempty* sets. It can be deduced from the more general results of [13] that this is decidable, although without getting an explicit algorithm with known complexity bounds. We give in Section 5 a direct NEXPTIME algorithm for solving the *ACID*-unification problem, in terms of nonempty finite sets of ground terms. Obviously this NEXPTIME upper bound holds also for *ACUID*-unification; and we show in an Appendix how to modify the lines of reasoning of [6] so as to get rid of the use of the unit element $U = 0$ made there, in order to deduce a DEXPTIME lower bound for *ACID*-unification.

We note that for the particular class of set constraints expressing the $AC(U)ID$-unification problem, if the solutions can be arbitrary sets – meaning possibly infinite, or empty – then satisfiability is also known to be NEXPTIME-decidable [2] and DEXPTIME-hard [9].

We express our thanks to Hubert Comon and to one of the referees, for having pointed out that $ACID$-unification can be done more simply (with the algorithm given in this paper) than by a more complicated construction based on labeled dag automata that we presented in the initial version of the paper (see [4]). Our thanks also to Franz Baader for a useful remark concerning the complexity of $ACUID_l$-unification.

## 2. *ACUI* with Commuting Homomorphisms

An $ACUIH^C$-signature consists of an $AC$-symbol denoted by "+" and assumed idempotent, a unit element for this "+" (denoted "0"), a finite set $S = \{\alpha, \beta, \ldots\}$ of symbols, additive homomorphisms $\{h_\alpha, h_\beta, \ldots\}$ indexed by the symbols in $S$, and commutativity relations between these homomorphisms. It is assumed that each homomorphism maps 0 to 0. The equational theory $ACUIH^C$ thus defined is, in more formal terms,

$$x + (y + z) \approx (x + y) + z, \qquad x + y \approx y + x,$$
$$x + 0 \approx x, \qquad x + x \approx x,$$
and for all $u, v \in S$:
$$h_u(x + y) \approx h_u(x) + h_u(y), \qquad h_u(h_v(x)) \approx h_v(h_u(x)), \qquad h_u(0) = 0.$$

As usual we also assume that there are finitely many free constants (not elements of $S$). The set of ground terms over such a signature $ACUIH^C$ is denoted by $\mathcal{G}$; a ground term is thus a sum, or set, of *commutative* strings over the symbols representing the commuting homomorphisms each applied itself to a ground term. The idempotent $AC$ symbol "+" is viewed in this section (implicitly) as the set-union operation. The unification problem over such a signature reduces to solving systems of linear equations over unknowns, with commutative strings over the homomorphism symbols as coefficients; the solutions are to be sets of such strings applied to ground constants. (For a systematic presentation on how to formulate such $E$-unification problems in terms of solving linear equations over appropriate semirings, the interested reader may consult [5, 17].) For instance, if $f, g$ are homomorphism symbols and $a$ is a free constant, then the unification problem $f(f^2ga + gX) = gf(fa + fX)$ reduces to solving the "linear equation"

$$f^3ga + fgX = gf^2a + gf^2X$$

for the unknown $X$ in $\mathcal{G}$. (Note: parentheses for the arguments of the homomorphism symbols are often omitted.) The substitution $\{X \leftarrow fa\}$ is a solution here.

We show in this section that the $ACUIH^C$-unification problem with free constants is undecidable. The proof is from a reduction of the reachability problem for

configurations of Minsky's counter machines. We briefly outline the ideas behind. As we just saw, the problem is that of solving a system of finitely many linear equations of the form

$$l_1^{(j)} X_1 + \cdots + l_n^{(j)} X_n + V^{(j)} = r_1^{(j)} X_1 + \cdots + r_n^{(j)} X_n + U^{(j)}, \quad 1 \leq j \leq m,$$

where the $X_i$, $1 \leq i \leq n$, are the "unknowns," the $U^{(j)}$, $V^{(j)}$ are sets of ground terms, and $l^{(j)}, r^{(j)}$ are commutative strings over the homomorphism symbols, which we also call *ground terms* or *ground strings* with no risk of confusion. Without any loss of generality, we *assume that the unit element "0" for the ACUI-symbol "+" is not in $U^{(j)}$ or $V^{(j)}$, for any $j$*. To such a given unification problem, we associate, with every $1 \leq j \leq m$, the set of rewrite rules $l_i^{(j)} \to r_i^{(j)}$, $1 \leq i \leq n$, and use the usual notion of AC-rewrite steps on ground terms, via these rules; this rewrite relation is denoted by "$\Longrightarrow_j$." The first step in our proof is to show that – provided the transitive closure of "$\Longrightarrow_j$" has no cycles for any $j$ – the system of linear equations of the type above has a solution if and only if, for every $j$, the set $V^{(j)}$ is a set of elements reachable from elements of the set $U^{(j)}$ for the relation "$\Longrightarrow_j^*$." The second step is a reduction of the halting problem of Minsky machines with two counters to this reachability problem.

### 2.1. *ACUIH*$^C$-UNIFICATION PROBLEM VS. REWRITE REACHABILITY

Before establishing our first step, a notational convention: for any binary relation "$\Longrightarrow$" between terms, we denote as usual by "$\Longrightarrow^+$" and "$\Longrightarrow^*$" its transitive closure and reflexive transitive closure, respectively. If $u, v$ are terms such that $u \Longrightarrow^* v$, we say that $v$ is reachable or derivable from $u$ (or that $u$ derives $v$) for the relation "$\Longrightarrow$." For any given pair of sets of terms $S1$, $S2$, we write $S1 \models\!\Longrightarrow^* S2$ iff

(i) for any $v \in S2$, there exists a $u \in S1$ such that $u \Longrightarrow^* v$; and
(ii) for any $u \in S1$, there exists a $v \in S2$ such that $u \Longrightarrow^* v$.

We then say the set $S2$ is a set of *reachable* (or *derivable*) elements from the set $S1$, for the relation "$\Longrightarrow$," which we omit to mention if clear from the context.

PROPOSITION 1. *With the above notation, let*

$$l_1^{(j)} X_1 + \cdots + l_n^{(j)} X_n + V^{(j)} = r_1^{(j)} X_1 + \cdots + r_n^{(j)} X_n + U^{(j)}, \qquad (\&)$$

*$1 \leq j \leq m$, be an ACUIH$^C$-unification problem. For every $j$, let $\{l_i^{(j)} \to r_i^{(j)},\ i = 1..n\}$ be the associated ground AC-rewrite system, and "$\Longrightarrow_j$" the corresponding rewrite relation. If the problem* (&) *is solvable,* and *if the relation $\Longrightarrow_j$ is acyclic (i.e., $\Longrightarrow_j^+$ is irreflexive) for every $j = 1..m$, then $U^{(j)} \models\!\Longrightarrow_j^* V^{(j)}$ for every $j$.*

*Proof.* Here is the proof when the number of equations $m$ in the system is 1 (the reasoning goes through verbatim, except for the notation, when $m > 1$); in what follows we therefore drop the indices $j$. By hypothesis (&) is solvable; hence (after

replacing each variable $X_i$ by the corresponding set of terms in a given solution) we assume that the $X_i$ stand for sets satisfying (&). The proof is by induction on the minimal "measure" $|X_1|+\cdots+|X_n|$ w.r.t. the given solutions $(X_1, \ldots, X_n)$, for any unification problem with the same associated rewrite systems (here $|S|$ denotes the cardinality of the set $S$, as usual). If $|X_1| + \cdots + |X_n| = 0$, then $U = V$, and we have the trivial base case; so let us suppose $U \neq V$.

*Case (a):* $U \not\subset V$. Let $w$ be a string in $U \setminus V$. Without loss of generality we assume $w \in l_1 X_1$, so $w = l_1 w_1$ for some $w_1 \in X_1$; then write $X_1 = w_1 + Y_1$ with $w_1 \notin Y_1$ (and $w_1 \neq 0$).

Thus $r_1 X_1 = r_1 w_1 + r_1 Y_1$ and $(Y_1, X_2, \ldots, X_n)$ has a strictly smaller measure than $(X_1, \ldots, X_n)$ and satisfies the unification problem:

$$l_1 Y_1 + \cdots + l_n X_n + (V + l_1 w_1) = r_1 Y_1 + \cdots + r_n X_n + (U + r_1 w_1).$$

By the induction hypothesis, therefore, we have $(U + r_1 w_1) \models\!\!\Longrightarrow^* (V + l_1 w_1)$.

This means that (i) every term in $V$ is reachable from some term in $U + r_1 w_1$ and hence from some term in $U + l_1 w_1 = U$, and (ii) every term in $U$ can derive some term in $V + l_1 w_1$. Furthermore, $r_1 w_1$ must derive elements in $V$, since $\Longrightarrow$ is assumed acyclic. Thus, every term in $U$ derives some term in $V$.

*Case (b):* $U \subset V$. Let $x$ be a term in $V$ not in $U$; without loss of generality we assume $x \in r_1 X_1$, that is, $x = r_1 x_1$ for some $x_1 \in X_1$. Let then $X_1 = x_1 + Z_1$ with $x_1 \notin Z_1$ (and $x_1 \neq 0$).

Thus $l_1 X_1 = l_1 x_1 + l_1 Z_1$; now $(Z_1, X_2, \ldots, X_n)$ has a strictly smaller measure than $(X_1, \ldots, X_n)$ and satisfies the unification problem:

$$l_1 Z_1 + \cdots + l_n X_n + (V + l_1 x_1) = r_1 Z_1 + \cdots + r_n X_n + (U + r_1 x_1).$$

Again, by the induction hypothesis we have $(U + r_1 x_1) \models\!\!\Longrightarrow^* (V + l_1 x_1)$. Now, since the relation $\Longrightarrow$ is assumed acyclic, it cannot be that $r_1 x_1 \Longrightarrow^* l_1 x_1$, so there must be some string in $U$ *other than* $r_1 x_1$ that reduces to $l_1 x_1$, and hence to $r_1 x_1$ as well. This implies then that any element of $V$ is reachable from elements of $U$. Since $U \subset V$, the result follows. □

*Remark 1.* The above proposition ceases to be true if the rewrite relation is not acyclic. Here is a counterexample. We suppose we are given two commuting homomorphisms $a, b$ over "+" and consider the following unification problem:

$$l_1 X_1 + l_2 X_2 + a^3 b^3 + a^4 b^2 + a^3 b^2 = r_1 X_1 + r_2 X_2 + a^3 b^3,$$

where $l_1 = a^2 b, r_1 = ab$ and $l_2 = a^2 b^2, r_2 = a^3 b^2$; the strings on $a, b$ with no following variables are all assumed evaluated at some fixed ground constant. This problem does have a solution, namely, $X_1 = a^2 b$, $X_2 = a$ (since "+" is assumed idempotent). But neither $a^4 b^2$ nor $a^3 b^2$ can be reached from $a^3 b^3$ via the rewrite rules $l_i \to r_i$, $i = 1, 2$: indeed, each rule consuming a power of $b$ brings back exactly the same power of $b$; and the rewrite relation defined by these rules has cycles $a^2 b^2 \Longrightarrow a^3 b^2 \Longrightarrow a^2 b^2$.

