On the Complexity of the Tiden-Arnborg Algorithm for Unification modulo One-Sided Distributivity

Paliath Narendran[*]
University at Albany–SUNY
College of Computing and Information
Computer Science Department
dran@cs.albany.edu

Andrew Marshall[†]
University at Albany–SUNY
College of Computing and Information
Computer Science Department
marshall@cs.albany.edu

Bibhu Mahapatra
New York State
Education Department
bmahapat@mail.nysed.gov

We prove that the Tiden and Arnborg algorithm for equational unification modulo *one-sided* distributivity is not polynomial time bounded as previously thought. A set of counterexamples is developed that demonstrates that the algorithm goes through exponentially many steps.

1 Introduction

Equational unification is central to automated deduction and its applications in areas such as symbolic protocol analysis. In particular, the unification problem for the theory *AC* ("Associativity-Commutativity") and its extensions *ACI* ("AC plus Idempotence") and *ACUI* ("ACI with Unit element") have been studied in great detail in the past. Distributivity (of one binary operator over another) has received less attention comparatively. Some significant results have been obtained such as Schmidt-Schauss' breakthrough decidability result [7] for unification modulo the theory of two-sided distributivity

$$\begin{aligned} x \times (y+z) &= (x \times y) + (x \times z) \\ (y+z) \times x &= (y \times x) + (z \times x) \end{aligned}$$

Other works include [4, 3].

One of the earliest papers that considered a subproblem of this is by Tiden and Arnborg [8]. They present an algorithm for equational unification modulo a *one-sided* distributivity axiom:

$$x \times (y+z) = x \times y + x \times z$$

This unification problem has recently been of interest in cryptographic protocol analysis since many cryptographic operators satisfy this property: for instance, modular exponentiation (used in the RSA and El Gamal public key algorithms) distributes over modular multiplication. Indeed, many electronic election protocols rely on the property of "homomorphic encryption" where encryption distributes over some other operator. (A new algorithm for this unification problem, using a novel approach, is given in [6, 1].)

Our goal in this paper is to analyze the Tiden-Arnborg algorithm. We prove that the algorithm is not polynomial time bounded as claimed in the Tiden-Arnborg paper. A set of counter examples is outlined that demonstrates that the present algorithm goes through exponentially many steps.

[*]Partially supported by the NSF grants CNS-0831209 and CNS-0905286
[†]Partially supported by the NSF grants CNS-0831209 and CNS-0905286

1.1 The Tiden-Arnborg Algorithm

We present a very brief description of the algorithm of Tiden and Arnborg using deduction (inference) rules. First of all, it should be pointed out that what they consider is the *elementary* unification problem [2], where the terms can only contain symbols in the signature of the theory and variables. (Thus free constants and free function symbols are not allowed.) Hence we can assume without loss of generality that the input is given as a set of equations where each equation is in one of the following forms:

$$X =^? Y,\ X =^? Y + Z,\ \text{and}\ X =^? Y \times Z$$

The key steps in the algorithm can be described by the following deduction rules:

(a) $$\frac{\{U =^? V\} \uplus \mathscr{EQ}}{\{U =^? V\} \cup [V/U](\mathscr{EQ})} \quad \text{if } U \text{ occurs in } \mathscr{EQ}$$

(b) $$\frac{\mathscr{EQ} \uplus \{U =^? V \times W,\ U =^? X \times Y\}}{\mathscr{EQ} \cup \{U =^? V \times W,\ V =^? X,\ W =^? Y\}}$$

(c) $$\frac{\mathscr{EQ} \uplus \{U =^? V + W,\ U =^? X + Y\}}{\mathscr{EQ} \cup \{U =^? V + W,\ V =^? X,\ W =^? Y\}}$$

(d) $$\frac{\mathscr{EQ} \uplus \{U =^? V \times W,\ U =^? X + Y\}}{\mathscr{EQ} \cup \{U =^? V \times W,\ W =^? W_1 + W_2,\ X =^? V \times W_1,\ Y =^? V \times W_2\}}$$

The W_1, W_2 in rule (d) are fresh variables and $\uplus$ is *disjoint union*. Furthermore, rule (d) (the "splitting rule") is applied only when the other rules cannot be applied. A set of equations is said to be *simple* if and only if none of the rules (a), (b) and (c) can be applied to it. In other words, in a simple system, no variable can occur as the left-hand side in more than two equations. A *sum transformation* is defined as a binary relation between two simple systems S_1 and S_2, where S_2 is obtained from S_1 by applying rule (d), followed by repeated exhaustive applications of rules (a), (b) and (c). Clearly, a sum transformation is applicable if and only if some variable occurs as the left-hand side in more than one equation.

