Journal of Automata, Languages and Combinatorics **u** (v) w, x–y

Unification over Distributive Exponentiation (Sub)Theories

Serdar Erbatur[1]
University at Albany–SUNY (USA)
e-mail: se@cs.albany.edu

Andrew M Marshall[2]
University at Albany–SUNY (USA)
e-mail: marshall@cs.albany.edu

Deepak Kapur[3]
University of New Mexico (USA)
e-mail: kapur@cs.unm.edu

and

Paliath Narendran[4]
University at Albany–SUNY (USA)
e-mail: dran@cs.albany.edu

ABSTRACT

Arithmetic operators are extensively used in cryptographic protocols. While a protocol using such operations may appear safe if semantic properties of these operations are not used by an intruder, the protocol can become vulnerable otherwise. Several such examples have been reported in the literature. The focus in this paper is on the modular exponentiation operator and its interaction with modular multiplication operators. Unification algorithms for theories involving exponentiation and multiplication operations play an important role in state exploration based approaches for finding attacks. This paper gives decidability results for unification problems for subtheories of exponentiation. The first property considered is the simplification of exponentiation when the exponent is an expression involving modular multiplication $\circledast$. The second property investigated is the simplification of exponentiation in which the base expression is expressed using yet another modular multiplication $*$. Extensions of these theories in which modular multiplication $\circledast$ is associative and/or commutative are investigated. The approach used for developing unification algorithms is novel and hierarchical, in the sense a unification algorithm for properties of the multiplication operator can be employed as a plug-in into the inference rules for unification derived from equational properties of exponentiation with multiplication operations. A table summarizing all known results about theories of exponentiation is included as well.

[1]Partially supported by the NSF grants CNS-0831209 and CNS-0905286
[2]Partially supported by the NSF grants CNS-0831209 and CNS-0905286
[3]Partially supported by the NSF grants CNS-0831462 and CNS-0905222
[4]Partially supported by the NSF grants CNS-0831209 and CNS-0905286

Keywords: Unification, Term Rewriting, Protocol Security

1. Introduction

The exponentiation operator on numbers is used extensively in cryptographic algorithms and protocols, some of the examples being the RSA and El Gamal encryption and signature schemes, Agnew-Mullin-Vanstone protocol for interactive data exchange, Yacobi-Shmuely protocol for public key distribution, JVSK protocols for multi-party key exchange, and Lim-Lee Schnorr based protocols for authentication and key exchange (see [7] for a review of these protocols). Many protocols using arithmetic operations may appear safe, devoid of any potential attacks, if the intruder chooses not to use the semantic properties of the arithmetic operators, an assumption often erroneously made in many protocol analysis tools. Instead we have been investigating an approach for analyzing protocols where semantic properties of arithmetic operators such as exponentiation are also considered for possible exploitation in breaking such protocols. A protocol is modeled as a state machine and an execution of a protocol is a sequence of state transitions. Search space is explored using unification and narrowing techniques to handle semantic properties of the operators[5].

This paper is a continuation of our earlier work [19, 17, 16, 18, 13], on developing unification algorithms for restricted versions of the theory of exponentiation, with the goal of avoiding undecidability and developing efficient unification algorithms when possible. In [17, 16, 18], we considered almost every possible property of the exponentiation operator and multiplication operators. Many decidable and undecidable subtheories were identified. (A table is given at the end.) Of late we have tried a different approach, assuming the fewest required properties of exponentiation and multiplication operators depending on the applications. In [13], for example, we did assume that exponentiation distributes over multiplication in the base (equality (2) below) but we only considered exponentiation with a *fixed* base. Since base and exponent multiplications usually are over different moduli, two multiplication operators are considered. A unification decision algorithm was given for this partial theory of exponentiation in [13]. It is also shown there that if both multiplication operators are assumed to have the properties constituting abelian groups, then the unification problem becomes undecidable.

In this paper, we consider the distributivity property of exponentiation over base multiplication ($*$), but we do not assume any other property of this multiplication. Instead of using a fixed exponentiation base, we consider a general base, along with an axiom that expresses how repeated exponentiation is simplified with $\circledast$ in the exponent:

$$exp(exp(X, Y), Z) = exp(X, Y \circledast Z)$$

This second multiplication ($\circledast$) is then considered to have any subset of associative and/or commutative properties. In each case, the unification problem is shown to be decidable when only distributivity property of exponentiation is assumed, as well as when both properties of exponentiation are considered.

[5] See [20] and [8] for other approaches that deal with modular exponentiation.

The discussion of the unification algorithms is patterned after the discussion in [13], where inference rules for exponentiation terms are derived from equational properties of exponentiation.

The approach used to develop unification algorithms in this paper appears to be more general than is given here. It is based on dividing function symbols appearing in an equational theory into two disjoint subsets: (i) function symbols which do not appear as outermost symbols, henceforth called *constructors*, in equations and (ii) function symbols appearing as outermost symbols, henceforth called *observers*; we do however allow permutative equations among constructors as part of an equational theory. In fact, further extensions may be possible in which observers are organized hierarchically, i.e., observers can serve as constructors for other observers.

Thus the axioms that we consider are

$$exp(exp(X,Y),Z) = exp(X, Y \circledast Z) \tag{1}$$

$$exp(X * Y, Z) = exp(X,Z) * exp(Y,Z) \tag{2}$$

$$(X \circledast Y) \circledast Z = X \circledast (Y \circledast Z) \tag{3}$$

$$X \circledast Y = Y \circledast X \tag{4}$$

Combining a selection of these axioms allows us to build several theories. That is, we can start with the basic axioms, (1) and (2), of exponentiation and add axiom (3) and or axiom (4). The addition of axiom (3) results in an associative $\circledast$ operator. The addition of axiom (4) results in a commutative $\circledast$ operator. The addition of both axioms (3) and (4) will thus result in an associative and commutative (AC) operator $\circledast$. We use the following notation to denote each of these theories. Let $\mathcal{E}$ denote the theory corresponding to just the axioms (1) and (2). Therefore in this theory, $\mathcal{E}$, there is no associative or commutative operator. Let $\mathcal{E}_{AC}$ denote the theory of $\mathcal{E}$ with the addition of axioms (3) and (4). Therefore, $\mathcal{E}_{AC}$ contains the AC operator $\circledast$. Let $\mathcal{E}_A$ denote $\mathcal{E}$ with the addition of axiom (3). Thus, $\mathcal{E}_A$ has an associative (A) operator $\circledast$. Let $\mathcal{E}_C$ denote $\mathcal{E}$ with the addition of axiom (4). Thus, $\mathcal{E}_C$ has a commutative (C) operator $\circledast$. Likewise let $\mathcal{F}$ denote the theory corresponding to just axiom (1). Therefore, $\mathcal{F}$ has no AC operator and in addition does not contain the distributivity axiom, axiom (2). As for $\mathcal{E}$ we can form the larger theories $\mathcal{F}_A$, $\mathcal{F}_C$ and $\mathcal{F}_{AC}$. For the convenience of the reader we include Table 1 that lists the name of each theory, the axioms it contains, which if any of the operators are A, C, or AC, and the section the theory is discussed.

The paper is organized as follows. In the next section basic definitions are introduced. In Section 3, we show that once the first axiom

$$exp(exp(X,Y),Z) = exp(X, Y \circledast Z)$$

is included we 'effectively' get associativity of $\circledast$ (Axiom (3)); this is shown by just considering $\mathcal{F}$, and proving that for elementary terms s and t over $\circledast$, if $exp(x,s) =_{\mathcal{F}} exp(x,t)$, then $s =_A t$ even though the associativity property of $\circledast$ is not included in the equational theory. In Section 4, we show the decidability of unifiability check for $\mathcal{E}_{AC}$ and $\mathcal{E}_A$, as well as give algorithms. For the associative and *non*-commutative theories discussed in Sections 3, 4 and 5, if two terms are A-unifiable, there are in

Name	Axioms	A, C, AC	Section
$\mathcal{E}_{AC}$	(1) $exp(exp(X,Y),Z) = exp(X, Y \circledast Z)$ (2) $exp(X * Y, Z) = exp(X,Z) * exp(Y,Z)$ (3) $(X \circledast Y) \circledast Z = X \circledast (Y \circledast Z)$ (4) $X \circledast Y = Y \circledast X$	AC-$\circledast$	4
$\mathcal{E}_{A}$	(1) $exp(exp(X,Y),Z) = exp(X, Y \circledast Z)$ (2) $exp(X * Y, Z) = exp(X,Z) * exp(Y,Z)$ (3) $(X \circledast Y) \circledast Z = X \circledast (Y \circledast Z)$	A-$\circledast$	4
$\mathcal{E}_{C}$	(1) $exp(exp(X,Y),Z) = exp(X, Y \circledast Z)$ (2) $exp(X * Y, Z) = exp(X,Z) * exp(Y,Z)$ (4) $X \circledast Y = Y \circledast X$	C-$\circledast$	4
$\mathcal{F}_{AC}$	(1) $exp(exp(X,Y),Z) = exp(X, Y \circledast Z)$ (3) $(X \circledast Y) \circledast Z = X \circledast (Y \circledast Z)$ (4) $X \circledast Y = Y \circledast X$	AC-$\circledast$	5
$\mathcal{F}_{A}$	(1) $exp(exp(X,Y),Z) = exp(X, Y \circledast Z)$ (3) $(X \circledast Y) \circledast Z = X \circledast (Y \circledast Z)$	A-$\circledast$	5
$\mathcal{F}_{C}$	(1) $exp(exp(X,Y),Z) = exp(X, Y \circledast Z)$ (4) $X \circledast Y = Y \circledast X$	C-$\circledast$	5

Table 1: Theories considered and their axioms.

general infinitely many most general unifiers. When commutativity is included too, as in Section 4 for $\mathcal{E}_{AC}$, the number of most general unifiers becomes finite, since AC-unification is finitary [15]. In this case our method can also be used to compute a finite complete set of unifiers. In Section 5, we briefly consider unification modulo $\mathcal{F}_{AC}$. The final section concludes with a table summarizing results about various theories involving exponentiation operator.

1.1. Related Work

There is a vast amount of literature on unification algorithms for various equational theories; an interested reader can consult survey papers [12, 5].

The focus of this paper is instead on unification problems used in protocol analysis. Over the last decade, this application of unification has been investigated extensively, along with the use of narrowing. Of particular interest are distributivity properties and the homomorphism property, such as distributivity of exponentiation over multiplication, as well as encryption over concatenation [1]. In addition, new concepts such as cap unification [2] have been introduced to focus on specific aspects of unification algorithms as used in cryptographic protocol analysis. In our work, we have focussed on the interaction between exponentiation and different multiplication operators, and developed decidability and undecidability results. We do not follow the narrowing approach [11], instead focussing on specialized algorithms.

As will be evident to the reader, developing unification algorithms for combina-

tion of theories is also critical in this application. However, the assumption about disjoint symbols [24, 4] (which is still useful to deal with uninterpreted symbols and constants), must be relaxed. Research on combination methods for unification for nondisjoint theories is somewhat limited [10, 6]. In this paper, we advocate a hierarchical combination approach as discussed in Section 4, where a unification algorithm for various subtheories of associative/commutativity of multiplication can be used as a plug-in. This is in contrast to our earlier works where for every subtheory, a specialized algorithm is developed.