PROPOSITION 2. *Let* (&) *be a given ACUIH$^C$-unification problem:*

$$l_1^{(j)}X_1 + \cdots + l_n^{(j)}X_n + V^{(j)} = r_1^{(j)}X_1 + \cdots + r_n^{(j)}X_n + U^{(j)},$$
$$1 \leq j \leq m; \qquad (\&)$$

*for every $j$, let $\{l_i^{(j)} \to r_i^{(j)}, i = 1..n\}$ be the associated ground AC-rewrite system, "$\Longrightarrow_j$" the corresponding AC-rewrite relation. Then* (&) *is solvable* if $U^{(j)} \models\!\!\Longrightarrow_j^* V^{(j)}$, *for every $j = 1..m$.*

*Proof.* We again give the proof only for $m = 1$ (the same reasoning holds for any $m$), and omit the indices $j$ in what follows. Let us define the *distance* between $U$ and $V$ (w.r.t. the relation $\Longrightarrow$) as the sum of the (minimal) number of rewrite steps needed for deriving the elements of $V$ from the elements of $U$. We reason by induction on this distance. The case of distance 0 corresponds to the base case $U = V$, where the assignment $X_i = 0, i = 1..n$, gives a solution. So assume this distance $d$ to be nonzero; then there exist $u \in U$ and $v \in V$ such that $u \Longrightarrow^+ v$; without loss of generality we assume that $u = l_1u'$ and $r_1u' \Longrightarrow^* v$.

Now write $U = u + U'$ with $u \notin U'$; then the sets $(r_1u' + U')$ and $V$ are such that $(r_1u' + U') \models\!\!\Longrightarrow^* V$, and the distance (w.r.t. $\Longrightarrow$) from $(r_1u' + U')$ to $V$ is strictly smaller than $d$. So, by the induction hypothesis there must exist sets $X_1', \ldots, X_n'$ such that

$$l_1X_1' + \cdots + l_nX_n' + V = r_1X_1' + \cdots + r_nX_n' + (r_1u' + U').$$

Setting $X_1 = X_1' + u'$ and $X_i = X_i'$ for all $i > 1$, we get a solution for the problem (&). □

Putting these two propositions together, we get the following theorem.

THEOREM 1. *Let* (&) *be a given ACUIH$^C$-unification problem:*

$$(\&):\ l_1^{(j)}X_1 + \cdots + l_n^{(j)}X_n + V^{(j)} = r_1^{(j)}X_1 + \cdots + r_n^{(j)}X_n + U^{(j)},$$
$$1 \leq j \leq m;$$

*for any $j = 1..m$, let $\{l_i^{(j)} \to r_i^{(j)}, i = 1..n\}$ be the ground AC-rewrite system associated, "$\Longrightarrow_j$" the corresponding rewrite relation, and assume that the relation $\Longrightarrow_j^+$ is acyclic. Then the problem* (&) *is solvable if and only if $U^{(j)} \models\!\!\Longrightarrow_j^* V^{(j)}$, for every $j = 1..m$.*

COROLLARY 1.1. *Let $a_1, \ldots, a_m$ be commuting homomorphisms. Then the equation $a_1X_1 + \cdots + a_mX_m + \epsilon = X_1 + \cdots + X_m + W$ is solvable if and only if $W$ consists only of strings over the $a_i, i = 1..m$. (The $\epsilon$ here stands for the empty substitution seen as a ground term.)*

In particular, if $a, b, c$ are commuting homomorphisms, then the equation $a(x_1) + b(x_2) + \epsilon = x_1 + x_2 + W$ forces $W$ to consist entirely of terms not containing $c$.

## 2.2. *ACUIH*$^C$-UNIFICATION IN TERMS OF ADMISSIBLE REACHABILITY

A ground term in $\mathcal{G}$ is said to contain another given ground term of $\mathcal{G}$ as a factor iff the latter is a subterm of the former up to associativity-commutativity.

DEFINITION 1. A marked rewrite rule over the set $\mathcal{G}$ is a rule denoted under the form $l \to r\,[a]$, where the *mark* $a$ is a given ground term, and $l, r$ are strings such that $l$ does *not contain the mark* $a$ *as a factor*. A rewrite step using the rewrite rule $l \to r\,[a]$ is said to be *admissible* iff the string rewritten does *not* contain the mark $a$ as a factor.

A rewrite rule with no attached mark may be seen as marked with the empty string; any ground string can be rewritten with such a rewrite rule. Given a set of marked rewrite rules over $\mathcal{G}$, we henceforth denote by $\Longrightarrow$ the associated relation of admissible reduction and by $\models\!\!\Longrightarrow^*$ the relation of *admissible reachability* (or derivability) between sets of terms, defined as in the previous subsection but here w.r.t. this restricted notion $\Longrightarrow$ of *admissible reduction*.

THEOREM 2. *Let* (&): $l_1X_1 + \cdots + l_nX_n + V = r_1X_1 + \cdots + r_nX_n + U$ *be a given ACUIH*$^C$*-unification problem. Let* $\{l_i \to r_i\}$ *be the associated AC-rewrite system, each rule marked respectively with either a constant* $a_i$ *or the empty string; and denote by "*$\Longrightarrow$*" the associated admissible AC-rewrite relation, which we assume to be acyclic. Consider then the following constraint on the* $X_i$*:*

($\mathcal{C}$) *If* $l_i \to r_i$ *is marked with* $a_i$*, then*
$X_i$ *does not contain terms in which the mark* $a_i$ *occurs.*

*Then the problem* (&) *is solvable under constraint* $\mathcal{C}$ *if and only if we have* $U \models\!\!\Longrightarrow^* V$.

The proof of Theorem 1, based on Propositions 1 and 2, goes through verbatim; we mention here the points to check, for completeness. (We are assuming each rewrite rule $l_i \to r_i$ is marked with an $a_i$, which is a constant or the empty string.)

Proposition 1, case (a): If $(X_1, \ldots, X_n)$ is a solution satisfying constraint $\mathcal{C}$, $w = l_1w_1$, and $X_1 = w_1 + Y_1$, then $(Y_1, X_2, \ldots, X_n)$ continues to satisfy the constraint $\mathcal{C}$, so induction goes through. The same reasoning holds also for case (b).

Proposition 2: The distance from $U$ to $V$ is defined now over admissible rewrite steps. If $u = l_1u' \Longrightarrow r_1u'$ admissibly, then $u'$ doesn't contain the mark $a_1$ on the rule $l_1 \to r_1\,[a_1]$; so if $(X'_1, X'_2, \ldots, X'_n)$ is a solution satisfying constraint $\mathcal{C}$ for the linear equation at the inductive step, then the solution $(X'_1 + u', X'_2, \ldots, X'_n)$ for the initial equation also satisfies the constraint $\mathcal{C}$.

We can actually get rid of the mark constraint $\mathcal{C}$ above.

LEMMA 1. *With the notation of the above theorem, there is a system* (&′) *of ACUIH*$^C$*-linear equations naturally associated to* (&) *such that*

(&) *is solvable under the constraint* $\mathcal{C}$ *if and only if* (&′) *is solvable.*

*Proof.* It suffices to observe, thanks to Corollary 1.1, that the constraint $\mathcal{C}$ is itself expressible in terms of linear $ACUIH^C$-equations: indeed the following equation

$$a_1X_1 + \cdots + a_{j-1}X_{j-1} + \epsilon + a_{j+1}X_{j+1} + \cdots + a_nX_n = X_1 + X_2 + \cdots + X_n$$

forces the component $X_j$ of the solution to be free from terms containing $a_j$, assuming that the marks $a_i$ are all distinct. □

We show in the next section that *admissible reachability is undecidable*, via reduction from the Minsky counter machine problem. From this and the above two results, we will be able to deduce the undecidability of $ACUIH^C$-unification.

### 2.3. MINSKY COUNTER MACHINES AND ADMISSIBLE REACHABILITY

A Minsky machine with two counters $C_1, C_2$ storing nonnegative integer values executes *programs* that are finite lists of *instructions* labeled with the natural numbers from 1 to $L$, each having one of the following forms where $1 \leq l \leq (L-1)$, $k \in \{1, \ldots, L\}$, $k \neq l$, and $i$ is 1 or 2:

(i) $l$: ADD 1 to $C_i$ and GOTO $l+1$;
(ii) $l$: If $C_i \neq 0$
then SUBTRACT 1 from $C_i$ and GOTO $l+1$;
else GOTO $k$;
(iii) $L$: STOP.

Any given program $\mathcal{P}$ is assumed to have exactly one instruction "STOP" labeled $L$. A *configuration* of such a two-counter machine, at any given stage of a computation, can be defined as a triple $(l, C_1, C_2)$, where $l$ is the (label of the) next instruction to execute and $C_i$, $i = 1, 2$, are the current integer values of the two counters. The following result on Minsky's two-counter machines is classical and will serve our purposes well, after some minor adaptations.

THEOREM 3 (Minsky [14]). *For every partial recursive function $f$ on natural numbers there exists a program $\mathcal{Q}$ such that applied with $(1, 2^d, 0)$ as starting configuration, $\mathcal{Q}$ halts with the final configuration $(L, 2^{f(d)}, 0)$ if $f(d)$ is defined on $d$, and does* not *halt otherwise. In particular it is undecidable whether an arbitrarily chosen program $\mathcal{P}$ will halt when applied with an arbitrarily given starting configuration $(1, 2^d, 0)$.*

We introduce a homomorphism symbol $h_l$ for every instruction $l = 1..L$ and add two more homomorphisms $h_{c_i}, i = 1, 2$, for the two counters. Any given machine configuration $(l, c_1, c_2)$ is then seen as the commutative string $h_l h_{c_1}^{m_1} h_{c_2}^{m_2}$.

