32nd International Workshop on Unification (UNIF 2018), Informal Proceedings
Oxford, 7th July, 2018

Bounded ACh Unification

Ajay Kumar Eeralla[1]*and Christopher Lynch[2]

[1] Department of Computer Science, University of Missouri
Columbia, USA, ae266@mail.missouri.edu
[2] Department of Computer Science, Clarkson University
Potsdam, USA, clynch@clarkson.edu

Abstract

We consider the problem of unification modulo an equational theory ACh, which consists of a function h which is homomorphic over an associative-commutative operator $+$. Unification modulo ACh is undecidable, so we define a bounded ACh unification problem. In this bounded version of ACh unification we essentially bound the number of times h can be recursively applied to a term, and only allow solutions that satisfy this bound. There is no bound on the number of occurrences of h in a term, and the $+$ symbol can be applied an unlimited number of times. We give inference rules for solving bounded ACh unification, and we prove that the rules are sound, complete and terminating. We have implemented the algorithm in Maude and give experimental results. We argue that this algorithm is useful in cryptographic protocol analysis.

1 Introduction

Unification is a method to find a solution for a set of equations. For instance, consider an equation $x + y \stackrel{?}{=} a + b$, where x and y are variables, and a, and b are constants. If $+$ is an uninterpreted function symbol then the equation has one solution $\{x \mapsto a, y \mapsto b\}$, and this unification is called syntactic unification. If the function symbol $+$ has the property of commutativity then the equation has two solutions: $\{x \mapsto a, y \mapsto b\}$ and $\{x \mapsto b, y \mapsto a\}$; and this is called unification modulo the commutativity theory.

Unification modulo equational theories plays a significant role in symbolic cryptographic protocol analysis [7]. An overview and references for some of the algorithms may be seen in [8, 6]. One such equational theory is the distributive axioms: $x \times (y + z) = (x \times y) + (x \times z); (y + z) \times x = (y \times x) + (z \times x)$. A decision algorithm is presented for unification modulo two-sided distributivity in [12]. A sub-problem of this, unification modulo one-sided distributivity, is in greater interest since many cryptographic protocol algorithms satisfy the one-sided distributivity. In their paper [13], Tiden and Arnborg presented an algorithm for unification modulo one-sided distributivity: $x \times (y+z) = (x \times y)+(x \times z)$, and also it has been shown that it is undecidable if we add the properties of associativity $x+(y+z) = (x+y)+z$ and a one-sided unit element $x \times 1 = x$. However, some counterexamples [11] have been presented showing that the complexity of the algorithm is exponential, although they thought it was polynomial-time bounded.

For practical purposes, one-sided distributivity can be viewed as the homomorphism theory, $h(x + y) = h(x) + h(y)$, where the unary operator h distributes over the binary operator $+$. Homomorphisms are highly used in cryptographic protocol analysis. In fact, homomorphism is a common property that many election voting protocols satisfy [9].

Our goal is to present a novel construction of an algorithm to solve unification modulo the homomorphism theory over a binary symbol $+$ that also has the properties of associativity

*Ajay K. Eeralla was partially supported by NSF CNS-1314338

and commutativity (ACh), which is an undecidable unification problem [10]. Given that ACh unification is undecidable but necessary to analyze cryptographic protocols, we developed an approximation of ACh unification, which we show to be decidable.

In this paper, we present an algorithm to solve a modified general unification problem modulo the ACh theory, which we call *bounded ACh unification*. We define the *h-height* of a term to be basically the number of h symbols recursively applied to each other. We then only search for ACh unifiers of a bounded h-height. The number of occurrences of the $+$ symbol is not bounded. In order to accomplish this we define the *h-depth* of a variable, which is the number of h symbols on top of a variable. We develop a set of inference rules for ACh unification that keep track of the h-depth of variables. If the h-depth of any variable exceeds the bound κ then the algorithm terminates with no solution. Otherwise, it gives all the unifiers or solutions to the problem.