Detection of failure is done using a kind of "extended occur-check" using two graph based data structures. We repeat the definitions of the graph structures and give a sketch of the algorithm presented in Tiden and Arnborg [8] for the convenience of the reader.

Definition 1.1. The *dependency graph* of a simple system, Σ, is an edge colored, directed multi-graph. It has as vertices the variables of Σ. For an equation $x = y + z$ in Σ it has an l_+-colored edge (x, y) and an r_+-colored edge (x, z). An equation $x = y \times z$ similarly generates two edges with colors $l_\times$ and $r_\times$.

Definition 1.2. The *sum propagation graph* of a simple system Σ is a directed simple graph. It has as vertices the equivalence classes of the symmetric, reflexive, and transitive closure of the relation defined by the $r_\times$-edges in the dependency graph of Σ. It has an edge (V, W) iff there is an edge in the dependency graph from a vertex in V, to a vertex in W with color l_+ or r_+.

The dependency graph structure is sufficient for finding all the occur-check like errors that may develop as the algorithm works with the system of equations. The propagation graph is needed to detect non-unifiable systems that cause infinitely many applications of the splitting rule (d). An example of this type of system is the following two equations:

$$Z =^? V_2 + V_3, Z =^? V_1 \times V_3.$$

These types of systems are shown not to have a unifier and as they will never produce a cycle in the dependency graph, the propagation graph is needed.

Tiden and Arnborg give a polynomial time procedure for producing a simple system of equations form an initial set of equations. We sketch their unification algorithm from the starting point of an initial simple system.

Algorithm 1 UNIFY [8]

```
Require: Simple system $\Sigma_1$.
  $k := 1$
  while The sum transformation can be applied do
    If either the dependency or propagation graph contains a cycle, then stop with failure.
    Using the sum transformation compute $\Sigma_{k+1}$
    $k := k+1$
  end while
  Compute the most general unifier (mgu) by back substitution.
```

It is shown that if a system is not unifiable it will, after finitely many applications of the sum transformation, produce a cycle in one of the graphs. It is also shown that if a system is unifiable then the algorithm will produce the *mgu*.

In the next section we present a family of unifiable systems that produce no cycles in either graph, but require exponentially many applications of the sum transformation.

2 Counterexamples

We present a family of *unifiable* simple systems on which the Tiden-Arnborg algorithm runs in exponential time. For ease of exposition, we only use the letters T, x and y for variables, along with subscripts for x and y which are strings over the alphabet $\{1,2\}$.

Definition 2.1. Let EQ be a subset of the simple system defined as follows: all multiplications are of the form $x_i =^? T \times y_j$ (or $y_j =^? T \times x_i$) where T is a unique variable and all additions are of the form $x_i =^? x_{i1} + x_{i2}$ or $y_i =^? y_{i1} + y_{i2}$.

As the left variable of the multiplication operation will not effect the complexity result we use the unique variable T in this position. This makes the proof simpler. Thus the splitting rule (d) above can be viewed as

$$\frac{\mathscr{EQ} \uplus \{U_i =^? T \times W_j,\ U_i =^? U_{i1} + U_{i2}\}}{\mathscr{EQ} \cup \{U_i =^? T \times W_j,\ W_j =^? W_{j1} + W_{j2},\ U_{i1} =^? T \times W_{j1},\ U_{i2} =^? T \times W_{j2}\}}$$

where $U, W \in \{x, y\}$.

Specifically, we examine the complexity of unifying a set of equations from EQ. It will be shown that to achieve a unifier, the Tiden-Arnborg algorithm requires exponentially many steps.