Each type of machine instruction of the above form can then be seen as an admissible rewrite step on these strings, using one among the following *marked* rewrite rules (where as previously $1 \leq l \leq (L-1)$, $k \in \{1, \ldots, L\}$, $k \neq l$; and $i$ is 1 or 2):

(i) For each instruction of type (i) as above, add the rule

$$h_l \to h_{c_i} h_{l+1}.$$

(ii) For each instruction of type (ii) as above, add the rules

$$h_{c_i} h_l \to h_{l+1}, \qquad h_l \to h_k[h_{c_i}].$$

Any given machine program can thus be transformed into a rewrite program, by translating each instruction as a *well-determined* rewrite rule from this set. Note that the last sets of rewrite rules are *marked*; they can be applied only to rewrite strings with no first (resp. no second) counter. It is important to note *also* that any sequence of admissible rewrite steps starting from the configuration string $h_1 h_{c_1}^{2^d}$ corresponds to a correct sequence of machine instructions and vice versa. The final statement of the above theorem of Minsky on two-counter machines can then be formulated as follows: Given two arbitrary strings $s = h_1 h_{c_1}^{2^d}$ and $f = h_L h_{c_1}^{2^m}$ over our $ACUIH^C$-signature, it is undecidable in general whether the latter is reachable from the former via admissible rewrite steps, using the marked rules of the above rewrite system.

In order to complete the proof that $ACUIH^C$-unification is undecidable via our Theorem 2 established above, there remains the hitch that this result made an assumption of acyclicity on the rewrite relation: this does not hold for the above rewrite system. To get that property, we introduce a *third* counter $C_3$, with the idea that

- initially counter $C_3$ has value 1,
- every instruction of type (i) or (ii) also increments $C_3$ by 2,
- an instruction of type (iii) increments $C_3$ by 1 and goes to a new instruction (iv),
- where $C_3$ is decremented by 2 as long as possible; else we STOP.

Following the same lines of thinking as above, such a three-counter machine program $\mathcal{P}'$ extending an earlier two-counter program $\mathcal{P}$ can be visualized as executions of appropriate admissible rewrite steps applying rules from the following system (where again, $1 \leq l \leq (L-1)$, $k \in \{1, \ldots, L\}$, $k \neq l$; and $i$ is 1 or 2):

(i) (Increment $C_1$ or $C_2$, add 2 to $C_3$): $h_l \to h_{c_i} h_{l+1} h_{c_3}^2$,
(ii) (Conditional decrement on $C_i$, add 2 to $C_3$):

$$h_{c_i} h_l \to h_{l+1} h_{c_3}^2, \qquad h_l \to h_k h_{c_3}^2[h_{c_i}],$$

(iii) (Pause) $h_L \to h_{L+1} h_{c_3}$,

(iv) (Decrement $C_3$ by 2 whenever possible, or STOP)

$$h_{L+1}h_{c_3}^2 \to h_{L+1}, \qquad h_{L+1} \to h_{L+2}[h_{c_3}].$$

(The program $\mathcal{P}'$ has obviously two more instructions than $\mathcal{P}$.) The marks on the rewrite rules above are $h_{c_1}$, $h_{c_2}$, and $h_{c_3}$. It is easily checked that this rewrite system is acyclic. The following fact is equally obvious: Our earlier two-counter machine run, on a program $\mathcal{P}$ starting from an initial configuration $h_1 h_{c_1}^{2^d}$, halts with $h_L h_{c_1}^{2^m}$ as final configuration *if and only if* our current three-counter machine, run on the extended program $\mathcal{P}'$ from the initial configuration $h_1 h_{c_1}^{2^d} h_{c_3}$, halts with $h_{L+2} h_{c_1}^{2^m}$ as final configuration.

We can now formulate our undecidability result, deduced by appealing to the results we proved in the earlier sections, combined with the above reformulation of Minsky's theorem in terms of a three-counter machine.

THEOREM 4. *The ACUIH$^C$-unification problem is undecidable.*

## 3. *ACUID$_l$*-Unification Is DEXPTIME-Complete

The equational theory *ACUID$_l$* we are considering in this section is

$$x + (y + z) \approx (x + y) + z, \qquad x + y \approx y + x, \qquad x + 0 \approx x,$$
$$x + x \approx x,$$
$$x * (y + z) \approx (x * y) + (x * z), \qquad x * 0 \approx 0, \qquad 0 * x \approx 0.$$

Any *ACUID$_l$*-unification problem can be transformed into a "standard form," that is, one of solving a set of equations where each equation has one of the following four forms:

$$\text{(i) } x = 0, \qquad \text{(ii) } x = a, \qquad \text{(iii) } x = y + z, \qquad \text{(iv) } x = y * z,$$

where $x, y, z$ are variables, 0 is the unit element for the *ACUI*-symbol "+," and $a$ is a ground constant. If equations of type (i) do not occur, then the problem is referred to as a *simple* unification problem.

DEFINITION 2. (i) A substitution is *deciduous* if each variable in its domain is replaced either by 0 or by another variable.

(ii) A unifier $\theta$ for an *ACUID$_l$*-unification problem $S$ is said to be a *discriminating unifier* iff the following holds for all variables in $Var(S)$:

$$\theta(u) \neq_{ACUID_l} 0 \quad \text{and} \quad \theta(v) =_{ACUID_l} \theta(w) \quad \text{iff} \quad v = w.$$

The following lemma is proved in a straightforward manner.

LEMMA 2. *An ACUID$_l$-unification problem $S$ is solvable iff there is a deciduous substitution $\eta$ such that*

(i) *either $\eta$ is a unifier for $S$,*
(ii) *or $\eta(S)$ has a "discriminating" unifier $\theta$.*

EXAMPLE A. The $ACUID_l$-problem $S'$: $V = X + Z, V = W + U, W = X * Y$ can be transformed under the deciduous substitution $\eta = \{Y \mapsto X, U \mapsto X, W \mapsto Z\}$ into the problem $S$: $V = X + Z, Z = X * X$.

Given any $ACUID_l$-problem in standard form, our first step is to choose nondeterministically a deciduous $\eta$, which reduces the given problem to a simple $ACUID_l$-unification problem. We get then the following.

**Instance**: A set of equations $S$, each of the form (ii), (iii), or (iv) defined above.
**Question**: Does $S$ have a discriminating $ACUID_l$-unifier?

Given such a unification problem $S$ where the goal is to find a discriminating unifier, our next step is to transform each equation of the form $x = y * z$ (case (iv) above) into the equation $x = h_y(z)$, where $h_y$ is a homomorphism we introduce. Let $\mathcal{H}(S)$ denote the set of all homomorphisms introduced in this way, and let $\mathcal{V}_h(S)$ denote the set of variables for which the homomorphisms get created. We thus get rid of "$*$," but our new equational theory consists of the *ACUI* axioms, plus the additional axioms

$$\{h(u_1 + u_2) \approx h(u_1) + h(u_2),\ h(0) \approx 0\}, \quad \text{for all } h \in \mathcal{H}(S).$$

We refer to the transformed unification problem as the *h-image* of $S$ and denote it by $T$. We can define this notion more precisely: Let $t$ be any term over the signature consisting of $*$, $+$, the variables, and free constants. We introduce distinct homomorphisms for the *ACUI*-equivalence classes of subterms of $t$ and define the following transformation $\xi$:

$$\begin{aligned}
&\xi(r) = r && \text{if } r \text{ is a variable or a constant,}\\
&\xi(x + y) = \xi(x) + \xi(y), &&\\
&\xi(s * t) = h_{[s]}(\xi(t)), && \text{where } [s] = \text{class of } s \text{ modulo } ACUI.
\end{aligned}$$

It is not difficult to see that $\xi$ is well defined and $\xi(t)$ is unique up to $=_{ACUI}$. This definition of $\xi$ can now be extended to sets of equations of terms. Thus, for the unification problem $S$, we have $h$-image$(S) = \xi(S)$. In what follows, we drop the square brackets for the indices of homomorphisms and consider them as defined modulo $=_{ACUI}$. The transformed terms are terms over an *ACUIH* signature and free constants, where $H$ is the set $\mathcal{H}(S)$ of (noncommuting) homomorphisms introduced above for the variables in $\mathcal{V}_h(S)$. It follows from the definitions that if $t_1, t_2$ are any two terms in the $ACUID_l$-signature and free constants, then $t_1 =_{ACUID_l} t_2$ if and only if $\xi(t_1) =_{ACUIH} \xi(t_2)$.

The transformed problem $T = h$-image of $S$ is thus an *ACUIH*-unification problem. However, it is not hard to see that the solvability of $T$ modulo *ACUIH* does not automatically ensure solvability of the original $ACUID_l$ problem. We need to impose additional constraints on the *ACUIH*-problem, which are reformulations of the usual occur-check condition on the variables: for instance, the substitution for a variable $x$ in $\mathcal{V}_h(S)$ must not contain $h_x$, the homomorphism introduced for $x$.

Cycles – for example, of the form $x$ replaced by a term containing $h_y$, and $y$ replaced by a term containing $h_x$ – should not occur either.

These occur-check constraints are formulated as requirements that the unifiability problem of $T = \xi(S)$ w.r.t. *ACUIH* be solved subject to *linear* constraints (specifying conditions like) $x \succ h_x$ for every homomorphism $h_x \in \mathcal{H}(S)$; that is, a unifier $\theta$ of $T$ should satisfy the condition that for every $x \in Var(T)$, $\theta(x)$ *does not contain* any occurrence of $h_x$.

DEFINITION 3. (i) Given a simple *ACUID*$_l$-unification problem $S$, and its $h$-image $T$ modulo *ACUIH*, a *linear constraint* $C$ is *a total order* $\succ_C$ over the set $Var(T) \cup \mathcal{H}(S)$ such that $x \succ_C h_x$ for all variables $x$ in $\mathcal{V}_h(S)$.

(ii) Let $T$ be an *ACUIH*-unification problem. A substitution $\beta$ whose domain is $Var(T)$ is said to satisfy a linear constraint $\succ_C$ if and only if the following holds: for every $x \in Var(T)$, $\beta(x)$ *does not contain* any of the function symbols below $x$ in $C$. In other words, if $x \succ_C h_j$, then $\beta(x)$ does not contain any occurrence of $h_j$.