2 Preliminaries

We assume the reader is familiar with basic notation of unification theory and term rewriting systems (see for example [3, 4]).

Definition 1 (More General Substitution)**.** A substitution σ is more general than substitution θ if there exists a substitution η such that $\theta = \sigma\eta$, denoted as $\sigma \lesssim \theta$. Note that the relation $\lesssim$ is a quasi-ordering, i.e., reflexive and transitive.

Definition 2 (Unifier, Most General Unifier)**.** A substitution σ is a unifier or solution of two terms s and t if $s\sigma = t\sigma$; it is a most general unifier if for every unifier θ of s and t, $\sigma \lesssim \theta$. Moreover, a substitution σ is a solution of set of equations if it is a solution of each of the equations. If a substitution σ is a solution of a set of equations Γ, then it is denoted by $\sigma \models \Gamma$.

A set of identities E is a subset of $\mathcal{T}(\mathcal{F}, \mathcal{V}) \times \mathcal{T}(\mathcal{F}, \mathcal{V})$ and are represented in the form $s \approx t$. An equational theory $=_E$ is induced by a set of fixed identities E and it is the least congruence relation that is closed under substitution and contains E.

Definition 3 (E-Unification Problem, E-Unifier, E-Unifiable)**.** Let $\mathcal{F}$ be a signature and E be an equational theory. An E-unification problem over $\mathcal{F}$ is a finite set of equations $\Gamma = \{s_1 \stackrel{?}{=}_E t_1, \ldots, s_n \stackrel{?}{=}_E t_n\}$ between terms. An E-unifier or E-solution of two terms s and t is a substitution σ such that $s\sigma =_E t\sigma$. An E-unifier of Γ is a substitution σ such that $s_i\sigma =_E t_i\sigma$ for $i = 1, \ldots, n$. The set of all E-unifiers is denoted by $\mathcal{U}_E(\Gamma)$ and Γ is called E-unifiable if $\mathcal{U}_E(\Gamma) \neq \emptyset$. If $E = \emptyset$ then Γ is a syntactic unification problem.

Let $\Gamma = \{s_1 \stackrel{?}{=}_E t_1, \ldots, s_n \stackrel{?}{=}_E t_n\}$ be a set of equations, and let θ be a substitution. We say that θ satisfies Γ modulo equational theory E if θ is an E-solution of each equation in Γ, that is, $s_i\theta =_E t_i\theta$ for $i = 1, \ldots, n$. We write it as $\theta \models_E \Gamma$. Let $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ and θ be substitutions, and let E be an equational theory. We say that θ satisfies σ in the equational theory E if $x_i\theta =_E t_i\theta$ for $i = 1, \ldots, n$. We write it as $\theta \models_E \sigma$.

Definition 4 (Complete Set of E-Unifiers)**.** A complete set of E-unifiers of an E-unification problem Γ is a set S of idempotent E-unifiers of Γ such that for each $\theta \in \mathcal{U}_E(\Gamma)$ in Γ there exists $\sigma \in S$ with $\sigma \lesssim_E \theta | Var(\Gamma)$, where $Var(\Gamma)$ is the set of variables in Γ.

A complete set S of E-unifiers is minimal if for two distinct unifiers σ and θ in S, one is not more general than the other; i.e., if $\sigma \lesssim_E \theta | Var(\Gamma)$ and $\sigma, \theta \in S$ then $\sigma = \theta$. A minimal

complete set of unifiers for a syntactic unification problem Γ has only one element if it is not empty. It is denoted by $mgu(\Gamma)$ and can be called most general unifier of unification problem Γ.

2.1 ACh Theory

The equational theory we consider is the theory of a homomorphism over a binary function symbol $+$. The symbol $+$ has the properties associativity and commutativity. We abbreviate this theory as ACh. The signature $\mathcal{F}$ includes a unary symbol h, and a binary symbol $+$, and other uninterpreted function symbols with fixed-arity. The function symbols h and $+$ in the signature $\mathcal{F}$ satisfy the identities: $x+(y+z) \approx (x+y)+z$ (Associativity, A for short); $x+y \approx y+x$ (Commutativity, C for short); $h(x+y) \approx h(x)+h(y)$ (Homomorphism, h for short).