Definition 2.2. For $n \geq 0$, let $\sigma(n)$ be the set of equations

$$\begin{aligned} x_{1^i} &=^? x_{1^{i+1}} + x_{1^i2}, \\ y_{2^i} &=^? y_{2^i1} + y_{2^{i+1}}, \\ y_{2^i1} &=^? T \times x_{1^i2}, \\ x &=^? T \times y, \\ x_{1^{i+1}} &=^? x_{1^{i+2}} + x_{1^{i+1}2} \end{aligned}$$

for all $0 \leq i \leq n$.

Thus $\sigma(0)$ is $\{x =^? x_1 + x_2, y =^? y_1 + y_2, x_1 =^? x_{11} + x_{12}, x =^? T \times y, y_1 =^? T \times x_2\}$.

Similarly $\sigma(2)$ is $\{x =^? x_1 + x_2, y =^? y_1 + y_2, x_1 =^? x_{11} + x_{12}, y_2 =^? y_{21} + y_{22}, x_{11} =^? x_{111} + x_{112}, y_{22} =^? y_{221} + y_{222}, x_{111} =^? x_{1111} + x_{1112}, x =^? T \times y, y_1 =^? T \times x_2, y_{21} =^? T \times x_{12}, y_{221} =^? T \times x_{112}\}$.

Note that $\sigma(k+1) = \sigma(k) \cup \{y_{2^{k+1}} =^? y_{2^{k+1}1} + y_{2^{k+2}}, y_{2^{k+1}1} =^? T \times x_{1^{k+1}2}, x_{1^{k+2}} =^? x_{1^{k+3}} + x_{1^{k+2}2}\}$ for all $k \geq 0$.

Definition 2.3. We denote a variable x_i (or y_i) as a *peak* iff there are equations $x_i =^? x_{i1} + x_{i2}$ and $x_i =^? T \times y_j$ (or $y_i =^? y_{i1} + y_{i2}$ and $y_i =^? T \times x_j$)

We claim that a system of equations, as defined in Definition 2.2, will result in exponentially many applications of the sum transformation rule.

3 Proof

For a set of equations S, let $m(S)$ denote the number of $\times$ symbols in it and $p(S)$ denote the number of + symbols in it. Consider the sets of equations defined in Definition 2.2. By the analysis in [8] the number of sum transformations should be bounded by $m(S) * p(S)$. We can see that according to Definition 2.2 $m(\sigma(n)) = n + 2$ and $p(\sigma(n)) = 2n + 3$. Thus the upper bound should be $2n^2 + 7n + 6$. However, the actual bound for systems of equations $\sigma(n)$ will be shown to be $2^{n+3} - (n+4)$.

We can view the sets of equations defined in Definition 2.2 as tree-like graphs. Nodes correspond to variables. We first add a dummy root node with outdegree 2 whose children are the initial nodes x and y. The summation equations are represented by downward edges, from every parent node to its two children. We represent the multiplication equations as lateral edges, i.e., edges between nodes at the same *level,* i.e., distance from the root node. (Thus the graph is not really a tree if lateral edges are considered.). Because all left multiplication edges goto T and have no effect on the complexity of the algorithm in these systems of equations, we leave these edges out of the diagrams for clarity. Let $G(n)$ be the graph of $\sigma(n)$ See Figure 1 for $G(0)$. Note that the height of the tree is 3, i.e, there are 3 levels. In general, the graph of $\sigma(n)$ has $n+3$ levels. We view the algorithm as proceeding down the tree, with sum transformations at a level completed before starting at the next level. We analyze the complexity of the Tiden-Arnborg algorithm in terms of transformations done on the graph as the algorithm proceeds. We show that if l is the height of the tree, then the number of sum transformations applied is $2^l - (l+1)$.

Observe that a variable is a peak if and only if its node has both downward and lateral edges. Figure 2 shows the effect of a sum transformation at a peak on the graph. Note that lateral edges are never deleted. Each application of the sum transformation increases the number of lateral edges by at most 2.