These linear constraints are similar to the linear constant restrictions of [7]. The following two theorems relate the unification problem $S$ over *ACUID*$_l$ to its $h$-image $T$ over *ACUIH* (recall that $H$ denotes the set of homomorphisms $\mathcal{H}(S)$).

THEOREM 5. *If a simple ACUID*$_l$*-unification problem S has a discriminating unifier, then its h-image T is solvable as an ACUIH-unification problem. Furthermore, there is a linear constraint* $\succ_C$ *that the ACUIH-unifier satisfies* (*naturally*).

*Proof.* Let $\theta$ be a discriminating ground unifier of $S$. One could imagine defining a substitution $\alpha$ on $T$ as $\alpha(x) = \xi(\theta(x))$. But $\alpha$ is not (yet) a unifier for $T$: Consider an equation $x = y*z$ in $S$; its $h$-image is $x = h_y(z)$; but $\xi(\theta(y*z))$ is $h_{\theta(y)}(\xi(\theta(z)))$ and not $h_y(\xi(\theta(z)))$.

However, such an $\alpha$ can be "transformed" into an *ACUIH*-unifier $\beta$ by replacing, for every variable $v$ in $\mathcal{V}_h(S)$, $h_{\theta(v)}$ by the corresponding $h_v$ in every term. This mapping is *one-to-one* because $\theta$ is assumed to be a discriminating unifier.

To derive a linear constraint that $\beta$ must satisfy, assume a total *AC*-simplification ordering $>$ on ground terms (cf., e.g., [16]); and add a new constant, say $\perp$, smaller than every other symbol. Now order the terms in the set

$$\{\theta(x) \mid x \in Var(S)\} \cup \{\theta(x_1) * \perp, \ldots, \theta(x_n) * \perp\}$$

using $>$, where $\{x_1, \ldots, x_n\} = \mathcal{V}_h(S)$. All these terms will be distinct because $\theta$ is a discriminating unifier. Note also that any term that properly contains $\theta(x_i)$ is $> \theta(x_i) * \perp$. Replacing the $\theta(x_i)$'s by the corresponding $x_i$, and replacing the terms $(\theta(x_i) * \perp)$ by the corresponding $h_{x_i}$, we get a linear chain. The linear constraint $C$ is obtained by reversing the ordering relation on this chain. □

EXAMPLE A (contd.). Consider the *ACUID*$_l$-unification problem $V = X + Z$, $Z = X * X$, whose $h$-image is the *ACUIH*-problem $V = X + Z$, $Z = h_X(X)$. The

former admits a discriminating unifier $\theta$: $X \mapsto a, Z \mapsto a * a, V \mapsto a + (a * a)$. Following the above reasoning, we derive the *ACUIH*-unifier $\alpha$: $X \mapsto a,\ Z \mapsto h_X(a),\ V \mapsto a + h_X(a)$ for the latter problem.

As for the linear constraint that $\alpha$ satisfies (naturally), observe that, in the notation of the above proof, we have $\theta(V) > \theta(Z) > (\theta(X) * \bot) > \theta(X)$; so, again following the lines of reasoning used above, we deduce the constraint $C$: $V \prec Z \prec h_X \prec X$.

THEOREM 6. *Let $S$ be a simple $ACUID_l$-unification problem, and $T$ its $h$-image, which is an ACUIH-unification problem. If $T$ has a solution satisfying a linear constraint, then $S$ is solvable.*

*Proof.* Let $\beta$ be a ground *ACUIH*-unifier of $T$ that satisfies a linear constraint $C$. From $C$, we get a subconstraint $C'$ on the variables in $Var(T)$. Assume without loss of generality that $C' = x_n \succ \cdots \succ x_i \succ \cdots \succ x_1$. Now we use induction on $C'$ to form a substitution $\theta$ as a discriminating unifier for $S$.

Let us first consider the variable $x_n$ in $C'$. Since $x_n$ is the first variable, and $\beta(x_n)$ should not contain any item below $x_n$ in $C$, it must be that $\beta(x_n)$ does not contain any of the homomorphisms that were introduced, and we define $\theta(x_n) := \beta(x_n)$.

Assume that we have already constructed all the $\theta(x_{j'})$, $j \leq j' \leq n$. For variable $x_{j-1}$, $\beta(x_{j-1})$ could contain constants and some $h_{x_{w_i}}$ for $x_{w_i} \in \mathcal{V}_h(S)$ where each $h_{x_{w_i}} \succ x_{j-1}$. Since $x_{w_i} \succ h_{x_{w_i}}$, we have $x_{w_i} \succ h_{x_{w_i}} \succ x_{j-1}$. By the induction hypothesis, we have already constructed these $\theta(x_{w_i})$'s. Therefore, we can define $\theta(x_{j-1}) := \boldsymbol{rep}(\beta(x_{j-1}))$, where the function ***rep*** is defined as

$$\begin{aligned}
&\boldsymbol{rep}(a) = a, \text{ where } a \text{ is any constant}\\
&\boldsymbol{rep}(A + B) = \boldsymbol{rep}(A) + \boldsymbol{rep}(B)\\
&\boldsymbol{rep}(h_{x_{w_i}}(A)) = \theta(x_{w_i}) * \boldsymbol{rep}(A),
\end{aligned}$$

where $A, B$ stand for any terms. It can be shown that $\theta$ is a solution for $S$. Indeed consider each equation in $T$ of the form $x_{u_i} = h_{x_{w_i}}(x_{v_i})$. Since $\beta(x_{u_i}) =_{ACUIH} h_{x_{w_i}}(\beta(x_{v_i}))$, we have $\theta(x_{u_i}) =_{ACUID_l} \theta(x_{w_i}) * (\theta(x_{v_i}))$ by definition of $\theta$. □

The next step in our reasoning is to prove that the linear constraints over the *ACUIH*-signature – which express the condition that certain variables of the *ACUIH*-problem should not get replaced with terms containing certain given sets of homomorphism symbols – can themselves be formulated as instances of *ACUIH*-unification problems. This is done by using the following lemma, similar to the propositions and lemma we used earlier in Section 2.1; its proof goes along similar lines.

LEMMA 3. *Let $a_1, \ldots, a_m$ be homomorphisms and $c$ be a free constant. Then the equation $a_1(X_1) + \cdots + a_m(X_m) + c =_{ACUIH} X_1 + \cdots + X_m + W$ is solvable if and only if $W \in Terms(\{+, a_1, \ldots, a_m, c\})$.*

Putting together the above two theorems and the fact that *ACUIH*-unification is DEXPTIME-complete (proved in [6]), we may now make the following deduction.

PROPOSITION 3. *$ACUID_l$-unification is decidable in EXPTIME.*

*Proof.* From the cited result of [6] and the two above theorems, we get that any simple $ACUID_l$-unification problem is solvable in EXPTIME. Now any arbitrarily given $ACUID_l$-unification problem can be transformed into a standard form in linear time. Subsequent transformation into a "simple" problem is done by the choice of a deciduous substitution; and the number of possible deciduous substitutions is exponential. □

In order to show that $ACUID_l$-unification is DEXPTIME-complete, it only remains to establish a DEXPTIME lower bound. For this purpose, we propose once again to use the result of [6] mentioned above, by formulating any given *ACUI***h**-unification problem as a suitable instance of an $ACUID_l$-problem. We can do so in a natural manner, as is illustrated by the following example:

EXAMPLE B. Let $f, g$ be two (noncommuting) homomorphisms over an *ACUI*-signature, and consider the following *ACUI***h**-unification problem:

$$f^2g^2a + f^2b + gY = fY + gfX,$$

where $a, b$ are free constants, and $X, Y$ are the unknowns. We then introduce additional free constants $c, d$ and visualize $f$ (resp. $g$) as the homomorphism $h_c$ (resp. $h_d$) of left-multiplication by $c$ (resp, by $d$) in an $ACUID_l$-signature with free constants $a, b, c, d$. From the given *ACUI***h**-problem we then deduce the following $ACUID_l$-problem:

$$c * (c * (d * (d * a))) + c * (c * b) + (d * Y) = (c * Y) + d * (c * X).$$

Now suppose the $ACUID_l$-problem (obtained by our translation) *is* solvable, and let $X', Y'$ be a solution; then substituting for $X', Y'$ in the $ACUID_l$-equation and checking for equality, we get from "pattern-matching" considerations that $X', Y'$ are *sets of right-parenthesized terms* over "$*$" and the constants, where the constants that are in the original problem appear only at the rightmost position. Thus, in our current example we obtain $Y' = (c * X')$ and then $c * (c * X') = c * (c * (d * (d * a))) + c * (c * b)$, so $X' = d * (d * a) + b$. From this $ACUID_l$-solution for the translated problem, we naturally deduce a solution to the *ACUI***h**-problem we started with, by reversing the translation: Replace $h_c$ by $f$ and $h_d$ by $g$, and deduce the solution $X = g^2a + b$ and $Y = fg^2a + fb$.

In other words, *if* the $ACUID_l$-problem obtained through such a translation from a given *ACUI***h**-problem is solvable, *then* we can naturally deduce from its solution a solution for the *ACUI***h**-problem itself; and the reverse assertion holds, too, by reversing the lines of reasoning.

THEOREM 7. *$ACUID_l$-unification is DEXPTIME-hard.*

*Proof.* (The lines of reasoning are those developed in Example B.) Consider any *ACUI***h**-unification problem – where *ACUI***h** is the theory over an *AC*-symbol "+," assumed idempotent with a unit $U = 0$, **h** is any given finite family of *non-commuting* homomorphisms w.r.t. "+," and given free constants. Represent the set of homomorphisms **h** as $\{h_c, h_d, \ldots\}$, indexed by finitely many free additional constants. We consider then an $ACUID_l$-signature for $\{+, *\}$ to which we add the given free ground constants as well as the additional constants $\{c, d, \ldots\}$. Now, the given *ACUI***h**-problem consists in solving equations of the form

$$l_1(X_1) + l_2(X_2) + \cdots + l_n(X_n) + T = r_1(X_1) + r_2(X_2) + \cdots + r_n(X_n) + S,$$

where the unknowns are the $X_i$, the $\{l_i, r_i\}_{i=1..n}$ are *right-parenthesized* strings over the homomorphism symbols $\{h_c, h_d, \ldots\}$, and $S$, $T$ are sets of terms obtained by applying such strings to the given ground constants. To this *ACUI***h**-problem we can naturally associate a unification problem over $ACUID_l$, by replacing each homomorphism of **h** by left-multiplication under "$*$" with its indexing constant.

