

Bounded ACh Unification

AJAY KUMAR EERALLA^{1†} and CHRISTOPHER LYNCH²

¹ *Department of Electrical Engineering and Computer Science, University of Missouri, Columbia, USA.*

² *Department of Computer Science, Clarkson University, Potsdam, USA*

Received 27 August 2019

We consider the problem of the unification modulo an equational theory ACh, which consists of a function symbol h that is homomorphic over an associative-commutative operator $+$. Since the unification modulo ACh theory is undecidable, we define a variant of the problem called *bounded ACh unification*. In this bounded version of ACh unification, we essentially bound the number of times h can be applied to a term recursively, 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 the bounded version of the problem and 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 play a significant role in symbolic cryptographic protocol analysis (Escobar *et al.* 2007). An overview and references for some of the algorithms may be seen in (Kapur *et al.* 2003; Escobar *et al.* 2011; Narendran *et al.* 2015). 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 (Schmidt-Schauß *et al.* 1998). 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 (Tiden and Arnborg 1987),

[†] Ajay Kumar Eeralla was partially supported by NSF Grant CNS 1314338

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 counter examples (Narendran *et al.* 2015) 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 (Kremer *et al.* 2010).

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 and commutativity (ACh), which is an undecidable unification problem (Narendran *et al.* 1996). 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. We do not restrict the h-height of terms in unification problems. Moreover, the number of occurrences of the $+$ symbol is bounded neither in a problem nor in its solutions. 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. Preliminary

2.1. Basic Notation

We briefly recall the standard notation of unification theory and term rewriting systems from (Baader and Nipkow 1998; Baader and Snyder 2001).

Given a finite or countably infinite set of function symbols \mathcal{F} , also known as a signature, and a countable set of variables \mathcal{V} , the set of \mathcal{F} -terms over \mathcal{V} is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$. The set of variables appearing in a term t is denoted by $Var(t)$, and it is extended to sets of equations. A term is called ground if $Var(t) = \emptyset$. Let $Pos(t)$ be the set of positions of a term t including the root position ϵ (Baader and Snyder 2001). For any $p \in Pos(t)$, $t|_p$ is the subterm of t at the position p and $t[s]_p$ is the term t in which $t|_p$ is replaced by s . A substitution is a mapping from \mathcal{V} to $\mathcal{T}(\mathcal{F}, \mathcal{V})$ with only finitely many variables

not mapped to themselves and is denoted by $\sigma = \{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$, where the domain of σ is $Dom(\sigma) := \{x_1, \dots, x_n\}$. The range of σ , denoted as $Range(\sigma)$, defined as union of the sets $\{x\sigma\}$, where x is a variable in $Dom(\sigma)$. The identity substitution is a substitution that maps all the variables to themselves. The application of substitution σ to a term t , denoted as $t\sigma$, is defined by induction on the structure of the terms:

- $x\sigma$, where t is a variable x
- c , where t is a constant symbol c
- $f(t_1\sigma, \dots, t_n\sigma)$, where $t = f(t_1, \dots, t_n)$ with $n \geq 1$

The restriction of a substitution σ to a set variables \mathcal{V} , denoted as $\sigma|_{\mathcal{V}}$, is the substitution which is equal to identity everywhere except over $\mathcal{V} \cap Dom(\sigma)$, where it coincides with σ .

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 a 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 $\Gamma = \{s_1 \stackrel{?}{=}_E t_1, \dots, s_n \stackrel{?}{=}_E t_n\}$ 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 $i = 1, \dots, n$. 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 $\Gamma = \{s_1 \stackrel{?}{=}_E t_1, \dots, s_n \stackrel{?}{=}_E t_n\}$ be a set of equations, and let θ be a substitution. We write $\theta \models_E \Gamma$ when θ is an E -unifier of Γ . Let $\sigma = \{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$ 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 $i = 1, \dots, n$. We write it as $\theta \models_E \sigma$.

Definition 4. Let E be an equational theory and \mathcal{X} be a set of variables. The substitution σ is *more general modulo E on \mathcal{X}* than θ iff there exists a substitution σ' such that $x\theta =_E x\sigma\sigma'$ for all $x \in \mathcal{X}$. We write it as $\sigma \lesssim_{\mathcal{X}}^E \theta$.

Definition 5 (Complete Set of E -Unifiers). Let Γ be a E -unification problem over \mathcal{F} and let $Var(\Gamma)$ be the set of all variables occurring in Γ . A complete set of E -unifiers of Γ is a set S of substitutions such that, each element of S is a E -unifier of Γ , i.e., $S \subseteq \mathcal{U}_E(\Gamma)$, and for each $\theta \in \mathcal{U}_E(\Gamma)$ there exists a $\sigma \in S$ such that σ is more general modulo E on $Var(\Gamma)$ than θ , i.e., $\sigma \lesssim_{Var(\Gamma)}^E \theta$.

A complete set S of E -unifiers is minimal if for any two distinct unifiers σ and θ in

S , one is not more general modulo E than the other, i.e., $\sigma \lesssim_E^{Var(\Gamma)} \theta$ implies $\sigma = \theta$. 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 .