2.2 h-Depth Set

For convenience, we assume that our unification problem is in *flattened* form, i.e., that every equation in the problem is in one of the following forms: $x \stackrel{?}{=} y$, $x \stackrel{?}{=} h(y)$, $x \stackrel{?}{=} y_1 + \cdots + y_n$, and $x \stackrel{?}{=} f(x_1, \ldots, x_n)$, where x, y, y_i, and x_i are variables, and f is a free symbol with $n \geq 0$. The first kind of equations are called *VarVar equations*. The second kind are called *h-equations*. The third kind are called *+-equations*. The fourth kind are called *free equations*.

Definition 5 (Graph $\mathbb{G}(\Gamma)$)**.** Let Γ be a unification problem. We define a graph $\mathbb{G}(\Gamma)$ as a graph where each node represents a variable in Γ and each edge represents a function symbol in Γ. To be exact, if an equation $w \stackrel{?}{=} f(x_1, \ldots, x_n)$, where f is a symbol with $n \geq 1$, is in Γ then the graph $\mathbb{G}(\Gamma)$ contains n edges $w \xrightarrow{f} x_1, \ldots, w \xrightarrow{f} x_n$. For a constant symbol c, if an equation $w \stackrel{?}{=} c$ is in Γ then the graph $\mathbb{G}(\Gamma)$ contains a vertex w. Finally, the graph $\mathbb{G}(\Gamma)$ contains two vertices if an equation $w \stackrel{?}{=} y$ is in Γ.

Definition 6 (h-Depth)**.** Let Γ be a unification problem and let x be a variable that occurs in Γ. Let h be a unary symbol and let f be a symbol (distinct from h) with arity greater than or equal to 1 and occur in Γ. We define h-depth of a variable x as the maximum number of h-symbols along a path to x in $\mathbb{G}(\Gamma)$, and it is denoted by $h_d(x, \Gamma)$. That is, $h_d(x, \Gamma) := \max\{h_{dh}(x, \Gamma), h_{df}(x, \Gamma), 0\}$, where $h_{dh}(x, \Gamma) := \max\{1 + h_d(y, \Gamma) \mid y \xrightarrow{h} x \text{ is an edge in } \mathbb{G}(\Gamma)\}$ and $h_{df}(x, \Gamma) := \max\{h_d(y, \Gamma) \mid \text{there exists } f \neq h \text{ such that } y \xrightarrow{f} x \text{ is in } \mathbb{G}(\Gamma)\}$.

Definition 7 (h-Height)**.** We define h-height of a term t as the following:

$$h_h(t) := \begin{cases} h_h(t') + 1 & \text{if} \quad t = h(t') \\ \max\{h_h(t_1), \ldots, h_h(t_n)\} & \text{if} \quad t = f(t_1, \ldots, t_n), f \neq h \\ 0 & \text{if} \quad t = x \text{ or } c \end{cases}$$

where f is a function symbol with arity greater than or equal to 1.

Without loss of generality, we assume that h-depth and h-height is not defined for a variable that occurs on both sides of the equation. This is because the occur check rule—concludes the problem with no solution—presented in the next section has higher priority over the h-depth updating rules.

Definition 8 (h-Depth Set)**.** Let Γ be a set of equations. The set h-depth of Γ, denoted $h_{ds}(\Gamma)$, is defined as $h_{ds}(\Gamma) := \{(x, h_d(x, \Gamma)) \mid x \text{ is a variable appearing in } \Gamma\}$. In other words, the

elements in the h-depth set are of the form $(x,\, c)$, where x is a variable that occur in Γ and c is a natural number representing the h-depth of x. Maximum value of h-depth set $\triangle$ is the maximum of all c values and it is denoted by $MaxVal(\triangle)$, i.e., $MaxVal(\triangle) := \max\{c \,|\, (x,\, c) \in \triangle\}$.