(Proceeding as in Example B above) One can check that if the associated $ACUID_l$-problem is solvable, then it admits a solution $X'_i, i = 1..n$, where each $X'_i$ is a *set of right-parenthesized terms* over "$*$" and the given constants, these latter appearing only at the rightmost position in these terms; from such a solution one can then deduce naturally a solution to the *ACUI***h**-problem we started with. The converse assertion is just as easy to check. So *ACUI***h**-unification can be seen as an instance of $ACUID_l$-unification. The former problem has been shown to be DEXPTIME-complete in [6]; so the latter is DEXPTIME-hard. □

## 4. The *ACUID*-Unification Problem: Some Special Cases

We turn our attention to the unification problem w.r.t. the following theory that we denote *ACUID*.

$$x + (y + z) \approx (x + y) + z, \qquad x + y \approx y + x,$$
$$x + x \approx x, \qquad x + 0 \approx x, \qquad x * 0 \approx 0, \qquad 0 * x \approx 0,$$
$$x * (y + z) \approx (x * y) + (x * z), \qquad (u + v) * w \approx (u * w) + (v * w).$$

This set of equations can be converted naturally to a convergent rewrite system, modulo the AC-axioms for "+"; every ground term (over any given set of free constants) in normal form w.r.t. this system can be viewed as a *finite set* of terms over "$*$" and the constants: indeed "+" can be viewed as set union. An ACUID-unification problem with free constants is that of solving modulo the above equational theory, a family of equations of the form $\{s_1 = t_1, \ldots, s_k = t_k\}$, where the terms in the equations or the solutions can involve the free constants.

*Note.* When we eliminate the element 0 and drop the equations involving it in *ACUID*, we get the theory that we denote as *ACID*.

### 4.1. UNDECIDABILITY IF "$*$" IS, IN ADDITION, AC OR ASSOCIATIVE

In this section we prove that the *ACUID*-unification problem is undecidable if we assume that the symbol "$*$" is, in addition, either *AC* or associative. The *AC*-case follows easily from our previous results on $ACUIH^C$-unification.

THEOREM 8 (With the above notation). *Unification modulo the theory E obtained by adding the axioms of associativity-commutativity on "$*$" to ACUID is undecidable.*

*Proof* (Similar to that of Theorem 7). Suppose given any $ACUIH^C$-unification problem, where $ACUIH^C$ is the theory over *AC*-symbol "$+$" assumed idempotent, with a unit $U = 0$, $H$ is any given finite family of *commuting* homomorphisms w.r.t. "$+$," and given free ground constants. Represent the set of homomorphisms $H$ as $\{h_c, h_d, \ldots\}$, indexed by finitely many additional free constants. Consider then the theory for $\{+, *\}$ consisting of the equations in $E$ to which we add the given free ground constants as well as the additional constants $\{c, d, \ldots\}$. Now, the given $ACUIH^C$-unification problem consists in solving equations of the form

$$l_1(X_1) + l_2(X_2) + \cdots + l_n(X_n) + T = r_1(X_1) + r_2(X_2) + \cdots + r_n(X_n) + S,$$

where the unknowns are the $X_i$, the $\{l_i, r_i\}_{i=1..n}$ are associative-commutative strings over the homomorphisms $\{h_c, h_d, \ldots\}$, and $S, T$ are sets of terms obtained by applying such strings to the given ground constants. To this $ACUIH^C$-problem we can naturally associate an $E$-unification problem, by replacing the $h_c, h_d, \ldots \in H$ by multiplication under "$*$" with the corresponding indexing constants in $\{c, d, \ldots\}$. Assume that the associated $E$-problem is solvable, and let $X'_i$, $i = 1..n$, be a solution; then from each $X'_i$ one can deduce naturally a set of ground $ACUIH^C$-terms $X_i$, by proceeding as follows. Suppress the "$*$"-symbol from the terms in $X'_i$, and replace the indexing constants $c, d, \ldots$ by the corresponding homomorphism symbol $h_c, h_d, \ldots$ in $H$; then, the following claim is easily checked: The $X_i$, $i = 1..n$, thus obtained, satisfy the $ACUIH^C$-unification problem that we started with; this is *because "$*$" has been assumed AC and the homomorphisms of H commute*. And the reverse claim holds, too. We thus get a contradiction to Theorem 4, where we proved that $ACUIH^C$-unification is undecidable. □

THEOREM 9 (With the above notation). *Unification modulo the theory E obtained by adding the equations of associativity on "$*$" to the theory ACUID is undecidable.*

*Proof.* The proof is by reduction from the so-called modified Post correspondence problem (MPCP). Recall that the formulation of MPCP goes as follows: Given a list of pairs $\{(w_i, w'_i), 0 \leq i \leq n\}$ of nonempty (finite) strings over some alphabet $\Sigma$, it is undecidable in general to determine whether there exist indices

$i_1, \ldots, i_k$, with $0 \leq i_j \leq n$ for all $j = 1..k$, such that

$$w_0 w_{i_1} w_{i_2} \ldots w_{i_k} = w'_0 w'_{i_1} w'_{i_2} \ldots w'_{i_k}.$$

Let $(w_0, w'_0), \ldots, (w_n, w'_n)$ be any given instance of MPCP; we may assume without loss of generality that $w_0 \neq w'_0$. Consider then the $E$-unification problem defined by the following two equations:

$$w_1 * X_1 * \overline{1} + \cdots + w_n * X_n * \overline{n} + w_0 * \# = X_1 + X_2 + \cdots + X_n + V, \quad (1)$$

$$w'_1 * Z_1 * \overline{1} + \cdots + w'_n * Z_n * \overline{n} + w'_0 * \# = Z_1 + Z_2 + \cdots + Z_n + V, \quad (2)$$

where the unknowns are the $X_i$, $Z_i$, and $V$ and the $\#, \overline{1}, \overline{2}, \ldots, \overline{n}$ are some fixed new symbols. Assume this $E$-unification problem to be solvable.

We observe first that not all of the $X_i$ can be 0 in the solution of (1); otherwise $V$ will have to be $w_0 * \#$, so $V$ contains none of the new symbols $\overline{1}, \overline{2}, \ldots, \overline{n}$; but then, all the $Z_i$ in the second equation have to be 0, and $V$ would have to be $w'_0 * \#$, which would contradict the assumption we made above.

Consequently, $V$ must contain words ending with symbols from $\{\overline{1}, \overline{2}, \ldots, \overline{n}\}$. We then look at the solution for equation (1) and observe that words of maximal length over the rhs (right-hand side) of (the solution for) this equation must come from the set $V$ and can be in no instance of any $X_j$ over the rhs: indeed, if $v$ is a word from an $X_j$, then $w_j * v * \overline{j}$ will be an element of some set over the lhs (left-hand side) and so must also be an element over the rhs and necessarily of bigger length than the word $v$.

Let then $v \in V$ be a word of maximal length over the rhs of (1). If $v$ is not equal to $w_0 * \#$, then it has to be of the form $w_i * w * \overline{i}$ for some unique $i$. We may suppose $i = 1$, write $v = w_1 * w * \overline{1}$, and set $X_1 = X'_1 + w$, for a $w \in X_1$, $w \notin X'_1$. Then $v$ has a unique occurrence in both the lhs and rhs of the equation, so we can cancel it from each side. After such a cancellation, we get an equality of the form

$$w_1 * X'_1 * \overline{1} + \cdots + w_n * X_n * \overline{n} + w_0 * \# = X'_1 + X_2 + \cdots + X_n + V',$$

where $V' = (V \setminus \{w_1 * w * \overline{1}\}) + w$.

Now (the solution for) $V'$ is smaller than (that of) $V$ for the multiset ordering over the lengths of words. Thus we can apply an inductive argument to deduce that each element of $V$ has to be a word of the form $w_{i_1} * \cdots * w_{i_n} * w_0 * \# * s$, where $s$ is some string over the symbols $\overline{1}, \ldots, \overline{n}$. A similar reasoning with Equation (2) leads us then to the assertion that each such element of $V$ must also be, at the same time, of the form $w'_{i_1} * \cdots * w'_{i_n} * w'_0 * \# * s$, with the same sequence of indices for the $w'_i$ as for the $w_i$, because of the new symbols introduced $\overline{i}$.

In other words, if our above $E$-unification problem were solvable (with a non-zero value for at least one unknown), then the solution for $V$ contains a solution for the instance of MPCP we started with. Conversely from a solution of MPCP we can build a solution to the $E$-unification problem defined by the above two Equations (1), (2), in terms of sets of terms over "$*$" and the constants. □

## 5. *ACUID*-Unification: The General Case

We assume henceforth that no further laws other than two-sided distributivity over "+" are assumed on the binary symbol "$*$" in *ACUID*. An *ACUID*-unification problem is said to be in *standard form* iff every equation in the problem has one of the following forms (respectively referred to as of type *product, sum,* or *constant*):

$$x = y * z, \qquad u = v + w, \qquad u = c,$$

where $u, v, w, x, y, z$ are variables and $c$ is any constant or 0. A given *ACUID*-unification problem can be reduced to a standard form in more than one manner (through normalization and decomposition steps). Since "+" is idempotent and "$*$" distributes left and right over "+," we may view this *ACUID*-unification problem as a set constraint problem; for example, in the first case, if $y$ and $z$ are interpreted as sets of terms over $*$ and the constants, then $y * z = \{s * t \mid s \in y,\ t \in z\}$.

The set constraints in this context are *with union* only, following the terminology introduced in [9], with the additional restriction that *all sets must be finite*. When the unification problem is *modulo the theory ACID, these sets must also be nonempty*. The problem of satisfiability of set constraints in general, that is, allowing arbitrary sets in the solutions, has been studied intensively over the past decade, in particular in [1, 2, 8, 10, 13, 9], although not all known positive results give a complexity estimate. However, very few results seem to be known for solvability in terms of finite sets or finite nonempty sets. The only result we know of is very general and is based on the $\Sigma$-graph automata of Gilleron, Tison, and Tommasi; see Proposition 13 [13]; by appealing to this result we may formulate our next result, although without any complexity estimate.

PROPOSITION 4. *The AC(U)ID-unification problem is decidable.*

Actually the results of [6] give also a lower bound for the complexity of unification modulo *ACUID*; indeed the proof of Theorem 7 above goes through verbatim for *ACUID* as well.