2. Preliminaries and General Results

We use the standard notation of equational unification [5] and term rewriting systems [3]. Given an equational theory E, an E-unification problem is a set of equations

$$\mathcal{S} = \{s_1 =^? t_1, \ldots, s_m =^? t_m\}$$

A solution to $\mathcal{S}$, called an *E-unifier*, is a substitution σ such that $\sigma(s_i) =_E \sigma(t_i)$ for all $1 \leq i \leq m$. A substitution σ is *more general modulo* E than θ on a set of variables V, denoted as $\sigma \leq_E^V \theta$, if and only if there is a substitution τ such that $\sigma\tau(x) =_E \theta(x)$ for all $x \in V$. Two substitutions θ_1 and θ_2 are *equivalent modulo* E on a set of variables V, denoted as $\theta_1 \equiv_E^V \theta_2$, if and only if $\theta_1(x) =_E \theta_2(x)$ for all $x \in V$. A substitution θ is said to be *discriminating* over a set of variables V (or V*-discriminating*) if and only if $\sigma(x) \neq_E \sigma(y)$ for all distinct x, y in V. For a substitution θ and a set of variables V, $\theta|_V$ denotes the restriction of the substitution to the variables in V, i.e.,

$$\theta|_V \;=\; \{\, x \mapsto \theta(x) \mid x \in V \,\}$$

We call a set Σ of substitutions a *complete set of E-unifiers* of $\mathcal{S}$ if and only if (i) for every $\theta \in \Sigma$, θ is an E-unifier and (ii) for every E-unifier θ, there is a substitution $\sigma \in \Sigma$ where $\sigma \leq_E^{Var(S)} \theta$ holds. A complete set of E-unifiers Σ of a unification problem $\mathcal{S}$ is *minimal* if and only if for any two E-unifiers σ and θ in Σ, $\sigma \leq_E^{Var(S)} \theta$ implies that $\sigma = \theta$.

Recall that there are four types in the unification hierarchy, based on the cardinality of minimal complete sets of unifiers [5]. An E-unification problem $\mathcal{S}$ is of type *unitary*, if the minimal complete set of E-unifiers of $\mathcal{S}$ has size one. $\mathcal{S}$ is *finitary (infinitary)* if the minimal complete set of E-unifiers of it is finite (infinite). We note that the minimal set of unifiers need not exist. We say $\mathcal{S}$ is of type *zero* in that case. A unification problem has one of types unitary, finitary, infinitary or nullary. Types are denoted as $1, \omega, \infty, 0$ respectively and sorted as $1 < \omega < \infty < 0$. Unification type of an equational theory E is the maximal unification type of a problem modulo E. Unification modulo associativity is known to be infinitary [22]. *AC*-unification, on the other hand, is finitary [15].

Equational unification problems are also classified based on the function symbols that appear in them, i.e., their signature (*Sig*). An E-unification problem S is *elementary* if and only if $Sig(S) = Sig(E)$. S is called an E-unification problem *with constants* if $Sig(S) \setminus Sig(E)$ contains only free constants. Finally, if there are un-

interpreted function symbols in $Sig(S) \setminus Sig(E)$, S is called a general E-unification problem.

A set of equations S is said to be in *standard form* over a signature F if and only if every equation in S is of the form $X =^? t$ where X is a variable and t, a term over F, is one of the following: (a) a variable different from X, (b) a constant, or (c) a term of depth 1 that contains no constants. We say S is *in standard form* if and only if it is in standard form over the entire signature. For a set of equations S in standard form, $lhs(S)$ denotes the set of left-hand sides of equations in S. It is not generally difficult to decompose equations of a given problem into simpler standard forms. For a set of equations S in standard form and a function f, let $S|_f$ denote the equations in S whose right-hand sides have f as a root symbol.

A set of equations is said to be in *dag-solved form* (or *d-solved form*) if and only if they can be arranged as a list

$$X_1 =^? t_1, \ \ldots, \ X_n =^? t_n$$

where (a) each left-hand side X_i is a distinct variable, and (b) $\forall\, 1 \leq i \leq j \leq n$: X_i does not occur in t_j ([12]). A set of equations S is said to be in *F-solved form* if and only if it is in standard form and the subset of equations $S \cap (V \times T(F,V))$ is in dag-solved form.

2.1. Subterm Collapse-Freeness

A theory E is *subterm-collapse-free* if and only if for all terms t it is not the case that $t =_E t'$ where t' is a proper subterm of t. A rewrite rule $l \to r$ is called *non-size-reducing* if and only if $|\theta(l)| \leq |\theta(r)|$ for all substitutions θ. It can be shown that $l \to r$ is non-size-reducing if $|r| \geq |l|$, where $|t|$ denotes the number of function symbols in the term t, and the multiset of variables that occur in r is a superset of the multiset of variables in l. In this section we show that the theories $\mathcal{E}$, $\mathcal{F}$ and their combinations with associativity and/or commutativity axioms satisfy the property of *subterm-collapse-freeness*. This property will be helpful for error checking in the algorithms. Note that it suffices to prove the property for $\mathcal{E}_{AC}$. This is due to the fact that $\mathcal{E}_{AC}$ is the most general theory, i.e., the other theories are subtheories of it.

Theorem 2.1 *$\mathcal{E}_{AC}$ is subterm-collapse-free.*

Proof. This theory has the following AC-convergent system $\mathcal{R}_e$:

1. $exp(exp(X,Y),Z) \to exp(X, Y \circledast Z)$
2. $exp(X * Y, Z) \to exp(X,Z) * exp(Y,Z)$

By the definition of *non-size-reducing* rules, we can see that both rewrite rules 1 and 2 are non-size-reducing. As this is also the case with the associativity and commutativity axioms, we cannot reduce a term t to a subterm of itself. □

Directly from Lemma 2.1 we get the following.

Lemma 2.2 *The theories $\mathcal{E}_{AC}$, $\mathcal{E}_A$, $\mathcal{E}_C$, $\mathcal{E}$, $\mathcal{F}_{AC}$, $\mathcal{F}_A$, $\mathcal{F}_C$ and $\mathcal{F}$ are subterm-collapse-free.*

3. Unification modulo $\mathcal{F}$

Here we consider the theory

$$exp(X, Y \circledast Z) \quad = \quad exp(exp(X, Y), Z)$$

and show that, although $\mathcal{F}$ does not contain the A axiom (3) for $\circledast$, unification modulo this theory is at least as hard as A-unification.

It is useful to note that this theory has a finite convergent system

$$exp(X, Y \circledast Z) \quad \rightarrow \quad exp(exp(X, Y), Z)$$

as well as an infinite one

$$\begin{aligned}
exp(exp(X, Y), Z) &\rightarrow exp(X, Y \circledast Z) \\
exp(X, (Y_1 \circledast Y_2) \circledast Z) &\rightarrow exp(X, (Y_1 \circledast (Y_2 \circledast Z))) \\
exp(X, Y_1 \circledast ((Y_2 \circledast Y_3) \circledast Z)) &\rightarrow exp(X, (Y_1 \circledast (Y_2 \circledast (Y_3 \circledast Z)))) \\
exp(X, Y_1 \circledast (Y_2 \circledast ((Y_3 \circledast Y_4) \circledast Z))) &\rightarrow exp(X, (Y_1 \circledast (Y_2 \circledast (Y_3 \circledast (Y_4 \circledast Z))))) \\
&\dots
\end{aligned}$$

The general pattern is

$$\begin{aligned}
exp(X, Y_1 \circledast (\dots \circledast (Y_n \circledast ((Y_{n+1} \circledast Y_{n+2}) \circledast Z)))) &\rightarrow \\
exp(X, (Y_1 \circledast (\dots \circledast (Y_n \circledast (Y_{n+1} \circledast (Y_{n+2} \circledast Z))))))
\end{aligned}$$

Lemma 3.1 *Let $\mathcal{C}$ be a set of (free) constants, $t_1, t_2 \in T(\{\circledast\} \cup \mathcal{C}, \mathcal{V})$ and $X \in \mathcal{V}$. Then*

$$exp(X, t_1) =_{\mathcal{F}} exp(X, t_2) \quad \text{iff} \quad t_1 =_A t_2$$

Proof. Without loss of generality suppose t_1 and t_2 are irreducible terms. If $t_1 =_{\mathcal{A}} t_2$, then it is easy to see that $exp(X, t_1) =_{\mathcal{F}} exp(X, t_2)$. The reason is that we can use the finite convergent rewrite system above to show that both left and right sides rewrite to same normal form. For the "only if" part, assume that $exp(X, t_1) =_{\mathcal{F}} exp(X, t_2)$. In case that $t_1, t_2 \in T(\{\circledast\} \cup \mathcal{C})$ or $t_1, t_2 \in \mathcal{V}$, it trivially follows that $t_1 =_{\mathcal{A}} t_2$. Otherwise, $exp(X, t_1)$ and $exp(X, t_2)$ will reduce to syntactically identical normal forms w.r.t $\mathcal{F}$. Notice that rewrite proof steps occur at root positions each time. By induction on the number of rewrite steps and backtracking from the normal forms to the original one we get $t_1 =_{\mathcal{A}} t_2$. □

Lemma 3.2 *Let $\mathcal{C}$ be a set of (free) constants, $t_1, t_2 \in T(\{\circledast\} \cup \mathcal{C}, \mathcal{V})$ and $a \in \mathcal{C}$. Then $exp(a, t_1) =^? exp(a, t_2)$ is $\mathcal{F}$-unifiable if and only if $t_1 =^? t_2$ is A-unifiable.*

Proof. The "if" part is straightforward. For the "only if" part, let θ be a ground $\mathcal{F}$-unifier of $exp(a, t_1) =^? exp(a, t_2)$. If $\theta(t_1)$ and $\theta(t_2)$ belong to $T(\{\circledast\} \cup \mathcal{C})$ then by the previous lemma $\theta(t_1) =_A \theta(t_2)$ and

$$\begin{array}{rcl} \theta(exp(a,t_1)) & =^?_{\mathcal{F}} & \theta(exp(a,t_2)) \\ \Rightarrow exp(\theta(a),\ \theta(t_1)) & =^?_{\mathcal{F}} & exp(\theta(a),\ \theta(t_2)) \\ \Rightarrow \theta(t_1) & =^?_{\mathcal{F}} & \theta(t_2) \end{array}$$

This implies that $t_1 =^? t_2$ is A-unifiable. Otherwise, replace every top-level subterm of $\theta(t_1)$ and $\theta(t_2)$ that has exp or $\circledast$ as a root symbol by a constant c. Alternatively, let

$$\begin{array}{ll} \hat{\theta}(x) = \theta(x) & \text{if } \theta(x) \in T(\{\circledast\} \cup \mathcal{C}) \\ = c & \text{otherwise.} \end{array}$$

$\hat{\theta}$ is also a $\mathcal{F}$-unifier for the system. Then by the previous lemma using the same idea, we get that $\hat{\theta}(t_1) =_A \hat{\theta}(t_2)$ and $t_1 =^? t_2$ is A-unifiable. □

If $\circledast$ also satisfies the commutativity property, then we can show in a similar manner that unification modulo $\mathcal{F}_C$ is at least as hard as AC-unification.

Lemma 3.3 *Let $\mathcal{C}$ be a set of (free) constants, $t_1, t_2 \in T(\{\circledast\} \cup \mathcal{C}, \mathcal{V})$ and $a \in \mathcal{C}$. Then $exp(a,t_1) =^? exp(a,t_2)$ is $\mathcal{F}_C$-unifiable if and only if $t_1 =^? t_2$ is AC-unifiable.*

4. Unification modulo the theories $\mathcal{E}_{AC}$ and $\mathcal{E}_A$

In this section we develop unification algorithm for the theory $\mathcal{E}_{AC}$ and show how the approach we develop implies an algorithm for the subtheory $\mathcal{E}_A$. As this is the most involved section of the paper let us first give an overview. Subsection 4.1 through Subsection 4.2. overview and then give a unifiability check algorithm for $\mathcal{E}_{AC}$. Subsection 4.3 through Subsection 4.4 provide the detailed proofs of the correctness of the $\mathcal{E}_{AC}$ algorithm. Subsection 4.4.1 shows how a complete set of unifiers can be obtained. The last section explains how the same methods used in the algorithm for $\mathcal{E}_{AC}$ will produce a unifiability check algorithm for $\mathcal{E}_A$.