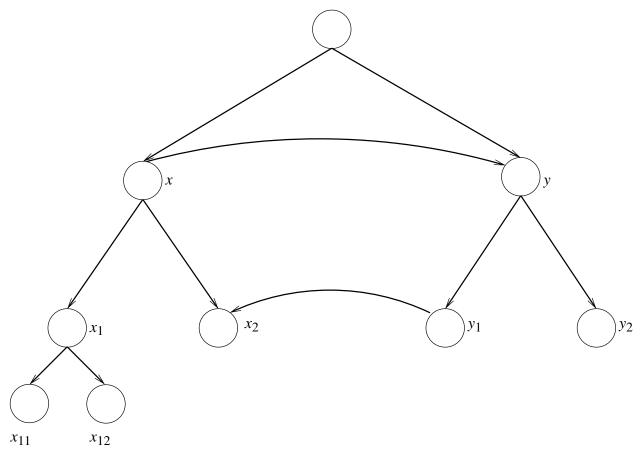

Figure 1: Example graph with a peak at node x

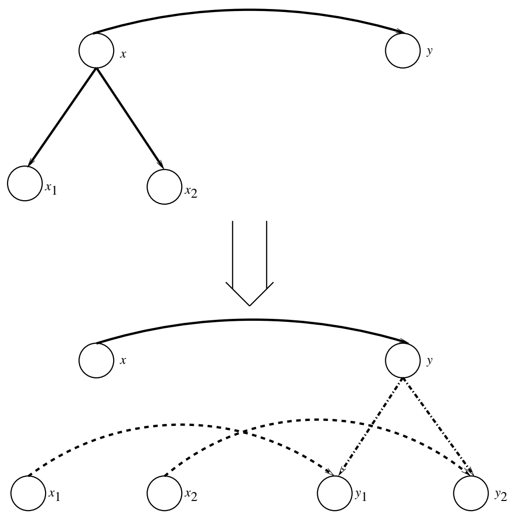

Figure 2: Sum transformation

Note also that other than at the lowest level (depth $n+3$) the graph will have initially one multiplication or exactly one edge between nodes at the same height in the graph. We can also see that the graph is partitioned between the left and right side or x and y side and that at level one, there is an edge from x to y. However, at all other lower levels, the initial edge between nodes of the same level goes from y to x.

We can also see that given the graph as described above, each time the sum transformation is applied, the peak moves from either the x side of the graph to the y side or from the y side to the x side, and the new peak was not previously a peak. To see this, take any system as defined by Definition 2.2 and examine the graph of that system. Initially all edges from nodes at the same height only go from one side to the other. In this limited formulation of Definition 2.2 these same level edges are the only multiplication functions. This ensures that any time a sum transformation is performed on some equation, $x_i =^? T \times y_j$ or $y_i =^? T \times x_j$, by definition the new edges created by the sum transformation must go from either x to y or y to x because there are no multiplication equations of the form $x_i =^? T \times x_j$. The fact that a new peak was not previosly a peak follows from Definition 2.2 and the definition of the sum transformation. Since we assume a simple system of equations, there are no two distinct equations of the form $x =^? x_i + x_j$, $x =^? x_k + x_l$ (i.e., with the same variable on the left-hand side): likewise for y. Once when the sum transformation is applied to equations $x_i =^? T \times y_j$ and $x_i =^? x_{i1} + x_{i2}$, the downward edges from x_i to both x_{i1} and x_{i2} are removed (see Figure 3). Thus x_i can never become a peak again.

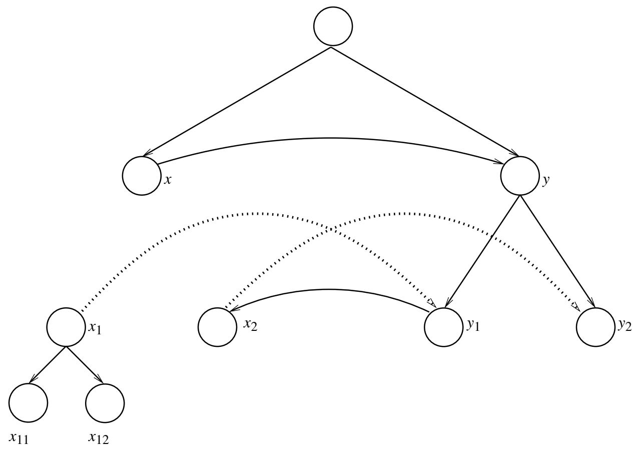

Figure 3: Graph after one application of the sum transformation at node x, creating a new peak at node x_1 and new edges from x_1 to y_1 and x_2 to y_2