Definition 9 (Bounded E-Unification Problem, Bounded E-Unifier)**.** A κ bounded E-unification problem over $\mathcal{F}$ is a finite set of equations $\Gamma = \{s_1 \stackrel{?}{=}_E t_1, \ldots, s_n \stackrel{?}{=}_E t_n\}$, $s_i, t_i \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, where E is an equational theory, and κ is a positive integer. A κ bounded E-unifier or κ bounded E-solution of Γ is a substitution σ such that $s_i\sigma =_E t_i\sigma$, $h_h(s_i\sigma) \leq \kappa$, and $h_h(t_i\sigma) \leq \kappa$ for all i.

3 Inference System $\mathfrak{I}_h$

3.1 Problem Format

An inference system is a set of inference rules that transforms an equational unification problem into other. In our inference procedure, we use a set triple $\Gamma||\triangle||\sigma$, where Γ is a unification problem modulo the ACh theory, $\triangle$ is an h-depth set, and σ is a substitution. Let $\kappa \in \mathbb{N}$ be a bound on the h-depth of the variables. A substitution θ satisfies the set triple $\Gamma||\triangle||\sigma$ if θ satisfies every equation in Γ and σ, $MaxVal(\triangle) \leq \kappa$, and we write that relation as $\theta \models \Gamma||\triangle||\sigma$. We also use a special set triple $\bot$ for no solution in the inference procedure. Generally, the inference procedure is based on priority of rules and also uses don't care determinism when there is no priority. i.e., any one rule applied from a set of rules without priority. Initially, Γ is the non-empty set of equations to solve and the substitution σ is the identity substitution. The inference rules are applied until either the set of equations is empty with most general unifier σ or $\bot$ for no solution. Of course, the substitution σ is a κ bounded E-unifier of Γ.

An inference rule is written in the following form $\frac{\Gamma||\triangle||\sigma}{\Gamma'||\triangle'||\sigma'}$. This means that if something matches the top of this rule, then it is to be replaced with the bottom of the rule. In the proofs we will write inference rules as follows: $\Gamma||\triangle||\sigma \Rightarrow_{\mathfrak{I}_h} \{\Gamma_1||\triangle_1||\sigma_1, \cdots, \Gamma_n||\triangle_n|\sigma_n\}$ meaning to branch and replace the left hand side with one of the right hand sides in each branch. The only inference rule that has more than one branch is *AC Unification*. So we often just write inference rules as follows: $\Gamma||\triangle||\sigma \Rightarrow_{\mathfrak{I}_h} \Gamma'||\triangle'||\sigma'$. Let $\mathcal{OV}$ be the set of variables occurring in the unification problem Γ and let $\mathcal{NV}$ be a new set of variables such that $\mathcal{NV} = \mathcal{V} \setminus \mathcal{OV}$. Unless otherwise stated we assume that x, $x_1, \ldots, x_n$, and y, $y_1, \ldots, y_n$, z are variables in $\mathcal{V}$, v, $v_1, \ldots, v_n$ are in $\mathcal{NV}$, and terms t, $t_1, \ldots, t_n$, s, $s_1, \ldots, s_n$ in $\mathcal{T}(\mathcal{F}, \mathcal{V})$, and f and g are uninterpreted function symbols. Recall that h is a unary, and the associativity and the commutativity operator $+$. A fresh variable is a variable that is generated by the current inference rule and has never been used before.

For convenience, we assume that every equation in the problem is in one of the following flattened forms: $x \stackrel{?}{=} y$, $x \stackrel{?}{=} h(y)$, and $z \stackrel{?}{=} y + x$, where x, y, and z are variables. If not, we apply flattening rules to put the equations into that form. These rules are performed before any other inference rule. They put the problem into flattened form and all the other inference rules leave the problem in flattened form, so there is no need to perform these rules again later. It is necessary to update the h-depth set $\triangle$ with the h-depth values for each variable during the inference procedure.