4.1. Algorithm Overview

We consider the unification problem modulo the equational theory induced by axioms (1)–(4), denoted as $\mathcal{E}_{AC}$. This theory has the following AC-convergent system $\mathcal{R}_e$:

$$\begin{array}{rcl} exp(exp(X,Y),Z) & \rightarrow & exp(X, Y \circledast Z) \\ exp(X * Y, Z) & \rightarrow & exp(X,Z) * exp(Y,Z) \end{array}$$

Without loss of generality, we assume equations to be in the following standard forms:

$$U =^? V,\ U =^? V \circledast Y,\ U =^? X * W \ \text{ and } U =^? exp(K,L)$$

where U, V, Y, X, W, K and L are variables. Equations of the form $U =^? X * W$, $U =^? exp(K,L)$ and $U =^? V \circledast Y$ are called $*$, exp, and $\circledast$-equations respectively. In an equation of the form $U =^? exp(K,L)$, K is called a *base variable* and L is called an *exponent variable.*

The main approach is to reduce an $\mathcal{E}_{AC}$-unification problem $\mathcal{S}$ to a general AC-unification problem. The algorithm for $\mathcal{E}_{AC}$ is a *hierarchy* based algorithm. The first or "higher theory" corresponds to $\mathcal{E}$ and the "lower theory" corresponding to the AC theory for $\circledast$. For an $\mathcal{E}_{AC}$-unification problem S, the algorithm first transforms, through a set of syntactic inference rules, the subset of exp and $*$ equations in S into a solved form. The algorithm then uses a general AC-unification algorithm to solve the remainder of the equations.

We present the algorithm for unification modulo $\mathcal{E}_{AC}$ step by step. Step 2 uses a nondeterministic procedure, denoted as $\mathfrak{R}_1$, that transforms terms with exp and $*$ as root symbols into a solved form. In Step 2 we might detect a failure in the form of function clash or (extended) occur-check. Step 3 is applied to the result of a successful execution of $\mathfrak{R}_1$ and corresponds to running (general) AC unification on the results of Step 2. The application of AC-unification may change the solved form produced in Step 2. Therefore, we develop a method that allows us to guess, in advance, the impact AC-unification will have on the solved form of Step 2. This method is based on the set of initial variables that could be effected by the AC-unification algorithm. Let $\mathcal{S}$ be any problem given in standard form. Let us consider the variables in the equations of type $X_i =^? Y_i \circledast W_i$, denoted as $\mathcal{S}|_{\circledast}$, and the exponent variables Z_j in equations such as $X_j =^? exp(T_j, Z_j)$. We call this set $\mathcal{V}(\mathcal{S})$ (or simply $\mathcal{V}$ if the problem $\mathcal{S}$ is obvious from the context), i.e.,

$$\mathcal{V}(\mathcal{S}) \;=\; Var(\mathcal{S}|_{\circledast}) \;\cup\; \{Z \mid \exists\, X =^? exp(Y, Z) \in \mathcal{S}\}$$

We explain why $\mathcal{V}(\mathcal{S})$ is needed and why it is correct in Section 4.3.
Thus the steps of the algorithm are:

Step 1: Partition $\mathcal{V}(\mathcal{S})$. For a given $\mathcal{E}_{AC}$-unification problem $\mathcal{S}$, guess a partition of $\mathcal{V}(\mathcal{S})$ and set the variables in each subset equal to each other. This requires adding new equations of type $U =^? W$, where $U, W \in \mathcal{V}(\mathcal{S})$ and U and W belong to the same subset in the partition. Suppose that one enumerates all partitions of $\mathcal{V}$ as $\Pi_1, \Pi_2, \ldots, \Pi_m$. We denote the modified problems by $\mathcal{S}^{\Pi_i}$, $1 \leq i \leq m$.

Step 2: Exhaust exp and $*$ terms. Apply the procedure $\mathfrak{R}_1$ on the selected $\mathcal{S}^{\Pi_i}$. The procedure converts, if possible, the exp- and $*$-equations into a solved form. $\mathfrak{R}_1$ is a nondeterministic procedure and thus may produce one of several solved forms for a given partition. For a partition Π_i of $\mathcal{V}(\mathcal{S})$, let us denote all possible outputs of $\mathfrak{R}_1$ on $\mathcal{S}^{\Pi_i}$ as $\mathcal{S}_1^{\Pi_i}, \mathcal{S}_2^{\Pi_i}, \ldots, \mathcal{S}_n^{\Pi_i}$. It remains to run general AC-unification algorithm on $\mathcal{S}_j^{\Pi_i}$ for some $1 \leq j \leq n$.

Step 3: Run general AC-unification. Once Step 2 has been completed successfully a general AC-unification algorithm is applied to unify the remaining $\circledast$-equations. Note that a general AC-unification algorithm must be used due to the fact that exp and $*$ are uninterpreted function symbols in the AC theory.

4.2. The Procedure $\mathfrak{R}_1$

We define $\mathfrak{R}_1$ based on a set of composite inference rules composed of the following list of basic rules.

(a) $$\frac{\mathcal{EQ} \uplus \{U =^? V\}}{\{U \mapsto V\}(\mathcal{EQ}) \cup \{U =^? V\}} \qquad \text{if } U \text{ occurs in } \mathcal{EQ}$$

(b) $$\frac{\mathcal{EQ} \uplus \{U =^? exp(V,W),\ U =^? exp(V,Y)\}}{\mathcal{EQ} \cup \{U =^? exp(V,W),\ W =^? Y\}}$$

(c) $$\frac{\mathcal{EQ} \uplus \{U =^? exp(V,W),\ U =^? exp(X,W)\}}{\mathcal{EQ} \cup \{U =^? exp(V,W),\ V =^? X\}}$$

(d) $$\frac{\mathcal{EQ} \uplus \{U =^? V * W,\ U =^? X * Y\}}{\mathcal{EQ} \cup \{U =^? V * W,\ V =^? X,\ W =^? Y\}}$$

(e) $$\frac{\mathcal{EQ} \uplus \{U =^? exp(V,W),\ U =^? X * Y\}}{\begin{array}{l}\mathcal{EQ} \cup \{U =^? exp(V,W),\ V =^? V_1 * V_2,\\ \qquad\qquad X =^? exp(V_1,W),\ Y =^? exp(V_2,W)\}\end{array}}$$

(f) $$\frac{\mathcal{EQ} \uplus \{U =^? exp(V,W),\ U =^? exp(X,Y)\}}{\mathcal{EQ} \cup \{U =^? exp(X,Y),\ Y =^? Z \circledast W,\ V =^? exp(X,Z)\}}$$

(g) $$\frac{\mathcal{EQ} \uplus \{U =^? exp(V,W),\ U =^? exp(X,Y)\}}{\mathcal{EQ} \cup \{U =^? exp(X,Y),\ V =^? X,\ W =^? Y\}}$$

(h) $$\frac{\mathcal{EQ} \uplus \{U =^? exp(V,W),\ U =^? exp(X,Y)\}}{\begin{array}{l}\mathcal{EQ} \cup \{U =^? exp(Z,L),\ V =^? exp(Z,Y'),\ X =^? exp(Z,W'),\\ \qquad L =^? W \circledast Y',\ L =^? Y \circledast W'\}\end{array}}$$

(i) $$\frac{\mathcal{EQ} \uplus \{U =^? exp(V,W),\ V =^? exp(X,Y)\}}{\mathcal{EQ} \cup \{U =^? exp(X,Z),\ V =^? exp(X,Y),\ Z =^? Y \circledast W\}}$$

In the above rules the variables Z, L, W', Y', V_1, V_2 are fresh variables. Rule (a) also has a special action on fresh variables: if a fresh variable is equated to a pre-existing variable then the fresh variable is replaced.

We apply these rules, (a) through (i), with a specific priority for each rule. The result is the following set of "composite" rules that, with the addition of error checking, define the procedure $\mathfrak{R}_1$.

(c1) $$i^* a^*$$

(c2) $$[b + c + d]\, a^*$$

(c3) $$[b + c + d]^*\, a^* i^* e\, [b + c + d]^*\, a^*$$

(c4) $$(i^*\, [b + c + d]^*\, a^*)^* (f + g + h) a^*$$

Let r_1 and r_2 denote rules from the set of rules (a) - (i). Then, r_1^* indicates *exhaustive* application of the rule r_1. Therefore, the composite rule $r_1^* r_2$ means to apply r_1 until it cannot be applied any more and then try to apply r_2. Note that even if r_1 cannot be applied the rule $r_1^* r_2$ can still be used if r_2 can be applied. Thus r_1^* does not indicate that r_1 *must* be applied but rather that if r_1 can be applied we do so exhaustively. $r_1 + r_2$ indicates apply rule r_1 *or* rule r_2. For example, rule (c3) means: if possible exhaustively apply the rules (b), (c) or (d) followed by, if possible, exhaustive applications of rules (a) and (i), we then apply rule (e), then if possible exhaustively apply the rules (b), (c) or (d) again and lastly exhaustively apply rule (a). Although the composite rules are composed of several smaller rules there exists at least one composite rule such that just a single application of any one of the rules (a) through (i) is possible, that rule can be applied. For rule (a) the composite rule is (c1), for rules (b) - (c) we can use rule (c2), for rule (e) we can use (c3), for rules (f) - (h) we can use rule (c4) and finally for rule (i) we can use (c1). Therefore, the composite rules simply enforce an order of precedence on the inference rules (a) - (i). In addition, a set of equations where none of the above rules is applicable is clearly in $\{exp, *\}$-solved form.

When $\mathfrak{R}_1$ terminates without failure for a given initial set of equations S, we call the resulting set in $\{exp, *\}$-solved form an *output* of $\mathfrak{R}_1$. Soundness and completeness of $\mathfrak{R}_1$ are addressed in Sections 4.2.3 and 4.2.4.

Let us first explain the origin of each rule included in $\mathfrak{R}_1$. Rule (a) is trivial. Rules (b) and (c) are subcases of rule (g) but (as mentioned earlier) are included because *exp* is semi-cancellative. This allows us often to avoid a non-deterministic step. Rule (d) is due to the cancellativity property of $*$. Rule (e) is a direct consequence of axiom (2). Rules (f) and (h) are consequences of axiom (1). Rule (g) corresponds to the remaining possibility, cancellation, for two equated *exp* equations. Finally, rule (i) is simply axiom (1), oriented in the direction given in $\mathcal{R}_e$, in the inference rule form and is included as it allows for simplification of the termination argument. We give several examples to illustrate why some of the less obvious rules are needed.

The reason to use rule (g) is that the operator $\circledast$ does not have an identity in this

theory. Consider the unification problem

$$\{Z =^? W \circledast W,\ Z =^? Y \circledast Y,\ U =^? exp(V,W),\ U =^? exp(X,Y)\}$$

In the absence of rule (g) this will lead to failure. But the unification problem is indeed solvable, by setting $W \mapsto Y$ and (hence) $V \mapsto X$.

The reason to use rule (h) is that when considering the set $\{U =^? exp(V,W),\ U =^? exp(X,Y)\}$, V and X may share a common base variable. Consider the unification problem

$$\{Z =^? W \circledast Y,\ Z =^? a \circledast b,\ U =^? exp(V,W),\ U =^? exp(X,Y)\}$$

Where a and b are constants. In the absence of rule (h) this will lead to failure. But the unification problem is indeed solvable, by setting $W \mapsto a$, $Y \mapsto b$ and (hence) $V \mapsto exp(Z,b)$ and $X \mapsto exp(Z,a)$.

We include an example for rule (f) in Section 4.3.

In addition to the inference rules above, error checking is also done in Step 2. The first error measures correspond to the following function clash rules. If either rule is found to be applicable, the algorithm exits with failure.

$$\text{(F1)} \qquad \frac{\mathcal{EQ} \uplus \{U =^? V * W,\ U =^? V \circledast Y\}}{FAIL}$$

$$\text{(F2)} \qquad \frac{\mathcal{EQ} \uplus \{U =^? exp(V,W),\ U =^? V \circledast Y\}}{FAIL}$$

In addition to function clashes two additional errors, occur-check and infinite applications of rule (e), must be addressed. We use a graph-based method (inspired by [26]) for detecting both types of errors. That is, two graph-based representations of $\mathcal{S}$ are built and updated after each application of an inference rule. The two errors will then correspond to cycles in the respective graphs. Therefore, if at any point a cycle is detected the algorithm halts with failure. The details and correctness of this method is given in Section 4.2.1. Clearly the result of Step 2, i.e., the output of $\mathfrak{R}_1$ depends on the choices that are made between rules (f), (g) and (h).