PROPOSITION 5. *ACUID-unification is DEXPTIME-hard.*

*Remarks 2.* (a) The undecidability results of Theorems 8 and 9 remain obviously true, for unification modulo the theories extending *ACID* by assuming "$*$" to be *AC*, or associative: indeed, decidability in the absence of 0 implies decidability in its presence.

(b) The proof of the DEXPTIME lower bound for *ACUI***h**-unification, as given in [6], makes explicit use of the "unit" 0 and therefore cannot be carried through as such for $ACID_l$ or *ACID*. An argument adapting its lines of proof to deduce a DEXPTIME lower bound for $ACID_l$- or *ACID*-unification is presented in the Appendix.

We show in the following section that *ACID*-unification is NEXPTIME-decidable. Our approach is based on solving the equations of any problem $\mathcal{P}$ given in standard form, in an "incremental" manner with the help of a graph representing the dependency relations between the variables of $\mathcal{P}$. We begin by "pruning" the problem, in order to eliminate syntactically some unsolvable cases, but more important also to introduce a notion of height equivalence between some of the variables of the "sum"-equations. Part of the pruning operation is in a sense comparable to the occur_check of standard unification.

## 6. *ACID*-Unification Is NEXPTIME-Decidable

We assume that all our *ACID*-unification problems are given in standard form.

### 6.1. PRUNING AN *ACID*-UNIFICATION PROBLEM

Pruning an *ACID*-unification problem $\mathcal{P}$ (given in standard form) is based on a *dependency graph* associated to $\mathcal{P}$. The variables $\{u, v, w, x, y, z, \ldots\}$ appearing in $\mathcal{P}$ form the nodes of the graph; its edges are directed and labeled by one of the two symbols "$\{*, \supset\}$" as follows: For any two variables $u$, $v$, $(u, v)$ is an edge labeled with "$*$" (resp. $\supset$) iff there is an equation of the form $u = v * w$ or $u = w * v$ (resp. $u = v + w$) in our *ACID*-problem. The arcs of the graph are called *dependency arcs*. A variable of the problem from which there is no outgoing edge is called an *end-variable* of $\mathcal{P}$.

If $u$, $v$ are any two variables in $\mathcal{P}$, we set $u \succ v$ iff there is a directed path on the graph from $u$ to $v$. We set $u \succ_* v$ iff at least one edge on such a directed path is labeled with "$*$," and we set $u \geq v$ iff either $v = u$ or *all* the edges on a path from $u$ to $v$ are labeled only with "$\supset$." If $u \geq v$, we say that $v$ is a *subset-variable* of $u$; if there is a path from $u$ to $v$ with only "$*$"-edges all along, we say that $v$ is a *factor-variable* of $u$. Pruning the problem $\mathcal{P}$ consists of carrying out four operations on $\mathcal{P}$ or its associated dependency graph, with a view to diminishing the number of variables and eliminating syntactically some of the unsolvable cases.

*Prune-0*: If there is a variable $u$ in $\mathcal{P}$ and a path on the graph from $u$ to itself, along which we have only "$\supset$"-edges, then replace all the variables at the intermediary nodes on the path by $u$.

For instance, if $X_1 = X_2 + X_3$, $X_2 = U + V$ and $U = X_1 + W$ are in $\mathcal{P}$, then we replace everywhere the variables $X_2$ and $U$ by $X_1$. This pruning step eliminates part of the redundancy in $\mathcal{P}$.

*Prune-1*: Return "**Fail**":

- if $u \succ_* v$ for some $v$, and $u = a$ is in $\mathcal{P}$;
- if there is a variable $u$ such that $u \succ_* u$;
- if there exist nodes $u$, $v$, $w$ such that $u \geq v$ *and* $u = v * w$.

The first check ensures that the problem $\mathcal{P}$ is "coherent" with its "constant"-equations; the two others correspond to conditions obviously necessary for the variables concerned to have *finite, nonempty* set solutions: a subset-variable of $u$ *cannot* also be a factor-variable of $u$.

EXAMPLE 1a. Prune-1 returns "Fail" on the following three problems:

(i) $z = a,\ z = x + y,\ y = u * v,$
(ii) $z = x + y,\ x = z * v,$
(iii) $z = x + y,\ z = x * u.$

Next, if the problem $\mathcal{P}$ admits a solution in terms of finite sets, then for every "sum"-equation $Z = X + Y$ in $\mathcal{P}$, terms of maximal height in $Z$ have to be in $X$ or in $Y$; hence we cannot have both $Z \succ_* X$ and $Z \succ_* Y$. This gives the following check on $\mathcal{P}$:

*Prune-2*: Return "**Fail**":

if there is a "sum"-equation $Z = X + Y$ in $\mathcal{P}$ such that $Z \succ_* X$ and $Z \succ_* Y$.

The final pruning step consists in "locating" the variable(s) to the right of every "sum"-equation $Z = X + Y$ in $\mathcal{P}$, which can contribute terms of maximal height to $Z$ in a *finite, nonempty* solution for $\mathcal{P}$. For doing that, we consider *all possible* relations $\sim$ between the set variables such that for every "sum"-equation $Z = X + Y$ in $\mathcal{P}$, we have $X \sim Z$ or $Y \sim Z$ or both, nonexclusively; we refer to these as *height relations*. If $\sim$ is such a given relation, then for every $Z = X + Y$ in $\mathcal{P}$, if $X \sim Z$ (resp. $Y \sim Z$), then we introduce a $\sim$-arc from $X$ (resp. from $Y$) to $Z$.

*Prune-3*: Choose *nondeterministically* a height relation $\sim$ on the set of variables of $\mathcal{P}$. If there is a node $X$ on the dependency graph of $\mathcal{P}$ such that

there is a path from $X$ to $X$ with $\supset$, $*$, or $\sim$-edges, and
*at least one of them is a* "$*$"-edge, then return "**Fail**."

EXAMPLE 1b. The following *ACID*-problem does not pass "Prune-3":

$$V = X + Y, \qquad V = W' + U', \qquad V = W'' + U'',$$
$$W' = X * Z', \qquad W'' = Y * Z''.$$

The failure is caused by the first "sum"-equation; indeed, if we assume $X \sim V$, we get the cycle $X \sim V \geq W' \succ_* X$; while if we assume $Y \sim V$, we get the cycle $Y \sim V \geq W'' \succ_* Y$.

We say that *pruning fails* on $\mathcal{P}$ iff "Fail" is the value returned by one of the above four pruning operations. The problem $\mathcal{P}$ is said to be in *pruned* form otherwise. In such a case, we *choose nondeterministically a height relation* "$\sim$" *for which Prune-3 does not fail*, and we introduce the corresponding "$\sim$"-arcs on the dependency graph. We henceforth assume that such arcs are taken into account by the strict ordering "$\succ_*$." The following proposition is then immediate.

PROPOSITION 6. *Let $\mathcal{P}$ be an ACID-unification problem given in standard form. Then:*

(i) *Pruning the problem $\mathcal{P}$ can be performed in time NP w.r.t. the number of variables in $\mathcal{P}$.*
(ii) *If $\mathcal{P}$ is in pruned form, then the dependency graph is "$*$"-loop free: there is no node $u$ such that $u \succ_* u$.*

The dependency graph being "$*$"-loop free for a pruned problem, we can associate a *height* and a *co-height* (also called *depth*) to any of its variables w.r.t. any given height relation "$\sim$" that passes Prune-3:

DEFINITION 4. Let $\mathcal{P}$ be a pruned *ACID*-problem with a given height relation $\sim$; and let $X$ be any given variable in $\mathcal{P}$. We then define the following:

- $ht(X) =$ *height* of $X$ in $\mathcal{P} =$ the maximal integer $m$ such that there is a path on the dependency graph from $X$ to some end-variable of $\mathcal{P}$ with $m$ "$*$"-edges.
- $Ht(\mathcal{P}) =$ *height* of the problem $\mathcal{P} = \max\{ht(Y) \mid Y$ is a variable in $\mathcal{P}\}$.
- $coht(X) =$ *co-height* (or *depth*) of $X$ in $\mathcal{P} = Ht(\mathcal{P}) - ht(X)$.

Please note: $Ht(\mathcal{P})$ is independent of any height relation, but $ht(X)$ and $depth(X)$ depend in general on the given "$\sim$."

EXAMPLE 1c. The problem $\mathcal{P}$: $V = X + Z$, $V = W + U$, $W = X * Y$, is in pruned form. Indeed $V \sim Z$ in the first "sum"-equation, while $W \sim V$ and $U \sim V$ in the second; and the dependency graph satisfies the "$*$"-loop-free condition. We get $ht(X) = 0 = ht(Y)$, and $ht(W) = 1 = ht(V) = ht(Z) = ht(U)$. So $Ht(\mathcal{P}) = 1$; and $X$ and $Y$ are the only variables at nonnull depth for this problem.

We shall see below that this problem *is* actually solvable.

## 6.2. A NEXPTIME-ALGORITHM

In this section we present a NEXPTIME-algorithm for solving any pruned *ACID*-unification problem $\mathcal{P}$. Let $h = Ht(\mathcal{P})$ be the height of the problem $\mathcal{P}$ as defined above. We have $h \leq n =$ the number of variables in $\mathcal{P}$, thanks to Prune-0 and Prune-1. The following observation is also a direct consequence of *Prune-1*.

LEMMA 4. *If $\mathcal{P}$ contains a "constant"-equation $X = a$, then for any height relation $\sim$, the depth of $X$ in $\mathcal{P}$ is $h = Ht(\mathcal{P})$.*

Let $\mathcal{C}$ be the set formed by all the ground constants appearing in $\mathcal{P}$, plus an additional dummy constant that we denote $\Theta$. For every $k \in \{0, \ldots, h\}$, define $\mathcal{T}_k$ as the set of all ground terms formed of "$*$" and the constants in $\mathcal{C}$, of height at most $(h - k)$; in particular we have $\mathcal{T}_h = \mathcal{C}$. For (any given height relation $\sim$ and)

any $k \in \{0, \ldots, h\}$, we define $\mathcal{P}_k$ as the *ACID*-problem formed of the equations in $\mathcal{P}$ that involve only the variables at depth $\geq k$ in $\mathcal{P}$; in particular $\mathcal{P}_0 = \mathcal{P}$.