3.2 Inference Rules

Flattening. We present a set of inference rules for flattening. The variable v represents a fresh variable in the following rules.

Flatten Both Sides

$$\frac{\{t_1 \stackrel{?}{=} t_2\} \cup \Gamma || \triangle || \sigma}{\{v \stackrel{?}{=} t_1,\, v \stackrel{?}{=} t_2\} \cup \Gamma || \{(v,\, 0)\} \cup \triangle || \sigma} \text{ if } t_1 \text{ and } t_2 \notin \mathcal{V}$$

Flatten Left $+$

$$\frac{\{t \stackrel{?}{=} t_1 + t_2\} \cup \Gamma || \triangle || \sigma}{\{t \stackrel{?}{=} v + t_2,\, v \stackrel{?}{=} t_1\} \cup \Gamma || \{(v,\, 0)\} \cup \triangle || \sigma} \qquad \text{if } t_1 \notin \mathcal{V}$$

Flatten Right $+$

$$\frac{\{t \stackrel{?}{=} t_1 + t_2\} \cup \Gamma || \triangle || \sigma}{\{t \stackrel{?}{=} t_1 + v,\, v \stackrel{?}{=} t_2\} \cup \Gamma || \{(v,\, 0)\} \cup \triangle || \sigma} \qquad \text{if } t_2 \notin \mathcal{V}$$

Flatten Under h

$$\frac{\{t_1 \stackrel{?}{=} h(t)\} \cup \Gamma || \triangle || \sigma}{\{t_1 \stackrel{?}{=} h(v),\, v \stackrel{?}{=} t\} \cup \Gamma || \{(v,\, 0)\} \cup \triangle || \sigma} \qquad \text{if } t \notin \mathcal{V}$$

Update h-Depth Set. We also present a set of inference rules to update the h-depth set. We apply these rules immediately after applying any other rule in the inference system.
Update h

$$\frac{\{x \stackrel{?}{=} h(y)\} \cup \Gamma || \{(x,\, c_1),\, (y,\, c_2)\} \cup \triangle || \sigma}{\{x \stackrel{?}{=} h(y)\} \cup \Gamma || \{(x,\, c_1),\, (y,\, c_1 + 1)\} \cup \triangle || \sigma} \qquad \text{If } c_2 < (c_1 + 1)$$

Update $+$

1. $$\frac{\{x_1 \stackrel{?}{=} y_1 + y_2\} \cup \Gamma || \{(x_1,\, c_1),\, (y_1,\, c_2),\, (y_2,\, c_3)\} \cup \triangle || \sigma}{\{x_1 \stackrel{?}{=} y_1 + y_2\} \cup \Gamma || \{(x_1,\, c_1),\, (y_1,\, c_1),\, (y_2,\, c_3)\} \cup \triangle || \sigma} \quad \text{If } c_2 < c_1$$
2. $$\frac{\{x_1 \stackrel{?}{=} y_1 + y_2\} \cup \Gamma || \{(x_1,\, c_1),\, (y_1,\, c_2),\, (y_2,\, c_3)\} \cup \triangle || \sigma}{\{x_1 \stackrel{?}{=} y_1 + y_2\} \cup \Gamma || \{(x_1,\, c_1),\, (y_1,\, c_2),\, (y_2,\, c_1)\} \cup \triangle || \sigma} \quad \text{If } c_3 < c_1$$

Splitting Rule. This rule takes the homomorphism theory into account. In this theory, we can not solve equation $h(y) \stackrel{?}{=} x_1 + x_2$ unless y can be written as the sum of two new variables $y = v_1 + v_2$, where v_1 and v_2 are in $\mathcal{NV}$. Without loss of generality we generalize it to n variables $x_1, \ldots, x_n$.