We argue the validity of this algorithm in the following subsections.

4.2.1. Failure Conditions

Lemma 4.1 *Rules (F1) and (F2) correspond to failure conditions.*

Proof. These are symbol clash conditions due to $\mathcal{R}_e$ being subterm collapse-free, Lemma 2.2, and the rules of $\mathcal{R}_e$. □

As mentioned earlier, other possible forms of failure are occur-check and infinite applications of rule (e). Before considering these remaining failure conditions we introduce several useful relations.

- $U \succ_e W$ iff there is an equation $U =^? exp(V, W)$.
- $U \succ_b V$ iff there is an equation $U =^? exp(V, W)$.
- $U \succ_{*_r} W$ iff there is an equation $U =^? V * W$.
- $U \succ_{*_l} V$ iff there is an equation $U =^? V * W$.
- $U \succ_{\circledast_r} W$ iff there is an equation $U =^? V \circledast W$.
- $U \succ_{\circledast_l} V$ iff there is an equation $U =^? V \circledast W$.
- $U \succ_m W$ iff $U \succ_{*_r} W$ or $U \succ_{*_l} W$
- $U \succ V$ iff there is an equation $U =^? t$ such that t is a non-variable term that contains V.

For a relation P let P^+ denote the transitive closure. Let $\sim_b$ stand for the reflexive, symmetric and transitive closure of $\succ_b$. Thus $\sim_b$ defines a set of equivalence classes over a set of variables.

We also define two graphs that will be used to check for the additional error cases. These graphs correspond to modifications of the graph-based method introduced in [26].

Definition 4.2 *The dependency graph D of a set of equations S in standard form, is an edge-labeled directed multi-graph. It has the variables of S as its vertices. For an equation $U =^? V \circledast Y$, it has a $\circledast_r$-labeled edge (U, Y) and a $\circledast_l$-labeled edge (U, V). For an equation $U =^? V * Y$, it has a $*_r$-labeled edge (U, Y) and a $*_l$-labeled edge (U, V). For an equation $U =^? exp(V, Y)$, it has an e-labeled edge (U, Y) and a b-labeled edge (U, V).*

Definition 4.3 *The propagation graph P of a set of equations S in standard form, is a directed simple graph. Its vertices are the equivalence classes of the symmetric, reflexive, and transitive closure of the relation defined by b-labeled edges in the graph D for the same system. It has an edge (V, W) iff there is and edge in D from a vertex in V to a vertex in W labeled by $*_l$ or $*_r$.*

Occur-check type errors can be defined in the following way.

Lemma 4.4 *If there is a variable X in $\mathcal{S}$ such that $X \succ^+ X$ then $\mathcal{S}$ is not unifiable.*

Proof. $\mathcal{R}_e$ is subterm collapse-free and non-size reducing by Lemma 2.2. Therefore, a term t cannot be made equal to a proper subterm of it. If we have $t = s$ and $p, q \in \mathbb{N}$ with $p \in Pos(t)$ and $q \in Pos(s)$ such that $t|_p = s|_q$, to get $t = s|_q$ the term must be reduced in size by either a size reducing or collapsing rule. Therefore, terms cannot be equal to proper subterms. □

It is easy to see that we can check for these $X \succ^+ X$ errors by a simple cycle check in the corresponding dependency graph. Therefore, after each application of an inference rule in Step 2 we update the dependency graph for that system and check for errors. If a cycle is found the algorithm terminates with failure.

An additional failure case arises due to rule (e). An example of this is the following set of equations.

$$\{X =^? V * Y,\ X =^? exp(V, W)\}$$

It can be seen that this will cause infinite application of rule (e). We can capture these kind of failures in the following Lemma.

Lemma 4.5 *Let $\mathcal{S}$ be a system of equations such that there exists variables U, W in S such that $U \succ_b^+ W$ and $U \succ_m^+ W$. Then $\mathcal{S}$ is not unifiable.*

Proof. This follows directly from the proof in [26] but can also be seen due to the fact that after any finite number of applications of rule (e) the system of equations will not be in solved form. □

Lemma 4.5 addresses a problem that could cause infinite applications of rule (e). This problem was solved in [26] by using the *propagation graph*. This is due to the fact that problems as described in Lemma 4.5 will cause cycles in the propagation graph. Therefore, each time the algorithm updates the dependency graph it also updates the propagation graph and checks for cycles. Likewise, if cycles are found the algorithm terminates with failure.

4.2.2. Termination of $\mathfrak{R}_1$

We show here that rules in $\mathfrak{R}_1$ terminate by defining a measure which decreases w.r.t. a well-founded order. Before the termination proof we need several additional results. First, $\sim_b$ defines[6] a set of equivalence classes on $Var(\mathcal{S})$. Let $[X]_b = \{V \mid V \sim_b X\}$. We can define a partial order, $\succcurlyeq_m$ on the $\sim_b$-equivalence classes based on $\succ_m$. That is, $[X]_b \succcurlyeq_m [Y]_b$ if and only if there exist $K_1 \in [X]_b$ and $K_2 \in [Y]_b$ such that $K_1 \succ_m K_2$. In what follows we assume that the number of $\sim_b$-equivalence classes does not include the classes for the new exponent variables created in rules (f), (h) and (i). We are justified in this assumption by the fact that a new exponent variable, Z, created in these rules will not appear on the left-hand side of an equation (thus possible effecting termination) unless equated, by rule (a), to a variable that already appears as the left-hand side of an equation (see Lemma 4.15). In addition new base variables, Z in $exp(Z, Y)$, are all contained in pre-existing $\sim_b$-equivalence classes.

Lemma 4.6 *The number of $\sim_b$-equivalence classes can never increase.*

Proof. Rule (e) does not increase the number of equivalence classes since the two new variables V_1 and V_2 are $\sim_b$-equivalent to the existing variables X and Y, respectively. Rule (f) does not increase the number of equivalence classes since the variable X remains in the $[U]_b$ equivalence class. Rules (a) through (d) and rule (g) can be seen not to increase the number of classes but they may decrease the number by merging two classes through equating two variables. □

[6] Definition of $\sim_b$ is given in Section 4.2.1

We call a variable, X, a $\succ_b$-*sink* if there does not exist a variable Z such that $X \succ_b Z$.

Lemma 4.7 *Each $\sim_b$-equivalence class contains at least one sink, with respect to $\succcurlyeq_b$.*

Proof. Otherwise there would be a relation $X \succ_b^+ X$, for some X, and the system would not be unifiable. □

Let us denote a variable U as an *f-peak* if U is the left-hand side of two equations satisfying the hypothesis of rule (f), and likewise an *e-peak* if the two equations satisfy the hypothesis of rule (e).

Lemma 4.8 *For any one $\sim_b$-equivalence class, say $[X]_b$, an application of rule (f) or rule (h) to a pair of equations, $U =^? t_1$ and $U =^? t_2$ such that $U \in [X]_b$, will reduce the number of $\succ_b$-sinks in $[X]_b$ by one.*

Proof. First we note that rules (a) through (c) and (i) are applied exhaustively before (f) or (h). Therefore, to apply rule (f) or rule (h) it must be that case that we have equations $U =^? exp(V, W)$, $U =^? exp(X, Y)$, where $V \neq X$ and $W \neq Y$. In addition, by rule (i) V will not be equal to an *exp*-rooted term. Thus V and X are sinks.
Consider rule (f) on the equations:

$$U =^? exp(V, W),\ U =^? exp(X, Y)$$

After applying rule (f) we have $V =^? exp(X, Z)$, removing V as a sink and reducing the total number of sinks by 1.
Consider rule (h) on the equations:

$$U =^? exp(V, W),\ U =^? exp(X, Y)$$

After applying rule (h) we have $V =^? exp(Z, Y')$ and $X =^? exp(Z, W')$, removing V and X as sinks but adding the new variable Z as a sink, thus reducing the total number of sinks by 1. □

Theorem 4.9 *Procedure $\mathfrak{R}_1$ terminates on any unification problem $\mathcal{S}$ modulo $\mathcal{E}_{AC}$ given in standard form.*

Proof. If a failure condition is detected the algorithm will terminate. Let us define the complexity measure

$$\langle\ n_1,\ n_2,\ n_3,\ n_4,\ n_5,\ n_6 \rangle$$

over $\mathcal{S}$ where n_1, n_2, n_4, n_5 and n_6 are natural numbers and n_3 is a finite sequence of natural numbers (of boounded length). The measure is ordered by (left-to-right) lexicographic ordering. We define each n_i as follows:

- n_1 = the number of $\sim_b$-equivalence classes.
- n_2 = the number of $\succ_b$-sinks.
- n_3 = the sequence $\langle m_1, \ldots, m_{n_1} \rangle$, where
  - there exists one $m_i \in \mathbb{N}$ for each $\sim_b$-equivalence class.

- each m_i is the number of e-peak *variables* in the i^{th} $\sim_b$-equivalence class.
- the m_i in $\langle m_1 \ldots m_{n_1} \rangle$ are ordered by $\succcurlyeq_m$.

- n_4 = the number of unsolved variables in $\mathcal{S}$ *excluding* the new exponent variables created in $\mathfrak{R}_1$.
- n_5 = the number of variables X such that X is a base variable of an *exp*-equation and $X = t$ such that t is an *exp*-rooted term. Thus, this is simply the number of variables rule (i) could be applied on.
- n_6 = the number of unsolved variables in $\mathcal{S}$.

Let us now examine the action of each composite rule on this measure.

Case 1, Rule (c1): i^*a^*
Exhaustive application of rule (i) clearly reduces n_5. In addition, the only measure rule (i) will possibly increase is n_6. Applications of rule (a) will clearly reduce n_6 and could reduce n_4. In addition, by equating two variables from two distinct $\sim_b$-equivalence classes the measure n_1 will be reduced. Now, rule (c1) could increase n_5 by applying rule (a) and equating a sink variable with a variable on the lhs of an *exp*-equation. That is, for the equations $U =^? exp(V, W),\ K =^? exp(X, Y)$ by equating K and V through rule (a), n_5 is increased. But, because K and V are not new *exp* exponent variables this will also reduce n_4. We need now to examine the action of rule (a) on n_3. In order to increase n_3 rule (a) could equate two variables X and Y such that $Y =^? exp(Y_1, Y_2),\ X =^? X_1 * X_2$. Because, rule (i) is applied first Y_1 must be a sink. Now, if X and Y are in different equivalence classes then by equating these two n_1 is decreased. Hence assume that X and Y are in the same class. If $X \succ_b Y$ or $Y \succ_b X$ then equating X and Y will produce a cycle in the dependency graph and cause failure. If X is a sink, then equating X and Y will reduce the number of sinks, i.e., n_2. If X is not a sink, then there must be an equation of the form $X =^? exp(X', X'')$ which means that X is already an e-peak variable. Hence equating X and Y will not increase the number of e-peak variables and n_3. That is, the resulting set of equations is $Y =^? exp(Y_1, Y_2),\ Y =^? X_1 * X_2,\ Y =^? exp(X', X'')$. But, this is just a single e-peak, because $Y =^? X_1 * X_2$ will be removed in an application of rule (e), and thus Y is a single e-peak variable.

Case 2, Rule (c2): $[b + c + d]\, a^*$
Rules (b) - (c) create the conditions for an application of rule (a). Therefore, rule (a) will be applied at each application of rule (c2) thus reducing n_6. By the same argument used in Case 1, if rule (c2) increases n_5 it will reduce n_4, if (a) increases n_3 it will reduce n_1 or n_2 and it will not increase n_1 or n_2.

Case 3, Rule (c3): $[b + c + d]^*\, a^* i^* e\, [b + c + d]^*\, a^*$
We first note that because of the prefix $[b + c + d]^*\, a^* i^*$ in rule (c3), for any pair of equations $U =^? exp(V, W)\ U =^? X * Y$, the variable V must be a sink, for otherwise rule (i) could have been applied. Also, there cannot be another $U =^? X' * Y'$ equation because rule (d) would have removed it. In addition, by cases 1 and 2 $[b + c + d]^*\, a^* i^*$ will not increase the measure.