Lemma 3.1. *Sum transformations at level k create a peak at x_{1^k} at level $k+1$ provided $k+1$ is not the lowest level.*

Proof. This follows inductively from the form of the graph in Definition 2.2. First from the definition of the set of equations, the first peak is located at x. After the first sum transformation a peak is created at x_1. Assume this propagates to level n. Then there is a peak at x_{1^n} to which the sum transformation is applied, adding an equation of the form $x_{1^{n+1}} =^? T \times y_{1^{n+1}}$, which creates a peak at the next level, provided there is already an equation $x_{1^{i+1}} =^? x_{1^{i+2}} + x_{1^{i+1}2}$. $\square$

Lemma 3.2. *There are no lateral edges from nodes corresponding to variables of the form y_{2^j} for $j \geq 0$. In other words, no equations of the form $y_{2^j} =^? T \times x_k$ are generated.*

Proof. This follows inductively from Definition 2.2. In any initial system there is no edge from any y_{2^k} node at level k. By the definition of sum transformation an outgoing lateral edge from y_{2^k} can be created only if the parent node, $y_{2^{k-1}}$ has an outgoing lateral edge. $\square$

We can also notice a fact about the order in which the nodes become peaks via the sum transformation. The order is a right-to-left lexicographical order of the digits of the nodes' indices (i.e., subscripts). For example, for level 4 the sequence is $x_{111} \rightarrow y_{111} \rightarrow x_{211} \rightarrow y_{211} \rightarrow x_{121} \rightarrow y_{121} \rightarrow x_{221} \rightarrow y_{221} \rightarrow x_{112} \rightarrow y_{112} \rightarrow x_{212} \rightarrow y_{212} \rightarrow x_{122} \rightarrow y_{122} \rightarrow x_{222} \rightarrow y_{222}$. Note, y_{222} is not necessarily a peak but is added to illustrate the path. Based on this observation we have the following lemma:

Lemma 3.3. *At any level of the tree, if $x_i =^? T \times y_j$ is an equation (i.e., if there is a lateral edge from x_i to y_j) then $i = j$. Similarly, if $y_i =^? T \times x_j$ is an equation, then $j = revlex(i)$ where* revlex *is the lexicographic successor of the index of the node, y_i, but starting with the 1^{st} bit (i.e., from right to left).*

Proof. This follows inductively from Definition 2.2 and the sum transformation. The base cases are $x =^? T \times y$ (level 1) and $y_1 =^? T \times x_2$ (level 2). Assume this property for level k. Now we will show that all the equations introduced at level $k+1$ by sum transformation at level k will satisfy the property. If $y_i =^? T \times x_{revlex(i)}$ is an equation at level k and $y_i =^? y_{i1} + y_{i2}$ is an equation (i.e., y_i is a peak at level k), then applying the sum transformation results in $y_{i1} =^? T \times x_{revlex(i)1}$ and $y_{i2} =^? T \times x_{revlex(i)2}$. Now note that $revlex(i1) = revlex(i)1$ and $revlex(i2) = revlex(i)2$ since i is not a string of 2's. Note also that if k is not the lowest level, then there will already be an equation $y_{2^k1} =^? T \times x_{1^k2}$ at level $k+1$ but this does not violate the property in the lemma since $revlex(2^k1) = 1^k2$.