$$\frac{\{w \stackrel{?}{=} h(y), w \stackrel{?}{=} x_1 + \cdots + x_n\} \cup \Gamma || \triangle || \sigma}{\{w \stackrel{?}{=} h(y), y \stackrel{?}{=} v_1 + \cdots + v_n, x_1 \stackrel{?}{=} h(v_1), \ldots, x_n \stackrel{?}{=} h(v_n)\} \cup \Gamma || \triangle' || \sigma}$$

where $n > 1$, $y \neq w$, $\triangle' = \{(v_1,\, 0), \ldots, (v_n,\, 0)\} \cup \triangle\}$, and $v_1, \ldots, v_n$ are fresh variables in $\mathcal{NV}$.
Trivial. The Trivial inference rule is to remove trivial equations in the given problem Γ.

$$\frac{\{t \stackrel{?}{=} t\} \cup \Gamma || \triangle || \sigma}{\Gamma || \triangle || \sigma}$$

Variable Elimination (VE). The Variable Elimination rule is to convert the equations into assignments. In other words, it is used to find the most general unifier. The rule VE-2 is performed last after all other inference rules have been performed. The rule VE-1 is performed eagerly.

1.
$$\frac{\{x \stackrel{?}{=} y\} \cup \Gamma || \triangle || \sigma}{\Gamma\{x \mapsto y\} || \triangle || \sigma\{x \mapsto y\} \cup \{x \mapsto y\}}$$

2.
$$\frac{\{x \stackrel{?}{=} t\} \cup \Gamma || \triangle || \sigma}{\Gamma || \triangle || \sigma\{x \mapsto t\} \cup \{x \mapsto t\}} \qquad \text{if } t \notin \mathcal{V} \text{ and } x \text{ does not occur in } t$$

Decomposition. The Decomposition rule decomposes an equation into several sub-equations if both sides top symbol matches.

$$\frac{\{x \stackrel{?}{=} f(s_1, \ldots, s_n), x \stackrel{?}{=} f(t_1, \ldots, t_n)\} \cup \Gamma || \triangle || \sigma}{\{x \stackrel{?}{=} f(t_1, \ldots, t_n),\ s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\} \cup \Gamma || \triangle || \sigma} \qquad \text{if } f \neq +$$

AC Unification. The AC Unification rule calls an AC unification algorithm to unify the AC part of the problem. Notice that we apply AC unification only once when no other rule except VE-2 can apply. In this inference rule Ψ represents the set of all equations with the + symbol on the right hand side. Γ represents the set of equations not containing a + symbol. *Unify* is a function that returns one of the complete set of unifiers returned by the AC unification algorithm. *GetEqs* is a function that takes a substitution and returns the equational form of that substitution. In other words, $GetEqs([x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]) = \{x_1 \stackrel{?}{=} t_1, \ldots, x_n \stackrel{?}{=} t_n\}$.

$$\frac{\Psi \cup \Gamma || \triangle || \sigma}{GetEqs(Unify\ \Psi) \cup \Gamma || \triangle || \sigma}$$

Note that we have written the rule for one member of the complete set of AC unifiers of Ψ. This will branch on every member of the complete set of AC unifiers of Ψ.

Occur Check. It is to determine if a variable on the left hand side of an equation occurs on the other side of the equation. If it does, then there is no solution to the unification problem. This rule has the highest priority.

$$\frac{\{x \stackrel{?}{=} f(t_1, \ldots, t_n)\} \cup \Gamma || \triangle || \sigma}{\bot} \qquad \text{If } x \in \mathcal{V}ar(f(t_1, \ldots, t_n)\sigma)$$

where $\mathcal{V}ar(f(t_1, \ldots, t_n)\sigma)$ represents set of all variables that occur in $f(t_1, \ldots, t_n)\sigma$.