Now, consider the results of applying rule (e). We get the following equations, $U =^? exp(V, W)$, $V =^? V_1 * V_2$, $X =^? exp(V_1, W)$, $Y =^? exp(V_2, W)$. By removing $U =^? X * Y$ we have removed an e-peak and reduced the m_i for the equivalence classes containing U. Since V is a sink $V =^? V_1 * V_2$ does not create a new e-peak. The equations $X =^? exp(V_1, W)$ and $Y =^? exp(V_2, W)$ could create new e-peaks if there are pre-existing equations $X =^? X_1 * X_2$ and/or $Y =^? Y_1 * Y_2$. But, these e-peaks would be added to classes which are *lower* in the $\succcurlyeq_m$ order. Thus, n_3 will be reduced and as the number of equivalence classes cannot increase, the overall measure is reduced.

Rule (e) could increase the number of sinks for single variable equivalence classes. This will occur for equations of the form $U =^? exp(X, Y)$, $U =^? V * V$, where V is a single variable equivalence class. This will add two new sinks, X_1 and X_2, to the class $[V]_b$. But, because of the prefix $[b + c + d]^* a^*$ in (c3), rule (c) will be applied to the equations $V =^? exp(X_1, Y)$ and $V =^? exp(X_2, Y)$. The result is that the number of sinks remains the same as before the application of rule (e).

Case 4, Rule (c4): $(i^* [b + c + d]^* a^*)^* (f + g + h) a^*$
For this rule we first note that because of the prefix $[b + c + d]^* a^* i^*$ for a pair of equations $U =^? exp(V, W)$, $U =^? exp(X, Y)$, V and X must be a sinks for otherwise rule (i) could have been applied. It can be seen that rules (f), (g) and (h), reduce the number of sinks. Therefore, n_2 is reduced. Rules (f) and (h) add new variables therefore n_4 and n_5 could increase. □

4.2.3. Soundness

Soundness of rules (a) through (i) is due to the following theorem.

Theorem 4.10 *Rules (a) through (i) are sound.*

Proof. The soundness of rules (e), (f) and (h) are due to Lemmas B.1, B.2 and B.3 respectively. The soundness of rule (i) follows from the AC-convergent system $\mathcal{R}_e$. Rule (a) is trivial. Rules (b), (c), (d) and (g) follow from soundness of cancellation [25]. □

4.2.4. Completeness

The following lemmas establish completeness of the algorithm in checking whether a given set of equations is $\mathcal{E}_{AC}$-unifiable.

Lemma 4.11 *Let S be a set of equations and θ be a ground $\mathcal{E}_{AC}$-unifier of S. If any of the inference rules (a)–(d) is applicable to S, then θ is an $\mathcal{E}_{AC}$-unifier of the resulting system.*

Proof. Let θ be a ground $\mathcal{E}_{AC}$-unifier of S. The result for rule (a) is trivial. If either rule (b) or (c) is applicable the result follows from Lemma A.1. If rule (d) is applicable the result follows from Lemma A.2. □

Lemma 4.12 *Let S be a set of equations and θ be a ground $\mathcal{E}_{AC}$-unifier of S. If the inference rules (e) or (i) is applicable to S, then the resulting system has a unifier $\hat{\theta}$ such that*

1. $\theta \equiv^{Var(S)}_{\mathcal{E}_{AC}} \hat{\theta}$ and
2. $\exists \gamma \colon \hat{\theta} = \gamma \circ \theta = \gamma \uplus \theta|_{Var(S)}$.

Proof. Let θ be a ground $\mathcal{E}_{AC}$-unifier of S. If rule (e) is applicable the result follows from Lemma A.3. Rule (i) is due to $\mathcal{R}_e$. □

Lemma 4.13 *Let S be a set of equations that contains two distinct equations $\{U =^? exp(V, W),\ U =^? exp(X, Y)\}$ where U, V, W, X, Y are variables. If θ is a ground $\mathcal{E}_{AC}$-unifier of S, then one of the rules (f), (g) or (h) can be applied to S such that the resulting system has a unifier $\hat{\theta}$ with the properties*

1. $\theta \equiv^{Var(S)}_{\mathcal{E}_{AC}} \hat{\theta}$ and
2. $\exists \gamma \colon \hat{\theta} = \gamma \circ \theta = \gamma \uplus \theta|_{Var(S)}$.

Proof. Let θ be a ground $\mathcal{E}_{AC}$-unifier of S. Lemma A.4 shows that one of the rules (f), (g) or (h) can be applied resulting in a unifier $\hat{\theta}$ with the required properties. □

Putting it all together,

Lemma 4.14 (Main Lemma) *Let S be a set of equations that is not in $\{exp, *\}$-solved form and let σ be a ground $\mathcal{E}_{AC}$-unifier for $\mathcal{S}$. Then one of the steps (a)–(i) can be applied to S and the resulting set of equations has a $\mathcal{E}_{AC}$-unifier $\hat{\sigma}$ such that*

1. $\sigma \equiv^{Var(S)}_{\mathcal{E}_{AC}} \hat{\sigma}$ and
2. $\exists \gamma \colon \hat{\sigma} = \gamma \circ \sigma = \gamma \uplus \sigma|_{Var(S)}$.

4.3. The Set $\mathcal{V}(\mathcal{S})$

The set $\mathcal{V}(\mathcal{S})$ and its partitioning are needed because running the procedure $\mathfrak{R}_1$ on $\mathcal{S}$ and then using AC-unification is not sufficient. This is due to the fact that the AC-unification algorithm may equate variable such that new applications of the inference rules would be required. The variables from $\mathcal{S}$ such that if they are equated could possibly create the condition for additional applications of the inference rules are exactly $\mathcal{V}(\mathcal{S})$. It is possible that the AC-unification algorithm will set some variables in $\mathcal{V}(\mathcal{S})$ equal to each other such that new applications of the inference rules could be necessary. The way to avoid running inference rules without termination is to guess a successful partition of $\mathcal{V}(\mathcal{S})$, i.e., to consider all possible variables that general AC-unification may equate and could thus necessitate additional applications of the inference rules. The key idea is to set the variables which are in the same subset of a partition equal to each other. This can be seen in the following example, that demonstrates the necessity of the partitioning of $\mathcal{V}(\mathcal{S})$ and also the need for

rule (f).

Consider the unification problem

$$\left\{ \begin{array}{c} U =^? exp(U_1, U_2),\ V =^? exp(V_1, V_2), \\ V_2 =^? X \circledast U_2,\ Y =^? W \circledast U,\ Y =^? W \circledast V \end{array} \right\}$$

If we did *not* first partition $\mathcal{V} = \{U, V, U_2, V_2, W, X\}$ and run Step 2 none of the inference rules (a)-(i) would apply. Thus the set of equations passed to the general AC-unification algorithm would be:

$$\left\{ \begin{array}{c} U =^? exp(U_1, U_2),\ V =^? exp(V_1, V_2), \\ V_2 =^? X \circledast U_2,\ Y =^? W \circledast U,\ Y =^? W \circledast V \end{array} \right\}$$

Clearly this has no general AC-unifier since U and V, and thus U_2 and V_2, will have to be equated. On the other hand, if we choose the partition $\{\{U, V\}, \{W\}, \{U_2\}, \{V_2\}, \{X\}\}$ to begin with, then after applying step (1) we have the following set of equations

$$\left\{ \begin{array}{c} U =^? exp(U_1, U_2),\ U =^? exp(V_1, V_2), \\ V_2 =^? X \circledast U_2,\ Y =^? W \circledast U \end{array} \right\}$$

Now after applying rule (f) to the first two equations we will eventually reach a $\{exp, *\}$-solved form that is AC-unifiable.

The purpose of the partitioning of $\mathcal{V}(\mathcal{S})$ is to "guess" which variables the general AC-unification step may equate resulting in the need to apply one of the rules from $\mathfrak{R}_1$ again. Therefore we need to ensure that new variables created after partitioning (i.e., during $\mathfrak{R}_1$) do not need to be considered. That is, we need to argue that it is sufficient to collect and partition $\mathcal{V}(\mathcal{S})$ in Step (1) of the algorithm rather then after applying the inference rules. This is an important property, as it allows us to avoid a series of repeated applications of $\mathfrak{R}_1$ followed by collecting and partitioning an updated $\mathcal{V}(\mathcal{S})$. In this case it is sufficient to show that the new variables created in the running of $\mathfrak{R}_1$ will not appear *both* on the left-hand side of either a $*$ or exp equation and in a $\circledast$-rooted equation. This is due to the fact that, in standard form, for a new variable to cause an application of one of the rules in $\mathfrak{R}_1$ it would need to be the left-hand variable of a new equation and then be equated, by the AC-unification algorithm, to another left-hand side variable.

Lemma 4.15 *Let Z be a <u>new</u> exponent variable created in $\mathfrak{R}_1$. Then it will never be the case, through application of $\mathfrak{R}_1$, that $Z =^? exp(X, Y)$ or $Z =^? X * Y$.*

Proof. First note that rule (a) is applied in a special way. Let $\mathcal{EQ} \uplus \{U =^? Z\}$ such that Z is a new variable created by $\mathfrak{R}_1$ and U is a pre-existing variable. Then all occurrences of Z are replaced by U.

Let $\mathcal{S}$ be a $\mathcal{E}_{AC}$ unification problem. We start by examining rules (f), (h) and (i), as they are the only rules creating and adding new exponent variables. Rule (f), (h) and (i), never create an equation of the form $Z =^? t$ such that $t|_\lambda = exp$ or $*$ and Z

is new. Therefore, to move a new exponent variable to the left-hand side of an *exp* or $*$ equation we must be able to obtain one of the following sets of equations:

(i) $Z' =^? exp(X, Y),\ U =^? exp(V, Z')$, or

(ii) $Z' =^? X * Y,\ U =^? exp(V, Z')$

where Z' is a *new variable*: for otherwise, i.e., if Z' is not a new variable, then equating a pre-existing variable to Z would remove Z by rule (a). But this essentially 'passes the buck' to Z'. Ultimately equations of the form (i) or (ii) must be created by the application of a single rule in $\mathfrak{R}_1$. $\mathfrak{R}_1$ contains no such rule and therefore the result follows. □

Directly from this lemma we get

Lemma 4.16 *Let S be a set of equations that are input to $\mathfrak{R}_1$ and S' be an output of $\mathfrak{R}_1$. Then*

$$\mathcal{V}(S') \cap lhs(S'|_{exp} \cup S'|_{*}) \subseteq \mathcal{V}(S)$$

4.4. Correctness of the Overall Algorithm

The termination of the overall algorithm follows from the termination of $\mathfrak{R}_1$ (i.e., Step 2), since the termination of Steps 1 and 3 is obvious. We now discuss its correctness.

First of all, note that the original problem $\mathcal{S}$ is modified by Step 2, first by adding new equations after a partition is selected and then applying rules (a) – (i). Of these, rules (e), (f), (h) and (i) introduce fresh variables. We give the following result that fresh variables do not effect the solution.