Let $\mathcal{H}$ be the set of all possible height relations on $\mathcal{P}$ that pass Prune-3. The algorithm given below starts by guessing a height relation $\sim$ in $\mathcal{H}$ and then follows a downward iterative loop from $k = h = Ht(\mathcal{P})$, to $k = 0$.

ALGORITHM $\mathcal{A}$.

Choose: Guess non-deterministically a height relation $\sim$ in $\mathcal{H}$.
Init: Set $k = h = Ht(\mathcal{P})$.
Step 1: Guess non-deterministically the terms in $\mathcal{T}_k$
that can be added to the set-variables of $\mathcal{P}_k$
such that the equations in $\mathcal{P}_k$ are all satisfied.
Step 2a: Case $k > 0$: Propagate to variables of $\mathcal{P}_{k-1}$ the
terms of $\mathcal{T}_{k-1}$ using the 'product'-equations
in $\mathcal{P}_{k-1}$. Decrement $k$ by 1. GOTO Step 1.
Step 2b: Case $k = 0$: RETURN '*Success*'.

As concerns the complexity of this algorithm $\mathcal{A}$, we observe the following:

- Guessing a height relation in $\mathcal{H}$ can be done in time NP w.r.t. the number of variables in the problem $\mathcal{P}$ (cf. Proposition 6).
- Guessing incrementally the sets of ground terms assigned to the variables of $\mathcal{P}$, so that the equations in $\mathcal{P}$ are all satisfied, can be done in NEXPTIME w.r.t. the number of variables in $\mathcal{P}$.

The NEXPTIME estimate above is due to the fact that the number of ground terms of height at most $h$ is bounded exponentially w.r.t. the number $n$ of variables in the problem $\mathcal{P}$. The algorithm $\mathcal{A}$ is obviously sound. We shall prove that it is also complete, but before doing that we show how it works on a couple of examples.

EXAMPLE 1c (contd.). We get back to the pruned problem $\mathcal{P}$: $V = X + Z$, $V = W + U$, $W = X * Y$. We saw above that $Ht(\mathcal{P}) = 1$; $X$ and $Y$ are the only variables at depth 1 for this problem, the others being at depth 0.

The above algorithm starts thus with $k = 1$ and $\mathcal{T}_1 = \{\Theta\}$. At Step 1a, we assign $\Theta$ to $X$ and $Y$; at Step 2a, $W$ gets the term $\Theta * \Theta$, and then $k$ gets decremented to 0. We go back to Step 1a with $\mathcal{T}_0 = \{\Theta, \Theta * \Theta\}$ and finish with the successful guess $U = \{\Theta\}$, $Z = \{\Theta * \Theta\}$, $V = \{\Theta, \Theta * \Theta\}$.

Here is a "more complete" example where the problem passes pruning but where the algorithm detects unsolvability.

EXAMPLE 2. Let $\mathcal{P}$ be the following *ACID*-problem:

(i) $X_1 = a$, (ii) $X_2 = b$, (iii) $X_3 = X_1 + X_2$,
(iv) $X_4 = X_1 * X_2$, (v) $X_5 = X_3 * X_2$, (vi) $X_6 = X_3 + X_4$,
(vii) $X_6 = X_1 + X_5$.

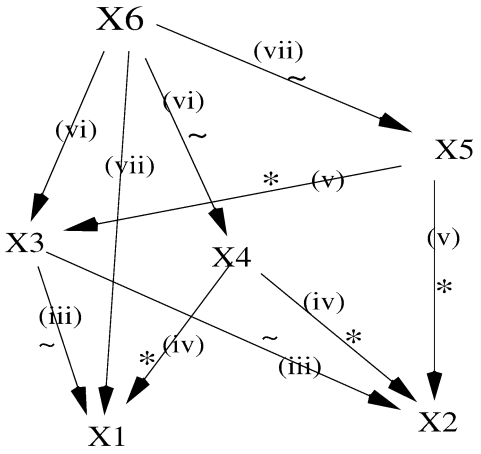


*Figure 1.* Dependency graph for the problem in Example 2.

This problem is pruned, and only one height relation passes Prune-3: $X_1 \sim X_3$, $X_2 \sim X_3$, $X_4 \sim X_6$, $X_5 \sim X_6$. See Figure 1 for its dependency graph: each edge is numbered with that of the corresponding equation; the "$*$"-edges are labeled explicitly, while the "$\supset$"-edges are left implicit; and the "$\sim$"-arcs are supposed to go in the opposite direction w.r.t. the arrows.

For the heights we get $h(X_1) = 0 = ht(X_2) = ht(X_3)$; and $ht(X_4) = 1 = ht(X_5) = ht(X_6)$. So $Ht(\mathcal{P}) = 1$, and the former are the variables at depth 1.

At the first step of iteration, the algorithm gives $X_1 = \{a\}$, $X_2 = \{b\}$, $X_3 = \{a, b\}$, which satisfies the subproblem of $\mathcal{P}$ formed of equations (i)–(iii) involving only the variables at depth 1. At the second iteration, we propagate the values via the "product"-equations and get $X_4 = \{a * b\}$, $X_5 = \{a * b, b * b\}$, and subsequently $X_6 = \{a * b, a, b\}$ from equation (vi), while $X_6 = \{a * b, b * b, a\}$ from equation (vii). The algorithm fails, since no other height relation is available.

### 6.3. THE ALGORITHM $\mathcal{A}$ IS COMPLETE

Let $\mathcal{P}$ be any *ACID*-unification problem given in standard (pruned) form, and let $n$ be the number of its variables. Suppose that $\mathcal{P}$ is solvable, and let $X_i = S_i, i = 1..n$, be a given solution of $\mathcal{P}$. Observe first that this given solution determines a unique set of height relation arcs $\sim$ on the dependency graph of $\mathcal{P}$. Our objective in this subsection is to prove that the algorithm of the previous subsection, when run on $\mathcal{P}$ with this uniquely determined height relation as $\mathcal{H}$, will end up with success. As above, we denote by $h$ the height of the problem $\mathcal{P}$.

The following notions will be used in proving this completeness. Let $t$ be any ground term over "$*$" and the set $\mathcal{C}$ of constants (cf. above); by height of $t$ we mean the usual height of the term $t$, denoted as $ht(t)$; if $t'$ is any subterm of $t$,

we set $depth_t(t') = ht(t) - ht(t')$; we then say that $t'$ is a subterm of $t$ at depth $depth_t(t')$.

DEFINITION 5. Let $k$ be any given integer such that $0 \leq k \leq h$. Then, for any ground term $t$ over "$*$" and the constants, $t_k$ is the ground term of height $\leq k$ obtained by replacing *every* subterm of $t$ at depth $k$ by the constant $\Theta$.

For any set $\mathcal{T}$ of ground terms over "$*$" and the constants, we set $\mathcal{T}_k = \{t_k \mid t \in \mathcal{T}\}$.

LEMMA 5. *Let $X, Y, Z$ be any finite nonempty sets of ground terms over "$*$" and the constants. Then*

(i) *If $Z = X + Y$, then for any $k \in \{0, \ldots, h\}$ we have $Z_k = X_k + Y_k$.*
(ii) *If $Z = X * Y$, then for any $k \in \{1, \ldots, h\}$ we have $Z_k = X_{k-1} * Y_{k-1}$.*

*Proof.* Only the second assertion needs (perhaps) to be justified. Observe first that the depth of a subterm $t'$ of any given ground term $t$ is the length of the path from the root node of $t$ to the root node of $t'$. Now, any $t \in Z$ is of the form $u * v$, with $u \in X$, $v \in Y$; thus a subterm of $t$ at depth $k \geq 1$ must be a subterm either of $u$ or of $v$ at depth $k - 1$. Hence, the term $t_k$ (by definition the ground term obtained by setting *all* the subterms of $t$ at depth $k$ to $\Theta$) is none other than $u_{k-1} * v_{k-1}$. (Note that if a ground term $s$ is such that $k > ht(s)$, then $s_k = s$, by definition.) □

PROPOSITION 7. *Let $\mathcal{P}$ be an ACID-unification problem in standard (pruned) form. Suppose $\mathcal{P}$ is solvable, and let $\sigma$ be a solution given as finite nonempty sets of ground terms for the variables in $\mathcal{P}$. For any variable $X_i$ in $\mathcal{P}$, let $h_i$ be its height in $\mathcal{P}$ w.r.t. the unique height relation induced by the given solution (cf. Definition 4), and define a ground substitution $\hat{\sigma}$ such that, for every $i \in 1..n$, $\hat{\sigma}(X_i) = \sigma(X_i)_{h_i}$. We then have the following:*

(ii) *The substitution $\hat{\sigma}$ satisfies the equations of the problem $\mathcal{P}$.*
(ii) *For every $i \in \{1, \ldots, n\}$, $\hat{\sigma}(X_i)$ is a nonempty finite set.*

*Proof.* We assume (as we may) that the height $h$ of the problem $\mathcal{P}$ is at least 1. In order to prove assertion (i), it suffices to apply Lemma 5 above, in conjunction with the following fact: If $\mathcal{P}$ has a "constant"-equation $X = a$, then $\hat{\sigma}(X) = \sigma(X) = \{a\}$, since the *depth* of $X$ in $\mathcal{P}$ is $h > 0$ (cf. Lemma 4). Assertion (ii) then gets proved, too, under the same argument via (i). □

COROLLARY 7.1. *Algorithm $\mathcal{A}$ of Section 6.2 is complete.*

*Proof.* This follows by the above proposition: indeed for any variable $X$ in $\mathcal{P}$ at maximal depth, $\hat{\sigma}(X)$ contains only the constants from $\mathcal{C}$; so one of the branches of the search mechanism will end up with success, by adding the ground terms of $\hat{\sigma}(X_i)$ to the set variable $X_i$, for each $i \in \{1, \ldots, n\}$. □

THEOREM 10. *ACID-unification and ACUID-unification are decidable in NEXPTIME.*

*Proof.* For *ACID*-unification, the assertion follows from what we just established. As concerns *ACUID*-unification, one can reason as follows: Choose nondeterministically the set of variables that are assigned the value $U = 0$, and solve for the others using the NEXPTIME-algorithm just described for *ACID*. □

Note that we have already observed that *ACUID*-unification and *ACID*-unification are both DEXPTIME-hard; cf. Proposition 5 above and Proposition 8 in the Appendix.