Clash. This rule checks if the *top symbol* on both sides of an equation is the same. If not, then there is no solution to the problem, unless one of them is h and the other $+$.

$$\frac{\{x \stackrel{?}{=} f(s_1, \ldots, s_m),\ x \stackrel{?}{=} g(t_1, \ldots, t_n)\} \cup \Gamma || \triangle || \sigma}{\bot} \qquad \text{If } f \notin \{h, +\} \text{ or } g \notin \{h, +\}$$

Bound Check. The Bound Check is to determine if a solution exists within the bound κ, a given maximum h-depth of any variable in Γ. If one of the h-depths in the h-depth set $\triangle$ exceeds the bound κ, then the problem has no solution.

$$\frac{\Gamma || \triangle || \sigma}{\bot} \qquad \text{If } MaxVal(\triangle) > \kappa$$

Unification Problem	Real Time	Solution	# Sol.	Bound
$\{h(y) \stackrel{?}{=} y + x\}$	674ms	$\perp$	0	10
$\{h(y) \stackrel{?}{=} y + x\}$	15880ms	$\perp$	0	20
$\{h(y) \stackrel{?}{=} x_1 + x_2\}$	5ms	Yes	1	10
$\{h(h(x)) \stackrel{?}{=} h(h(y))\}$	2ms	Yes	1	10
$\{x + y_1 \stackrel{?}{=} x + y_2\}$	3ms	Yes	1	10
$\{v \stackrel{?}{=} x + y, v \stackrel{?}{=} w + z, s \stackrel{?}{=} h(t)\}$	46ms	Yes	10	10
$\{v \stackrel{?}{=} x_1 + x_2, v \stackrel{?}{=} x_3 + x_4, x_1 \stackrel{?}{=} h(y), x_2 \stackrel{?}{=} h(y)\}$	100ms	Yes	6	10
$\{h(h(x)) \stackrel{?}{=} v + w + y + z\}$	224ms	Yes	1	10
$\{v \stackrel{?}{=} (h(x) + y), v \stackrel{?}{=} w + z\}$	55ms	Yes	7	10
$\{f(x, y) \stackrel{?}{=} h(x_1)\}$	0ms	$\perp$	0	10
$\{f(x_1, y_1) \stackrel{?}{=} f(x_2, y_2)\}$	1ms	Yes	1	10
$\{v \stackrel{?}{=} x_1 + x_2, v \stackrel{?}{=} x_3 + x_4\}$	17ms	Yes	7	10
$\{f(x_1, y_1) \stackrel{?}{=} g(x_2, y_2)\}$	0ms	$\perp$	0	10
$\{h(y) \stackrel{?}{=} x, y \stackrel{?}{=} h(x)\}$	0ms	$\perp$	0	10

Table 1: Tested results with bounded ACh-unification algorithm

4 Proof of Correctness

The proposed inference process eventually halts.

Lemma 1. There is no infinite sequence of inference rules.

Our inference system is truth-preserving.

Lemma 2 (Soundness). Let $\Gamma||\triangle||\sigma \Rightarrow_{\mathfrak{I}_h} \{\Gamma_1||\triangle_1||\sigma_1, \cdots, \Gamma_n||\triangle_n|\sigma_n\}$ be an inference rule. Let θ be a substitution such that $\theta \models \Gamma_i||\triangle_i||\sigma_i$. Then $\theta \models \Gamma||\triangle||\sigma$.

Of course, our inference system never loses any solution.

Lemma 3 (Completeness). Let $\Gamma||\triangle||\sigma$ be a set triple. Let $\Gamma||\triangle||\sigma \Rightarrow_{\mathfrak{I}_h} \{\Gamma_1||\triangle_1||\sigma_1, \cdots, \Gamma_n||\triangle_n|\sigma_n\}$ be an inference rule. If $\theta \models \Gamma||\triangle||\sigma$, then there exists an i and a θ', whose domain is the variables in $Var(\Gamma_i) \setminus Var(\Gamma)$, such that $\theta\theta' \models \Gamma_i||\triangle_i||\sigma_i$.