Lemma 4.17 *Let $\mathcal{S}$ be a $\mathcal{E}_{AC}$-unification problem and σ be a ground $\mathcal{E}_{AC}$-unifier for $\mathcal{S}$. Then there exist a partition Π_i of $\mathcal{V}(S)$, an index j and a substitution $\hat{\sigma}$ such that $\hat{\sigma}$ is an $\mathcal{E}_{AC}$-unifier of $\mathcal{S}_j^{\Pi_i}$ and $\sigma \equiv_{Var(S)} \hat{\sigma}$. Besides, σ can be uniquely extended to $\hat{\sigma}$.*

Proof. Let Π_i be the partition induced by σ on $\mathcal{V}$: variables x and y in $\mathcal{V}$ belong to the same subset if and only if $\sigma(x) \equiv_{AC} \sigma(y)$. We can apply the steps of $\mathfrak{R}_1$ on $\mathcal{S}^{\Pi_i}$ based on σ (essentially guided by σ). By Lemma 4.14, every step satisfies the property in the lemma. Since the procedure always terminates, the result follows. □

Lemma 4.18 *Let $\mathcal{S}$ be a $\mathcal{E}_{AC}$-unification problem in $\{exp, *\}$-solved form. Let $\mathcal{S}_1 = \mathcal{S}|_{exp} \cup \mathcal{S}|_{*}$, $\mathcal{S}_2 = \mathcal{S}|_{\circledast}$ and $V_c = lhs(\mathcal{S}_1) \cap Var(\mathcal{S}_2)$. If $\mathcal{S}$ has a V_c-discriminating $\mathcal{E}_{AC}$-unifier, then $\mathcal{S}$ is AC-unifiable.*

Proof. Let θ be a V_c-discriminating $\mathcal{E}_{AC}$-unifier of $\mathcal{S}$. Restricting θ to $\mathcal{S}_2$, it is not hard to see that θ can be split into $\theta|_{V_c} \circ \sigma$, where σ is an AC-unifier for $\mathcal{S}_2$ where variables in V_c are treated as constants.[7] Since $\mathcal{S}_1$ is in dag-solved form, there is

[7] This is similar to what Baader and Schulz do with their concept of *theory index*. We can view *exp* and $*$ as the first theory and $\circledast$ as the second.

a most general standard unifier, say γ, for $\mathcal{S}_1$. We claim that $\delta = \gamma \cup \sigma$ is a sequential AC-unifier for $\mathcal{S}$. δ is clearly a valid substitution since the domains of γ and σ are disjoint. Furthermore, γ and σ, written as equations, cannot be cyclic (i.e., there cannot be a variable X such that $X \succ^+ X$) because that would contradict the assumption that θ is a unifier of $\mathcal{S}$. □

Lemma 4.19 *The $\mathcal{E}_{AC}$ algorithm is sound: if $\mathcal{S}$ is an $\mathcal{E}_{AC}$-unification problem and the algorithm terminates on $\mathcal{S}$ without failure, then $\mathcal{S}$ is $\mathcal{E}_{AC}$-unifiable.*

Proof. The algorithm reduces the problem to a general AC-unification problem with uninterpreted functions symbols $*$ and exp. Let $\mathcal{S}_j^{\Pi_i}$ be the output of Step 2. By Lemma 4.10 any $\mathcal{E}_{AC}$-unifier of $\mathcal{S}_j^{\Pi_i}$ is a unifier of $\mathcal{S}$. Thus if σ is an AC-unifier of $\mathcal{S}_j^{\Pi_i}$, then it is also an $\mathcal{E}_{AC}$-unifier for $\mathcal{S}$. □

We next give the proof of completeness for the entire method.

Lemma 4.20 *The $\mathcal{E}_{AC}$ algorithm is complete.*

Proof. Assume $\mathcal{S}$ is $\mathcal{E}_{AC}$-unifiable with a ground unifier σ over a set of constants C. By Lemma 4.17 there exist a partition Π_i and an index j such that σ can be uniquely extended to $\hat{\sigma}$ which is a unifier of $\mathcal{S}_j^{\Pi_i}$. Let $\mathcal{S}' = \mathcal{S}_j^{\Pi_i}$. Now, as $\mathcal{S}'$ is in $\{exp, *\}$-solved form we need only show that $\hat{\sigma}$ is a V_c-discriminating unifier for $\mathcal{S}'|_{\circledast}$ where V_c is as in the statement of Lemma 4.18, i.e., it is the set of variables that appear in $\mathcal{S}'|_{\circledast}$ that also appear on the left-hand sides of non-$\circledast$ equations. Consider the partition Π_i on the set $\mathcal{V}$. After applying rule (a) we can consider each subset induced by Π_i as being replaced by a unique variable representative for that subset. We denote this reduced $\mathcal{V}$ as $\mathcal{V}'$. The new variables created by rule (f), (h) and (i) are never on the left-hand sides of equations by Lemma 4.15. Therefore, by Lemma 4.18, $\mathcal{S}_j^{\Pi_i}$ is AC-unifiable.

If $\mathcal{S}$ is AC-unifiable, then it is also obviously $\mathcal{E}_{AC}$-unifiable. □

Hence we get the main result,

Theorem 4.21 *Unifiability modulo $\mathcal{E}_{AC}$ is decidable.*

Proof. Let $\mathcal{S}$ be a $\mathcal{E}_{AC}$-unification problem. If the algorithm terminates on $\mathcal{S}$ without failure, i.e., Steps 2 and 3 do not encounter failure, then $\mathcal{S}$ is $\mathcal{E}_{AC}$-unifiable. This follows from Lemma 4.19. If $\mathcal{S}$ is $\mathcal{E}_{AC}$-unifiable, then the algorithm terminates without failure. This follows from Lemma 4.17 and Lemma 4.20. □

4.4.1. Computing Complete Sets of Unifiers

The algorithm can indeed be used to compute a finite complete set of unifiers for a given problem by considering all possible $\mathcal{S}_j^{\Pi_i}$s.

Directly from Section 4.4 we get the following result.

Lemma 4.22 *Let $\mathcal{S}$ be a $\mathcal{E}_{AC}$-unification problem and θ be a ground $\mathcal{E}_{AC}$-unifier of $\mathcal{S}$ (over a set of constants $\mathcal{C}$). Then there exist a partition Π_i of $\mathcal{V}(S)$, an index j and a substitution σ such that*

1. *σ is an AC-unifier of $\mathcal{S}_j^{\Pi_i}$,*
2. *σ is an $\mathcal{E}_{AC}$-unifier of $\mathcal{S}$, and*
3. *$\sigma \leq_{\mathcal{E}_{AC}}^{Var(S)} \theta$.*

Consider an arbitrary unifier θ for a $\mathcal{E}_{AC}$-unification problem $\mathcal{S}$. We can replace each variable in the range of θ with a distinct new constant. That is, for each variable $y \in VRan(\theta) = \bigcup_{x \in Dom(\theta)} Var(\theta(x))$, replace y with a new constant. Therefore, for each unifier we can construct a corresponding ground unifier, say θ_g, and by Lemma 4.22 the algorithm will produce, in *finitely many nondeterministic steps,* an $\mathcal{E}_{AC}$-unifier more general than θ_g. Thus we can get a complete set of unifiers by exhaustively considering all $\mathcal{S}_j^{\Pi_i}$.

Since AC-unification is finitary, $\mathcal{E}_{AC}$-unification is also finitary. This follows from the fact that an $\mathcal{E}_{AC}$-unification problem is converted to a general AC-unification problem at the end. On the other hand, the complexity of $\mathcal{E}_{AC}$-unification is also related to AC-unification, which was shown to be NP-complete in [14]. $\mathcal{E}_{AC}$-unification thus turns out to be NP-hard since we can reduce any AC-unification problem to an $\mathcal{E}_{AC}$-unification problem.

4.5. Unification modulo $\mathcal{E}_A$

Section 3 demonstrates that associativity is in a way unavoidable when doing unification modulo $\mathcal{F}$ or any of its extensions. Then, if we wish to consider unification modulo $\mathcal{E}_C$ we must also include associativity and are thus effectively doing unification modulo $\mathcal{E}_{AC}$.

For $\mathcal{E}_A$, it is not hard to see that the unification algorithm for $\mathcal{E}_{AC}$ can be modified for $\mathcal{E}_A$ by replacing the general AC-unification algorithm with a general A-unification algorithm. Decidability of general A-unification was shown in [4]. The consequence of this replacement is that the unification type of $\mathcal{E}_A$-unification is not finitary. This is due to the fact that A-unification is infinitary [22].

Theorem 4.23 *Unifiability modulo $\mathcal{E}_A$ is decidable.*

It is not hard to show that unifiability modulo $\mathcal{E}_C$ is decidable too; we omit the details here.

5. Unification modulo $\mathcal{F}_{AC}$

Here we consider extensions of $\mathcal{F}$. For the subtheory $\mathcal{F}_{AC}$, we consider equations of the following standard forms;

$$U =^? V,\ U =^? V \circledast Y \ \text{ and } U =^? exp(V, Y)$$

where U, V and Y are variables. So the only standard form equation not considered from the previous section is $U = V * Y$.

When the $\mathcal{E}_{AC}$ algorithm is run on a $\mathcal{F}_{AC}$-unification problem $\mathcal{S}$, the rules (d) and (e) will never be applied as none of the equations has $*$ in them, i.e., $\mathcal{S}|_* = \emptyset$. In addition, as rule (e) is the only rule creating new equations whose right-hand sides have $*$ as a root symbol, no such equations will be introduced. Therefore, it can be seen that a $\mathcal{F}_{AC}$-unification problem is just an $\mathcal{E}_{AC}$-unification problem without any equations involving $*$ symbols. An immediate consequence is the following result.

Theorem 5.1 *The $\mathcal{E}_{AC}$-unification Algorithm is (also) a correct $\mathcal{F}_{AC}$-unification Algorithm.*

In addition, we also have:

Corollary 5.2 *Unifiability modulo $\mathcal{F}_A$ is decidable.*

In both of these cases, unification is infinitary because of A-unification being infinitary.

For the theory $\mathcal{F}_C$, the unification problem is finitary since it is essentially the same as $\mathcal{F}_{AC}$.

Corollary 5.3 *Unifiability modulo $\mathcal{F}_{AC}$ is decidable.*

This result is not really new. It is implied in [21] and stated explicitly in [9, 11].

6. Conclusions

We have established decidability results for unification theories involving two properties of modular exponentiation, a critical component of most cryptographic protocols. The first property expresses how repeated exponentiation gets simplified into multiplication at the exponent level, whereas the second property captures how exponentiation distributes when the base involves multiplication. Since multiplication could in general be using different moduli, two multiplication operators are used.

The decision algorithm is given as an inference system derived from the properties of the above two equational properties. A hierarchical combination approach is developed to give unification algorithms for various subtheories $\mathcal{F}$ and $\mathcal{E}$ augmented with subsets of associative and commutative properties of $\circledast$, in which A and AC unification algorithms are invoked as plug-ins. As a result, subtheories in which AC-unification is invoked are finitary, whereas those in which A-unification is used are not.

The reader should notice that no property of $*$ is used. It may be possible to make $*$ commutative and still have decidability, but as shown in other papers, making $*$ to be AC will make the unification problems undecidable.

A more general issue is that of unification of hierarchical combinations of theories when the individual theories are decidable and finitary. Here one could view the exponentiation axioms as the "higher" theory and properties of $\circledast$ as the "lower" theory. A general combination technique along this line will be very interesting to explore.

Table 2 gives a summary of our results for the unification problem for equational systems that contain some type of exponentiation.

Ref	Equational Theory	Unification Problem: Results
[19]	$exp(X,1) = X$ and $exp(exp(X,Y),Z) = exp(X, Y \circledast Z)$ along with abelian group axioms for $\circledast$	NP-complete
[17]	Two theories, denoted $\mathcal{E}_1$ and $\mathcal{E}_2$. $\mathcal{E}_1$ is $\mathcal{E}_{AC}$ with abelian group axioms for $*$ and the axioms $exp(X,1) = X$, $exp(1,X) = 1$ and $y \circledast 1 = Y$. $\mathcal{E}_2$ adds the axiom $X \circledast i(X) = 1$, i being the inverse function, to the theory $\mathcal{E}_1$.	Undecidable for both $\mathcal{E}_1$ and $\mathcal{E}_2$
[16]	Two main results: Theory $\mathcal{E}_3$ consists of an abelian group for operator $*$ along with the axioms $exp(X,1) = X$, $exp(1,X) = 1$ and $exp(X*Y,Z) = exp(X,Z)*exp(Y,Z)$. Theory $\mathcal{E}_4$ is $\mathcal{E}_3 \cup \mathcal{E}_{AC}$ plus the axiom $X \circledast 1 = X$.	$\mathcal{E}_3$ is decidable and $\mathcal{E}_4$ is undecidable.
[18]	Two theories, denoted $\mathcal{E}$ and $\mathcal{E}_0$. $\mathcal{E}$ consists of abelian group axioms for $*$, and the axioms $exp(X,1) = X$, $exp(1,X) = 1$, $exp(X*Y,Z) = exp(X,Z) * exp(Y,Z)$, and $exp(exp(X,Y),Z) = exp(X, Y*Z)$. $\mathcal{E}_0$ is the same as $\mathcal{E}$ but the axiom $exp(exp(X,Y),Z) = exp(X,Y*Z)$ is replaced with the axiom $exp(exp(X,Y),Z) = exp(exp(X,Z),Y)$.	$\mathcal{E}$ is undecidable and $\mathcal{E}_0$ is decidable.
[13]	Two theories: $\mathcal{E}_1$ consists of the two axioms $exp(g(X),Y) = g(X \circledast Y)$ and $exp(X*Y,Z) = exp(X,Z) * exp(Y,Z)$. $\mathcal{E}_2$ is $\mathcal{E}_1$ with abelian group axioms for $\circledast$ and $*$.	$\mathcal{E}_1$ is decidable and $\mathcal{E}_2$ is undecidable.