If $x_i =^? T \times y_i$ and $x_i =^? x_{i1} + x_{i2}$, then by application of the sum transformation $x_{i1} =^? T \times y_{i1}$ and $x_{i2} =^? T \times y_{i2}$ and the result follows. □

Lemma 3.4. *If there is a path of lateral edges from node u_i to node v_i in the graph at some point where node u_i is a peak, then every node on the path, except possibly v_i, will become a peak at some point.*

Proof. Straightforward, by induction on the length of the path. □

Lemma 3.5. *At every level $k < n+3$ a path of lateral edges between $x_{1^{k-1}}$ to $y_{2^{k-1}}$ is created.*

Proof. For brevity, we refer to such paths as RL paths. Clearly there (already) is an RL path at level 1. We show that if a RL path exists at level k and $k+1 < n+3$, then a RL path will be created at level $k+1$. By Lemma 3.1 there will be a peak at $x_{1^{k-1}}$ and by Lemma 3.4 every node other than $y_{2^{k-1}}$ will become a peak. This creates, at level $k+1$, edges of the form $x_{i1} =^? T \times y_{i1}$, $x_{i2} =^? T \times y_{i2}$, $y_{i1} =^? T \times x_{revlex(i)1}$ and $y_{i2} =^? T \times x_{revlex(i)2}$ for every $i \neq 2^{k-1}$. Since the edge $y_{2^{k-1}1} =^? T \times x_{1^{k-1}2}$ is already there to begin with, we get the RL path at level $k+1$. □

Lemma 3.6. *At each level $k < n+3$ of the graph, the sum transformation can be applied $2^k - 1$ times.*

Proof. Follows from Lemma 3.5. At each level k, 2^k nodes will be created eventually. An RL path can be created and thus the sum transformation must be applied to all nodes except $y_{2^{k-1}}$ resulting in $2^k - 1$ applications at each level k. □

Theorem 3.7. *For a graph of height n, $2^{n+1} - n - 2$ sum transformations are used.*

Proof. This easily follows from Lemmas 3.1– 3.6 and the fact that $\sum_{i=0}^{n} \left(2^i - 1\right) = 2^{n+1} - n - 2$. □

We see that the current algorithm fails to achive polynomial complexity for at least a subset of possible unification problems. Further counter examples may be found that also cause this exponential growth with the sum transformation. This naturally results in the question of whether a polynomial time algorithm can be found, either by a modification of the current algorithm or by a new approach.

4 An Illustrated Example

In this section we give an example of the process on a system of equations defined as in Definition 2.2. We begin with $\sigma(0)$, i.e., the following set of initial equations:

$$\begin{aligned} x &=^? T \times y, \\ x &=^? x_1 + x_2, \\ y &=^? y_1 + y_2, \\ y_1 &=^? T \times x_2, \\ x_1 &=^? x_{11} + x_{12} \end{aligned}$$

This can be represented by a graph as shown in Figure 4. Note that the first peak is located at node x. The