5 Implementation

We have implemented the algorithm in Maude [5]. We chose the Maude language because the inference rules are very similar to the rules of Maude and an implementation will be integrated into the Maude-NPA tool at some time. The Maude-NPA tool is written in Maude. The system specifications are Ubuntu 14.04 LTS, Intel Core i5 3.20 GHz, and 8 GiB RAM with Maude 2.6.

We give a table to show some of our results. In the given table, we use five columns: Unification problem, Real Time, time to terminate the program in ms (milli seconds), Solution either $\perp$ for no solution or Yes for solutions, # Sol. for number of solutions, and Bound κ. It makes sense that the real time keeps increasing as the given h-depth κ increases for the first problem where the other problems give solutions, but in either case the program terminates.

6 Conclusion

We introduced a set of inference rules to solve the unification problem modulo the homomorphism theory h over an AC symbol $+$, by enforcing bound k on the h-depth of any variable.

Homomorphism is a property which is very common in cryptographic algorithms. So, it is important to analyze cryptographic protocols in the homomorphism theory. Some of the algorithms and details in this direction can be seen in [2, 6, 1]. However, none of those results perform ACh unification because that is undecidable. One way around this is to assume that an identity and an inverse exist, but because of the way the Maude-NPA works it would still be necessary to unify modulo ACh. So an unification algorithm there becomes crucial. We believe that our approximation is a good way to deal with it. We also tested some problems and the results are shown in Table 1.

References

[1] S. Anantharaman, H. Lin, C. Lynch, P. Narendran, and M. Rusinowitch. Cap unification: application to protocol security modulo homomorphic encryption. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 192–203. ACM, 2010.

[2] S. Anantharaman, H. Lin, C. Lynch, P. Narendran, and M. Rusinowitch. Unification Modulo Homomorphic Encryption. In *booktitle of Automated Reasoning*, pages 135–158. Springer, 2012.

[3] F. Baader and T. Nipkow. *Term Rewriting and All that.* Cambridge University Press, 1998.

[4] F. Baader and W. Snyder. Unification Theory. In *Handbook of Automated Reasoning*, pages 447–533. Elsevier, 2001.

[5] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and Carolyn L. Talcott. *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic.* Springer, 2007.

[6] S. Escobar, D. Kapur, C. Lynch, C. Meadows, J. Meseguer, P. Narendran, and R. Sasse. Protocol Analysis in Maude-NPA Using Unification Modulo Homomorphic Encryption. In *Proceedings of the 13th International ACM SIGPLAN Symposium on Principles and Practices of Declarative Programming*, pages 65–76. ACM, 2011.

[7] S. Escobar, C. Meadows, and J. Meseguer. Maude-Npa: cryptographic protocol analysis modulo equational properties. In *Foundations of Security Analysis and Design V: FOSAD 2007/2008/2009 Tutorial Lectures*, pages 1–50. Springer, 2007.

[8] D. Kapur, P. Narendran, and L. Wang. An E-unification Algorithm for Analyzing Protocols That Use Modular Exponentiation. In *Rewriting Techniques and Applications*, pages 165–179. Springer, 2003.

[9] S. Kremer, M. Ryan, and B. Smyth. Election Verifiability in Electronic Voting Protocols. In *Computer Security – ESORICS*, pages 389–404. Springer, 2010.

[10] P. Narendran. Solving Linear Equations over Polynomial Semirings. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 466–472. IEEE Computer Society, 1996.

[11] P. Narendran, Andrew M. Marshall, and B. Mahapatra. On the Complexity of the Tiden-Arnborg Algorithm for Unification modulo One-Sided Distributivity. In *Proceedings 24th International Workshop on Unification*, pages 54–63. Open Publishing Association, 2010.

[12] M. Schmidt-Schauß. A Decision Algorithm for Distributive Unification. In *Theoretical Computer Science*, pages 111–148. Elsevier, 1998.

[13] E. Tidén and Stefan Arnborg. Unification Problems with One-Sided Distributivity. In *booktitle of Symbolic Computation*, pages 183–202. Springer, 1987.