Table 2: Results for E-unification with exponentiation.

References

[1] S. Anantharaman, H. Lin, C. Lynch, P. Narendran, M. Rusinowitch, Unification Modulo Homomorphic Encryption. In: S. Ghilardi, R. Sebastiani (eds.), *Frontiers of Combining Systems*. Lecture Notes in Computer Science 5749, Springer Berlin / Heidelberg, 2009, 100–116. 10.1007/978-3-642-04222-5_6.

[2] S. Anantharaman, H. Lin, C. Lynch, P. Narendran, M. Rusinowitch, Cap unification: application to protocol security modulo homomorphic encryption. In: D. Feng, D. A. Basin, P. Liu (eds.), *ASIACCS*. ACM, 2010, 192–203.

[3] F. Baader, T. Nipkow, *Term rewriting and all that*. Cambridge University Press, New York, NY, USA, 1998.

[4] F. Baader, K. U. Schulz, Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures. *Journal of Symbolic Computation* **21** (1996) 2, 211 – 243.

[5] F. Baader, W. Snyder, Unification Theory. In: J. A. Robinson, A. Voronkov (eds.), *Handbook of Automated Reasoning*. Elsevier and MIT Press, 2001, 445–532.

[6] F. Baader, C. Tinelli, Combining Decision Procedures for Positive Theories Sharing Constructors. In: S. Tison (ed.), *Rewriting Techniques and Applications*. Lecture Notes in Computer Science 2378, Springer Berlin / Heidelberg, 2002, 151–246.

[7] C. Boyd, A. Mathuria, *Protocols For Key Establishment And Authentication*. Springer, 2002.

[8] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, Complexity results for security protocols with Diffie-Hellman exponentiation and commuting public key encryption. *ACM Trans. Comput. Logic* **9** (2008), 24:1–24:52.

[9] H. Comon-Lundh, S. Delaune, The Finite Variant Property: How to Get Rid of Some Algebraic Properties. In: J. Giesl (ed.), *Rewriting Techniques and Applications*. Lecture Notes in Computer Science 3467, Springer, 2005, 294–307.

[10] E. Domenjoud, F. Klay, C. Ringeissen, Combination techniques for non-disjoint equational theories. In: A. Bundy (ed.), *Automated Deduction CADE-12*. Lecture Notes in Computer Science 814, Springer Berlin / Heidelberg, 1994, 267–281.

[11] S. Escobar, J. Meseguer, R. Sasse, Variant Narrowing and Equational Unification. *Electronic Notes Theor. Comput. Science* **238** (2009) 3, 103–119.

[12] J.-P. Jouannaud, C. Kirchner, Solving Equations in Abstract Algebras: A Rule-Based Survey of Unification. In: *Computational Logic - Essays in Honor of Alan Robinson*. 1991, 257–321.

[13] D. Kapur, A. M. Marshall, P. Narendran, Unification modulo a partial theory of exponentiation. In: M. Fernandez (ed.), *Proceedings of UNIF-2010*. EPTCS 42, 2010, 12–23.

[14] D. Kapur, P. Narendran, Complexity of unification problems with associative-commutative operators. *Journal of Automated Reasoning* **9** (1992), 261–288.

[15] D. Kapur, P. Narendran, Double-exponential Complexity of Computing a Complete Set of AC-Unifiers. In: *Logic in Computer Science*. IEEE Computer Society, 1992, 11–21.

[16] D. Kapur, P. Narendran, L. Wang, An E-unification Algorithm for Analyzing Protocols That Use Modular Exponentiation. In: R. Nieuwenhuis (ed.), *Rewriting Techniques and Applications*. Lecture Notes in Computer Science 2706, Springer, 2003, 165–179.

[17] D. Kapur, P. Narendran, L. Wang, Undecidability of unification over two theories of modular exponentiation. In: *Seventeenth International Workshop on Unification (UNIF-2003), Valencia, Spain*. 2003.

[18] D. Kapur, P. Narendran, L. Wang, A Unification Algorithm for Analysis of Protocols with Blinded Signatures. In: *Mechanizing Mathematical Reasoning*. Lecture Notes in Computer Science 2605, Springer Berlin / Heidelberg, 2005, 433–451.

[19] C. Meadows, P. Narendran, A Unification Algorithm for the Group Diffie-Hellman Protocol. In: *Proceedings of WITS-2002*. 2002, 14–15.

[20] J. Millen, V. Shmatikov, Symbolic protocol analysis with an Abelian group operator or Diffie–Hellman exponentiation. *J. Comput. Security* **13** (2005), 695–695.

[21] P. Narendran, F. Pfenning, R. Statman, On the Unification Problem for Cartesian Closed Categories. *J. Symbolic Logic* **62** (1997) 2, 636–647.

[22] G. Plotkin, Building in Equational Theories. *Machine Intelligence* **7** (1972), 73–90.

[23] A. Rubio, A Fully Syntactic AC-RPO. *Information and Computation* **178** (2002) 2, 515 – 533.

[24] M. Schmidt-Schauss, Unification in a combination of arbitrary disjoint equational theories. *J. Symbolic Computation* **8** (1989), 51–99.

[25] W. Snyder, *A Proof Theory for General Unification*. Progress in Computer Science and Applied Logic 11, Birkhauser, 1991.

[26] E. Tidén, S. Arnborg, Unification Problems with One-Sided Distributivity. *J. Symbolic Computation* **3** (1987) 1/2, 183–202.

A. Some properties of $\mathcal{E}_{AC}$

We mentioned earlier that $\mathcal{E}_{AC}$ has the following AC-convergent system

1. $exp(exp(X,Y),Z) \rightarrow exp(X, Y \circledast Z)$
2. $exp(X * Y, Z) \rightarrow exp(X,Z) * exp(Y,Z)$

which we called $\mathcal{R}_e$. In fact, there is another AC-convergent system where the distributivity axiom is oriented differently:

$$\begin{array}{rcl} exp(X,Z) * exp(Y,Z) & \rightarrow & exp((X * Y), Z) \\ exp(exp(X,Y),Z) & \rightarrow & exp(X, Y \circledast Z) \\ exp(X,(Y \circledast Z)) * exp(W,Z) & \rightarrow & exp((exp(X * Y) * W), Z) \\ exp(X,Z) * exp(W,(Y \circledast Z))) & \rightarrow & exp((X * exp(W,Y)), Z) \\ exp(X,(Y \circledast Z)) * exp(W,(V \circledast Z)) & \rightarrow & exp((exp(X,Y) * exp(W,V)), Z) \end{array}$$

We refer to this system by $\mathcal{R}_e^{'}$. This system can be shown to be terminating via a reduction order based on the size of the terms, the number of occurrences of variables in terms and, for the second rule, the number of exp equations. Alternatively, the system can be shown to be terminating by use of Rubio's AC-RPO [23] and the symbol ordering $* > exp > \circledast$.

Lemma A.1

1. *For all terms* s, t_1, t_2, $exp(s,t_1) =_{\mathcal{E}_{AC}} exp(s,t_2)$ *if and only if* $t_1 =_{\mathcal{E}_{AC}} t_2$.
2. *For all terms* s_1, s_2, t, $exp(s_1,t) =_{\mathcal{E}_{AC}} exp(s_2,t)$ *if and only if* $s_1 =_{\mathcal{E}_{AC}} s_2$.

Proof. We only prove the second statement of the lemma. Consider the system $\mathcal{R}_e^{'}$. Assume without loss of generality that s_1, s_2 and t are irreducible. If both $exp(s_1,t)$ and $exp(s_2,t)$ are irreducible, then we are done. Otherwise, suppose $exp(s_1,t)$ is reducible. Since the only rule that can be applied is the second one, s_1 can be written as $exp(s_{11}, t_{11})$, where s_{11} and t_{11} are irreducible. Hence the normal form of $exp(s_1,t)$ is $exp(s_{11}, t_{11} \circledast t)$. Now it cannot be that $exp(s_2,t)$ is irreducible, since t cannot be AC-equivalent to $t_{11} \circledast t$. Thus $s_2 =_{AC} exp(s_{21}, t_{21})$, and thus the normal form of $exp(s_2,t)$ is $exp(s_{21}, t_{21} \circledast t)$. The two normal forms must be A-equivalent, which implies that $s_{11} =_{AC} s_{21}$ and $t_{11} \circledast t =_{AC} t_{21} \circledast t$. Since the latter implies that $t_{11} =_{AC} t_{21}$, we are done. □

Lemma A.2 *For all terms* s_1, s_2, t_1, t_2, $s_1 * s_2 =_{\mathcal{E}_{AC}} t_1 * t_2$ *if and only if* $s_1 =_{\mathcal{E}_{AC}} t_1$ *and* $s_2 =_{\mathcal{E}_{AC}} t_2$

Proof. There is no rule in $\mathcal{R}_e$ that has $*$ as the root symbol of its left-hand side. □

Lemma A.3 *For all terms* s_1, s_2, t_1, t_2, $exp(s_1,s_2) =_{\mathcal{E}_{AC}} t_1 * t_2$ *if and only if there exist terms* s_{11} *and* s_{12} *such that* $s_1 =_{\mathcal{E}_{AC}} s_{11} * s_{12}$, $t_1 =_{\mathcal{E}_{AC}} exp(s_{11}, s_2)$ *and* $t_2 =_{\mathcal{E}_{AC}} exp(s_{12}, s_2)$.

Lemma A.1 justifies the inference rules (b) and (c), Lemma A.2 justifies (d) and Lemma A.3 justifies rule (e).

Rules (f), (g) and (h) are interesting in that they each correspond to valid syntactic transformations of two equations of the type $U =^{?} exp(V, W)$, $U =^{?} exp(X, Y)$. We argue in the following lemma that (f), (g) and (h) are complete in the sense that they encompass all such valid transformations.

Lemma A.4 *For all terms* s_1, s_2, t_1, t_2, $exp(s_1, s_2) =_{\mathcal{E}_{AC}} exp(t_1, t_2)$ *if and only if one of the following holds:*
(a) $s_1 =_{\mathcal{E}_{AC}} t_1$, $s_2 =_{\mathcal{E}_{AC}} t_2$.
(b) $\exists\alpha$: $s_1 =_{\mathcal{E}_{AC}} exp(t_1, \alpha)$, $t_2 =_{\mathcal{E}_{AC}} s_2 \circledast \alpha$.
(c) $\exists\alpha$: $t_1 =_{\mathcal{E}_{AC}} exp(s_1, \alpha)$, $s_2 =_{\mathcal{E}_{AC}} t_2 \circledast \alpha$.
(d) $\exists\alpha_1, \alpha_2, s_3$: $t_1 =_{\mathcal{E}_{AC}} exp(s_3, \alpha_1)$, $s_1 =_{\mathcal{E}_{AC}} exp(s_3, \alpha_2)$, *and* $s_2 \circledast \alpha_1 =_{\mathcal{E}_{AC}} t_2 \circledast \alpha_2$.