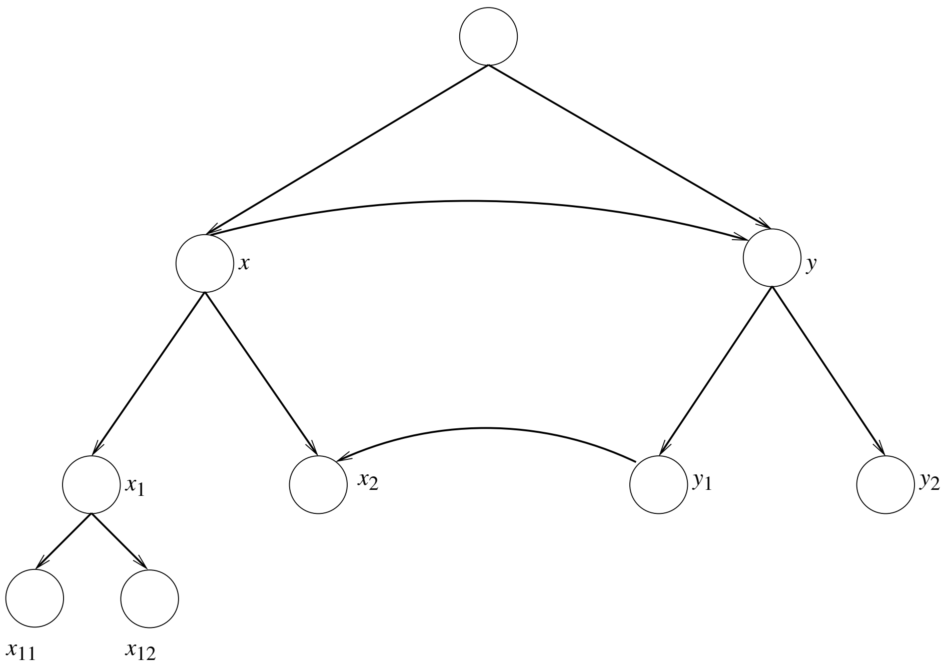

Figure 4: Graph for $\sigma(0)$

first peak, x, is selected and the sum transformation can be applied, resulting in the removal of equation $x =^? x_1 + x_2$ from the set of equations and the addition of the two equations $x_1 =^? T \times y_1$ and $x_2 =^? T \times y_2$. The direction of the new edges are from x to y due to the fact that the multiplication equation from the peak, to which the sum transformation was applied was also from x to y. After the sum transformation is applied x is no longer a peak because of the removal of $x =^? x_1 + x_2$, but now the node x_1 is a peak due to the addition of $x_1 =^? T \times y_1$ (see Lemma 3.6). This new graph is shown in Figure 5.

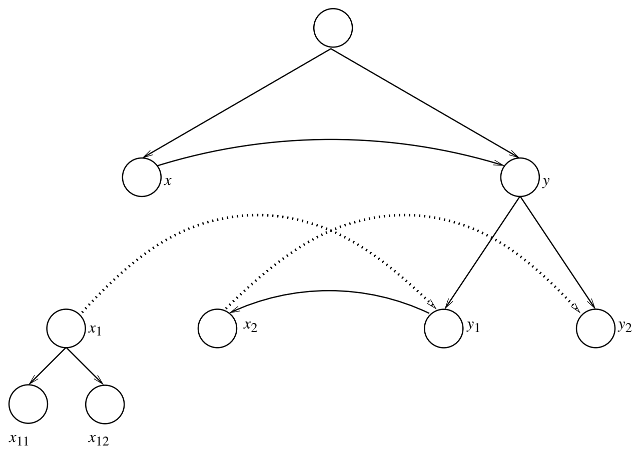

Figure 5: After one application of the sum transformation

We see that only one application of the sum transformation can be applied at level 1. At the next level we continue the process begining with the new peak x_1. The result of applying the sum transformation on x_1 is the removal of $x_1 =^? x_{11} + x_{12}$ from the set of equations and the addition of two new edges, $x_{11} =^? T \times y_{11}$ and $x_{12} =^? T \times y_{12}$, to the set of equations. The two new y nodes are also created adding $y_1 =^? y_{11} + y_{12}$ to the set of equations. The result is that x_1 is no longer a peak but now y_1 is (see Lemma 3.3). The resulting graph can be seen in Figure 6. Also, now that y_1 is the peak the direction of the multiplication path has switched to the direction of y_1 to x_2 (see Lemma 3.1).

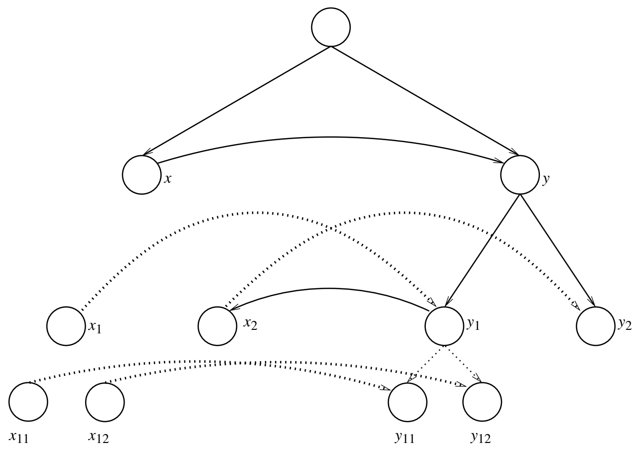

Figure 6: After two applications of the sum transformation

We can now continue the process, applying the sum transformation to the peak at y_1. This will remove $y_1 =^? y_{11} + y_{12}$ fom the set of equations and add $x_2 =^? x_{21} + x_{22}$ to the set of equations, creating a peak at node x_2 and removing the peak at node y_1. Lastly, a third sum transformation is applied to node x_2, removing $x_2 =^? x_{21} + x_{22}$ from the set of equations and adding $y_2 =^? y_{21} + y_{22}$ to the set of equations. Note that because there is no multiplication path from y_2 to some x node y_2 is not a peak and no more sum transformations can be applied at the current level. Because there are also no additional nodes at the next level we stop with a total of 4 applications of the sum transformation. The final graph is shown in Figure 7.