## 7. Conclusion

Distributivity (along with other axioms) has been addressed in many papers, the earliest being the undecidability results of Siekmann and Szabo [21] for the theories *AD*, *ACD*, and *ACUD*. Unification modulo one-sided distributivity was shown to be decidable by Tiden and Arnborg [23]. When associativity and commutativity are added (i.e., the theory $ACUD_l$, with no idempotence assumption for the *ACU*-symbol), the problem is again undecidable [15]. *D*-unification, which remained an open problem for a long time, was finally settled by Schmidt-Schauss, who showed it to be decidable [18, 19].

Our concern in this paper has been the unification problems w.r.t. the theories *ACUID* and $ACUID_l$, which seem to be of practical interest in the analysis of programs modeled as process algebras, thanks to the added idempotence assumption. Both the problems are DEXPTIME-hard; we have shown unification modulo $ACUID_l$ (resp. *ACUID*) to be decidable in EXPTIME (resp. in NEXPTIME). For $ACUID_l$, the results obtained are by reduction to the *ACUIH*-unification problem – where $H$ is a finite set of noncommuting homomorphisms over the *ACUI*-symbol "+" – which was treated completely in [6]. Unification modulo *ACUID*, where "$*$" is assumed two-sided distributive over "+," cannot be handled in such a manner; our lines of reasoning are based on interpreting the problem as a special class of set constraints.

We have also shown that unification *becomes undecidable* if the axioms of associativity-commutativity, or just of associativity, on the distributing symbol "$*$" are added on to $AC(U)ID$. However, over the theory obtained by adding the *commutativity* axiom of "$*$" to $AC(U)ID$, unification remains decidable: the algorithm presented in Section 6 of this paper seems to remain valid modulo the commutativity of "$*$." The approach presented in [4], based on the construction of a labeled dag automaton, LDA, remains valid, too: it suffices to check the progression conditions on the states of the LDA modulo the commutativity of "$*$." We also mention that unification remains decidable modulo the theory $PAC(U)ID$ obtained by adding monadic function symbols (*prefixes*) to the $AC(U)ID$-signature: indeed, it suffices to add suitable unary transitions between the states of the LDA in the

approach of [4], corresponding to every monadic function symbol. We believe that unification modulo $PAC(U)ID$ can be used as a tool for the analysis of information flow between processes, when they are modeled as (value-passing) process algebras.

## Appendix: The DEXPTIME Lower Bound without $U = 0$

PROPOSITION 8. *The unification problem over ACID or $ACID_l$ is DEXPTIME-hard.*

*Proof.* We show that the proof of the DEXPTIME-hardness of unification modulo $ACUI\mathbf{h}$, as given in [6], can be adapted so as to get rid of the role played by the unit element $U = 0$ of the theory.

Recall that this proof relies on Seidl's result [20] that if $A_i, i = 1..k$ are given finitely many DTTA's (deterministic top-down tree automata), then deciding the emptiness of their intersection is DEXPTIME-hard. In [6], an $ACUIH$-equation $E_i$ is constructed for each of the $A_i$; these equations share exactly one common variable $X$. Solutions to any equation $E_i$ are assignments of finite unions of "run tree sets" constructed from finite trees accepted by the corresponding automaton $A_i$; the unique common variable $X$ serves to glue these different assignments together into a single one, whenever there is a common solution for *all* the equations $E_i, i = 1..k$. A single global equation combining all these equations is defined such that it is solvable iff there is a *common tree t* accepted by *all* the $A_i, i = 1..k$, in which case the run tree set derived from $t$ is assigned to this unique common variable $X$; if the acceptance of this common tree $t$ by any of the $A_i$ does not involve all the transitions in the DTTA $A_i$, then the value assigned to this variable $X$ is $U = 0$. We propose to restructure this reasoning in such a manner that the solvability of the global equation amounts to the existence of *one run tree set* common to the assignments for the different variables (which is weaker than asking for a common accepted tree $t$).

First we show that Seidl's result holds even if we restrict to signatures with *one* binary symbol $g$, *one* unary symbol $s$, and *one* constant '#'. The key idea is this: suppose have a set of binary function symbols $\{g_1, \ldots, g_n\}$. Then we can replace each $g_i$ with $\lambda(x, y).s(s(\ldots s(g(x, y))))$, with $i$ occurrences of the symbol $s$. (In other words, the term is $s^i(g(x, y))$.) A DTTA on the original alphabet can thus be converted into a DTTA on the new alphabet in polynomial time.

Next, we form the equations $E_i$ for each DTTA $A_i$ as in [6] and following the notation therein, except that the variables are all *disjoint*:

$$\{q_f\}\cdot X_i \cup \bigcup_{(q,g)\in Q\times\Sigma} \{q\}\cdot {X^i}_{(q,g)}$$
$$= \{q_0\} \cup \bigcup_{\delta(q,g)=(q_1,\ldots,q_k)} \{q_1 1 g, \ldots, q_k k g\}\cdot {X^i}_{(q,g)}.$$

(Recall that the $\{q_f\}\cdot X_i$ in each of these equations $E_i$ will be "storing" finite unions of run tree sets.) We then form an additional DTTA $A$ with $q_f$ as its **only** state and with the transitions $\delta(q_f, g) := (q_f, q_f)$, $\delta(q_f, s) := q_f$. We form an equation for this DTTA, too:

$$\{q_f\} \cdot Z \cup q_f \cdot X_{(q_f,g)} \cup q_f \cdot X_{(q_f,s)} \\ = \{q_f\} \cup \{q_f 1g, q_f 2g\} \cdot X_{(q_f,g)} \cup \{q_f 1s\} \cdot X_{(q_f,s)}.$$

Then we add the following equations, which express the conditions that $\{q_f\} \cdot Z$ is a subset of all the other $\{q_f\} \cdot X_i$, for the variables $X_i$ coming from the various equations $E_i$, $i = 1..k$:

$$\{q_f\} \cdot X_i = \{q_f\} \cdot Z \cup \{q_f\} \cdot X_i.$$

The rest of the reasoning in [6] can now be carried through: we can solve all these equations simultaneously with nonempty run tree sets for the variables if and only if there is a common run tree set that can be assigned to the variable $Z$, that is, iff the DTTAs $A_i$, $i = 1..k$ and $A$ have a nonempty intersection. □

## References


1. Aiken, A. and Wimmers, E.: Solving systems of set constraints, in *Proc. 7th IEEE Symposium on Logic in Computer Science (LICS'92)*, 1992, pp. 329–340.
2. Aiken, A., Kozen, D., Vardi, M. and Wimmers, E.: The complexity of set constraints, in *Proc. Conf. CSL'93*, EACSL, September 1993, pp. 1–18.
3. Anantharaman, S., Narendran, P. and Rusinowitch, M.: Unification over ACUI plus distributivity/homomorphisms, in *Proc. CADE-19*, LNAI 2741, Springer-Verlag, pp. 442–458.
4. Anantharaman, S., Narendran, P. and Rusinowitch, M.: *ACID*-unification is NEXPTIME-decidable, in *Proc. MFCS'03*, LNCS 2747, Springer-Verlag, pp. 169–179.
5. Baader, F.: Unification in commutative theories, *J. Symbolic Comput.* **8** (1989), 479–497.
6. Baader, F. and Narendran, P.: Unification of concept terms in description logics, *J. Symbolic Comput.* **31**(3) (2001), 277–305.
7. Baader, F. and Schulz, K. U.: Unification in the union of disjoint equational theories: Combining decision procedures, in *Proc. 11th Conference on Automated Deduction (CADE-11)*, Saratoga Springs (NY), LNAI 607, Springer-Verlag, 1992, pp. 50–65.
8. Bachmair, L., Ganzinger, H. and Waldmann, U.: Set constraints are the monadic class, in *Proc. 8th IEEE Symposium on Logic in Computer Science (LICS'93)*, 1993, pp. 75–83.
9. Charatonik, W. and Podelski, A.: Set constraints with intersection, in *Proc. 12th IEEE Symposium on Logic in Computer Science (LICS'97)*, Warsaw, 1997, pp. 362–372. (To appear in *Information and Computation*.)
10. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S. and Tommasi, M.: Tree automata techniques and applications, `http://www.grappa.univ-lille3.fr/tata/`
11. Focardi, R.: Analysis and automatic detection of information flows and network systems, Doctoral Thesis, Technical Report UBLCS-99-16, University of Bologna, July 1999.
12. Gilleron, R., Tison, S. and Tommasi, M.: Solving systems of set constraints using tree automata, in *Proc. STACS'93*, LNCS 665, Springer-Verlag, pp. 505–514.
13. Gilleron, R., Tison, S. and Tommasi, M.: Set constraints and tree automata, *Inform. and Comput.* **149** (1999), 1–41 (see also Tech. Report IT 292, Laboratoire-LIFL, Lille, 1996).

14. Minsky, M.: *Computation: Finite and Infinite Machines*, Prentice-Hall, London, 1972.
15. Narendran, P.: On solving linear equations over polynomial semirings, in *Proc. 11th Annual Symp. on Logic in Computer Science*, July 1996, pp. 466–472.
16. Narendran, P. and Rusinowitch, M.: Any ground associative-commutative theory has a finite canonical system, *J. Automated Reasoning* **17** (1996), 131–143.
17. Nutt, W.: Unification in monoidal theories, in M. Stickel (ed.), *Proc. CADE-10*, LNAI 449, Springer-Verlag, pp. 618–632.
18. Schmidt-Schauss, M.: Decidability of unification in the theory of one-sided distributivity and a multiplicative unit, *J. Symbolic Comput.* **22**(3) (1997), 315–344.
19. Schmidt-Schauss, M.: A decision algorithm for distributive unification, *Theoret. Comput. Sci.* **208**(1–2) (1998), 111–148.
20. Seidl, H.: Haskell overloading is DEXPTIME-complete, *Inform. Process. Lett.* **52**(2) (1994), 57–60.
21. Siekmann, J. and Szabo, P.: The undecidability of $D_A$-unification problem, *J. Symbolic Logic* **54**(2) (1989), 402–414.
22. Talbot, J.-M., Devienne, Ph. and Tison, S.: *Generalized Definite Set Constraints*, In CONSTRAINTS: An International Journal 5(1–2): 161–202, 2000.
23. Tiden, E. and Arnborg, S.: *Unification Problems with One-sided Distributivity*, Journal of Symbolic Computation 3(1–2): 183–202, 1987.