Proof. Here again we consider the *AC*-convergent system $\mathcal{R}_e^{'}$.

Without loss of generality assume that s_1, s_2, t_1 and t_2 are in normal form modulo $\mathcal{R}_e^{'}$. If any of (a) through (d) hold we can see that $exp(s_1, s_2) =_{\mathcal{E}_{AC}} exp(t_1, t_2)$. It remains to be shown that no additional rules are required. If both $exp(s_1, s_2)$ and $exp(t_1, t_2)$ are in normal form, then case (a) is the only possibility. If either one is reducible then the reduction has to be by the second rewrite rule. Thus if one of $exp(s_1, s_2)$ and $exp(t_1, t_2)$ is reducible and the other is not, then one of cases (b) or (c) has to be satisfied. If both are reducible then both s_1 and t_1 must be *exp*-rooted terms. Let $s_1 = exp(s, \alpha_1)$ and $t_1 = exp(s', \alpha_2)$. Thus $exp(s_1, s_2) \rightarrow_{\mathcal{R}_e^{'}} exp(s, \alpha_1 \circledast s_2)$ and $exp(t_1, t_2) \rightarrow_{\mathcal{R}_e^{'}} exp(s', \alpha_2 \circledast t_2)$. Since the terms $exp(s, \alpha_1 \circledast s_2)$ and $exp(s', \alpha_2 \circledast t_2)$ must be irreducible, it must be that $s =_{AC} s'$, satisfying case (d). □

B. Lemmas for Theorem 4.10

We justify the use of rule (e) with the following lemma.

Lemma B.1 *Every unifier of the set of equations*

$$\{U =^? exp(V, W),\ V =^? V_1 * V_2,\ X =^? exp(V_1, W),\ Y =^? exp(V_2, W)\}$$

is a unifier of the set of equations $\{U =^? exp(V, W),\ U =^? X * Y\}$.

Proof. (This is equivelent to the proof first presented in [26]) Let σ be a unifier for

$$\{U =^? exp(V, W),\ V =^? V_1 * V_2,\ X =^? exp(V_1, W),\ Y =^? exp(V_2, W)\}$$

and consider σ restricted to the varibles $\{U, V, W, X, Y\}$, denoted as σ_Δ. Then

$$\begin{aligned}
\sigma_\Delta(X) &= exp(\sigma(V_1), \sigma_\Delta(W)),\\
\sigma_\Delta(Y) &= exp(\sigma(V_2), \sigma_\Delta(W)),\\
\sigma_\Delta(V) &= \sigma(V_1) * \sigma(V_2), \quad \text{and}\\
\sigma_\Delta(W) &= \sigma(W).
\end{aligned}$$

Plugging these into $exp(V, W)$ and $X * Y$, we get

$$exp((\sigma(V_1) * \sigma(V_2)), \sigma(W)) \quad \text{and} \quad exp(\sigma(V_1), \sigma(W)) * exp(\sigma(V_2), \sigma(W))$$

which are equivalent modulo $\mathcal{E}_{AC}$. □

Rules (f) and (h) also correspond to non-trivial properties and we therefore prove similar lemmas.

Lemma B.2 *Every unifier of the set of equations* $\{U =^? exp(X, Y),\ V =^? exp(X, Z),\ Y =^? Z \circledast W\}$ *is a unifier of the set of equations* $\{U =^? exp(V, W),\ U =^? exp(X, Y)\}$.

Proof. Let σ be a unifier for

$$\{U =^? exp(X, Y),\ V =^? exp(X, Z),\ Y =^? Z \circledast W\}$$

Then, $\sigma|_\Delta$, where $\Delta = \{U, X, Y, V, W\}$, is a unifier for

$$\{U =^? exp(V, W),\ U =^? exp(X, Y)\}$$

First, we can note that

$$\sigma|_\Delta(exp(X, Y)) = \sigma_\Delta(exp(X, Y)) = exp(\sigma_\Delta(X), \sigma_\Delta(Y)).$$

In addition, $\sigma|_\Delta(Y) = \sigma(Z) \circledast \sigma(W)$ and $\sigma|_\Delta(V) = exp(\sigma(X), \sigma(Z))$.
Plugging into $exp(\sigma_\Delta(X), \sigma_\Delta(Y))$, we get

$$exp(\sigma(X), (\sigma(Z) \circledast \sigma(W))) =_{\mathcal{E}_{AC}} exp(exp(\sigma(X), \sigma(Z)), \sigma(W))$$

This is equivalent to $exp(\sigma(V), \sigma(W))$. □

Lemma B.3 *Every unifier of the set of equations* $\Gamma = \{U =^? exp(Z, L),\ V =^? exp(Z, Y^{'}),\ X =^? exp(Z, W^{'})\ L =^? W \circledast Y^{'}\ L =^? Y \circledast W^{'}\}$ *is a unifier of the set of equations* $\{U =^? exp(V, W),\ U =^? exp(X, Y)\}$.

Proof. Let σ be a unifier for Γ. We show that $\sigma|_\Delta$, where $\Delta = \{UX, Y, V, W\}$, is a unifier for

$$\{U =^? exp(V, W),\ U =^? exp(X, Y)\}$$

First, we can note that

$$\sigma|_\Delta(U) = exp(\sigma(Z), \sigma(L)),$$
$$\sigma|_\Delta(X) = exp(\sigma(Z), \sigma(Y^{'})),$$
$$\sigma|_\Delta(V) = exp(\sigma(Z), \sigma(W^{'}))$$

and $\sigma|_\Delta(W) = \sigma(W)$ and $\sigma|_\Delta(Y) = \sigma(Y)$. Plugging into $exp(\sigma|_\Delta(X), \sigma|_\Delta(Y))$, we get

$$exp(exp(\sigma(Z), \sigma(Y^{'})), \sigma(W)) =_{\mathcal{E}_{AC}} exp(\sigma(Z), \sigma(Y^{'} \circledast \sigma(W))$$

Doing the same for $\sigma|_\Delta(exp(X, Y))$, we get

$$exp(\sigma(Z), \sigma(Y^{'} \circledast \sigma(W))$$

and the results are equivalent modulo the AC axioms for $\circledast$. □

C. Examples

To aid in the understanding of the paper we add an example that illustrates several of the key concepts presented above.

Consider the following unification problem.

$$\left\{ \begin{array}{c} U =^? exp(U_1, U_2),\ V =^? exp(V_1, V_2), \\ V_2 =^? X \circledast U_2,\ Y =^? W \circledast U,\ Y =^? W \circledast V \end{array} \right\}$$

The algorithm first collects the set $\mathcal{V} = \{U, V, U_2, V_2, W, X\}$.

The next step is to partition the set $\mathcal{V}$. Based on the initial set of equations we can rule out several of these partitions as they will immediately lead to errors. This includes any partition such that $U = U_2$ and $V = V_2$, which will lead to failure due to Lemma 4.4. If we first consider the partition

$$\Pi_1 = \{\{U\}, \{V\}, \{U_2\}, \{V_2\}, \{W\}, \{X\}\}$$

and run Step 2 none of the inference rules (a)-(i) would apply. In addition, it can be seen that the *exp*-equations,

$$\left\{ U =^? exp(U_1, U_2),\ V =^? exp(V_1, V_2) \right\}$$

are in $\{exp, *\}$-solved form

Thus the set of equations passed to the general AC-unification algorithm would be:

$$\left\{ \begin{array}{c} U =^? exp(U_1, U_2),\ V =^? exp(V_1, V_2), \\ V_2 =^? X \circledast U_2,\ Y =^? W \circledast U,\ Y =^? W \circledast V \end{array} \right\}$$

By AC-unification U and V will have to be equated. Then, as *exp* is uninterpreted in the general AC-unification algorithm, $U =^? exp(U_1, U_2)$, $V =^? exp(V_1, V_2)$ will be decomposed with $U_1 = V_1$ and $U_2 = V_2$. This results in a non-unifiable system.

This illustrates the fact that only partitions that equate U and V, with the above restrictions, need be considered.

If we choose the partition $\Pi_2 = \{\{U, V\}, \{W\}, \{U_2\}, \{V_2\}, \{X\}\}$ to begin with, then after applying step (1) we have the following set of equations

$$\left\{ \begin{array}{c} U =^? exp(U_1, U_2),\ U =^? exp(V_1, V_2), \\ V_2 =^? X \circledast U_2,\ Y =^? W \circledast U \end{array} \right\}$$

Now the algorithm moves to step (2) at which point it tries to apply the inference rules. We can see from the set of equations that the only rules applicable are rules (f) through (h), the nondeterministic rules. Therefore, the algorithm non-deterministically selects one of the rules. We can see by the example of Π_1 that selecting rule (g) will lead to failure. Thus we need only consider rules (f) and (h). We first consider rule (f). After applying rule (f) to the first two equations we get an $\{exp, *\}$-solved form of

$$\left\{ \begin{array}{c} U =^? exp(V_1, V_2),\ U_1 =^? exp(V_1, Z),\ V_2 =^? Z \circledast U_2, \\ V_2 =^? X \circledast U_2,\ Y =^? W \circledast U \end{array} \right\}$$

where Z is a new variable. By Lemma 4.16 we have that Z will never be on both the left hand side of a exp or $*$-equation and be contained in a $\circledast$-equation. Therefore, even though Z is fresh, not contained in $\mathcal{V} = \{U, V, U_2, V_2, W, X\}$ and will be equated to a pre-existing variable, X, it will not cause more application of the inference rules. This is because there cannot be an equation of the form $Z =^? exp(X_1, X_2)$, for some X_1 and X_2 as mentioned above. Thus we do not need additional applications of any of the inference rules.

Now consider the remaining non-deterministic choice, rule (h), on the set of equations:

$$\left\{ \begin{array}{c} U =^? exp(U_1, U_2),\ U =^? exp(V_1, V_2), \\ V_2 =^? X \circledast U_2,\ Y =^? W \circledast U \end{array} \right\}$$

After applying rule (h) we get the following $\{exp, *\}$-solved form:

$$\left\{ \begin{array}{c} U =^? exp(Z, L),\ V_1 =^? exp(Z, U_2^{'}),\ U_1 =^? exp(Z, V_2^{'}), \\ L =^? U_2 \circledast V_2^{'},\ L =^? V_2 \circledast U_2^{'},\ V_2 =^? X \circledast U_2,\ Y =^? W \circledast U \end{array} \right\}$$

which can be seen to be AC-unifiable. In addition, a possible AC-unifier can equate the new variables $V_2^{'}$ and $U_2^{'}$. But, by Lemma 4.16 $V_2^{'}$ and $U_2^{'}$ are not on the left hand side of any equation that would require a reapplication of the inference rules.

We now examine the remaining partitions. We can first note that in addition to the restrictions above we can also remove any partition that equates U_2 and V_2. This is due to the fact that if equated they will cause an application of rule (c). An application of rule (c) will lead to the same failure that occurred for partition Π_1. In addition we can rule out the partitions that equate X and V_2 as this will also lead to failure. The remaining partitions include:

$$\begin{aligned}
\Pi_3 &= \{\{U, V, W\}, \{U_2\}, \{V_2\}, \{X\}\}, \\
\Pi_4 &= \{\{U, V, W, X\}, \{U_2\}, \{V_2\}\}, \\
\Pi_5 &= \{\{U, V, W\}, \{U_2, X\}, \{V_2\}\}, \\
\Pi_6 &= \{\{U, V, X\}, \{W\}, \{U_2\}, \{V_2\}\}, \\
\Pi_7 &= \{\{U, V, X\}, \{U_2, W\}, \{V_2\}\}, \\
\Pi_8 &= \{\{U, V, X\}, \{U_2\}, \{V_2, W\}, \}, \\
\Pi_9 &= \{\{U, V\}, \{U_2\}, \{V_2\}, \{X, W\}\}
\end{aligned}$$

In each of these cases we get results similar to those of Π_2. That is, we will have to apply either rule (h) or rule (f) as rules (g), (b) and (c) will lead to failure. In each case the solved form presented for Π_2 is more general.