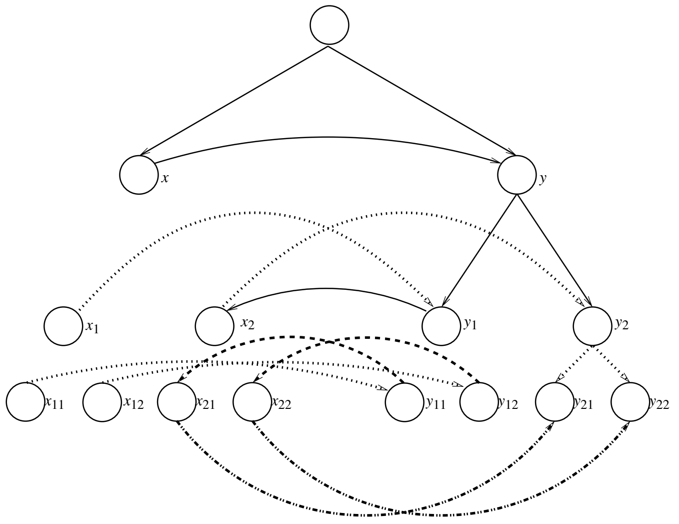

Figure 7: After 4 applications of the sum transformation

5 Conclusions

We have shown that the Tiden-Arnborg algorithm does not run in polynomial time as claimed in [8]. It is also not hard to see that the algorithm produces exponentially large *mgus* for the set of systems $\sigma(n)$. However, it may still be that the *unifiability* problem, i.e., whether a unifier exists modulo this theory, is in P. We are currently working on this and related problems.

References

[1] Siva Anantharaman, Hai Lin, Christopher Lynch, Paliath Narendran & Michaël Rusinowitch (2010): *Cap unification: application to protocol security modulo homomorphic encryption*. In: Dengguo Feng, David A. Basin & Peng Liu, editors: *ASIACCS*, ACM, pp. 192–203. Available at `http://doi.acm.org/10.1145/1755688.1755713`.

[2] Franz Baader & Wayne Snyder (2001): *Unification Theory*. In: John Alan Robinson & Andrei Voronkov, editors: *Handbook of Automated Reasoning*, Elsevier and MIT Press, pp. 445–532.

[3] Evelyne Contejean (1993): *A Partial Solution for D-Unification Based on a Reduction to AC1-Unification*. In: Andrzej Lingas, Rolf G. Karlsson & Svante Carlsson, editors: *ICALP*, *Lecture Notes in Computer Science* 700, Springer, pp. 621–632. Available at `http://dx.doi.org/10.1007/3-540-56939-1_107`.

[4] Evelyne Contejean (1993): *Solving *-Problems Modulo Distributivity by a Reduction to AC1-Unification*. *J. Symb. Comput.* 16(5), pp. 493–521.

[5] Jean-Pierre Jouannaud & Claude Kirchner (1991): *Solving Equations in Abstract Algebras: A Rule-Based Survey of Unification*. In: *Computational Logic - Essays in Honor of Alan Robinson*, pp. 257–321.

[6] Hai Lin (2009): *Algorithms for Cryptographic Protocol Verification in Presence of Algebraic Properties*. Ph.D. thesis, Clarkson University.

[7] Manfred Schmidt-Schauß (1998): *A Decision Algorithm for Distributive Unification*. *Theor. Comput. Sci.* 208(1-2), pp. 111–148. Available at `http://dx.doi.org/10.1016/S0304-3975(98)00081-4`.

[8] Erik Tidén & Stefan Arnborg (1987): *Unification Problems with One-Sided Distributivity*. *J. Symb. Comput.* 3(1/2), pp. 183–202.