Definition 6. Let E be an equational theory. We say that a multi-set of equations Γ' is a conservative E -extension of another multi-set of equations Γ if any solution of Γ' is also a solution of Γ and any solution of Γ can be extended to a solution of Γ' . This means for any solution σ of Γ , there exists θ whose domain is the variables in $Var(\Gamma') \setminus Var(\Gamma)$ such that $\sigma\theta$ is a solution of Γ . The property of conservative E -extension is transitive.

Let \mathcal{F} be a signature, and l, r be \mathcal{F} -terms. A *rewrite rule* is an identity, denoted as $l \rightarrow r$, where l is not a variable and $Var(r) \subseteq Var(l)$. A *term rewriting system* (TRS) is a pair (\mathcal{F}, R) , where R is a finite set of rewrite rules. In general, a TRS is represented by R . A term u rewrites to a term v with respect to R , denoted by $u \rightarrow_R v$ (or simply $u \rightarrow v$), if there exist a position p of u , $l \rightarrow r \in R$, and substitution σ such that $u|_p = l\sigma$ and $v = u[r\sigma]_p$. A TRS R is said to be *terminating* if there is no infinite reduction sequences of the form $u_0 \rightarrow_R u_1 \rightarrow_R \dots$. A TRS R is *confluent* if, whenever $u \rightarrow_R^* s_1$ and $u \rightarrow_R^* s_2$, there exists a term v such that $s_1 \rightarrow_R^* v$ and $s_2 \rightarrow_R^* v$. A TRS R is convergent if it is both confluent and terminating.

2.2. ACh Theory

The equational theory we consider is the theory of a homomorphism over a binary function symbol $+$ which satisfies the properties of associativity and the 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 following 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)

2.3. Rewriting Systems

We consider two convergent rewriting systems R_1 and R_2 for homomorphism h modulo associativity and commutativity.

- $R_1 := \{h(x_1 + x_2) \rightarrow h(x_1) + h(x_2)\}$ and
- $R_2 := \{h(x_1) + h(x_2) \rightarrow h(x_1 + x_2)\}.$

2.4. 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: $x \stackrel{?}{=} y$, $x \stackrel{?}{=} h(y)$, $x \stackrel{?}{=}$

$y_1 + \dots + y_n$, and $x \stackrel{?}{=} f(x_1, \dots, x_n)$, where x and y are variables, y_i s and x_i s are pairwise distinct 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 7 (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 $y \stackrel{?}{=} f(x_1, \dots, x_n)$, where f is a symbol with $n \geq 1$, is in Γ then the graph $\mathbb{G}(\Gamma)$ contains n edges $y \xrightarrow{f} x_1, \dots, y \xrightarrow{f} x_n$. For a constant symbol c , if an equation $y \stackrel{?}{=} c$ is in Γ then the graph $\mathbb{G}(\Gamma)$ contains a vertex y . Finally, the graph $\mathbb{G}(\Gamma)$ contains two vertices y and x if an equation $y \stackrel{?}{=} x$ is in Γ .

Definition 8 (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 occurring 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 9 (h-Height). We define h-height of a term t as the following:

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

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

Definition 10 (h-Depth Set). Let Γ be a set of equations. The h-depth set 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 Δ is the maximum of all c values and it is denoted by $MaxVal(\Delta)$, i.e., $MaxVal(\Delta) := \max\{c \mid (x, c) \in \Delta \text{ for some } x\}$.

Definition 11 (ACh-Unification Problem, Bounded ACh-Unifier). An ACh-unification problem over \mathcal{F} is a finite set of equations $\Gamma = \{s_1 \stackrel{?}{=}_{ACh} t_1, \dots, s_n \stackrel{?}{=}_{ACh} t_n\}$, $s_i, t_i \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, where ACh is the equational theory defined above. A κ bounded ACh-unifier or κ bounded ACh-solution of Γ is a substitution σ such that $s_i\sigma =_{ACh} t_i\sigma$, $h_h(s_i\sigma) \leq \kappa$, and $h_h(t_i\sigma) \leq \kappa$ for all i .

Notice that the bound κ has no role in the problem but in the solution.

3. Inference System \mathfrak{I}_{ACh}

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 $\Gamma||\Delta||\sigma$ similar to the format presented in (Liu and Lynch 2011), where Γ is a unification problem modulo the ACh theory, Δ is an h-depth set of Γ , 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 $\Gamma||\Delta||\sigma$ if θ satisfies σ and every equation in Γ , $MaxVal(\Delta) \leq \kappa$, and we write that relation as $\theta \models \Gamma||\Delta||\sigma$. We also use a special set triple \perp for no solution in the inference procedure. Generally, the inference procedure is based on the priority of rules and also uses *don't care* non-determinism when there is no priority. i.e., any rule applied from a set of rules without priority. Initially, Γ is the non-empty set of equations to solve, Δ is an empty set, and σ is the identity substitution. The inference rules are applied until either the set of equations is empty with most general unifier σ or \perp for no solution. Of course, the substitution σ is a κ bounded E -unifier of Γ . An inference rule is written as $\frac{\Gamma||\Delta||\sigma}{\Gamma'||\Delta'||\sigma'}$. This means that if something matches the top of this rule, then it is to be replaced with the bottom of the rule.

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, \dots, x_n , and y, y_1, \dots, y_n, z are variables in \mathcal{V} , v, v_1, \dots, v_n are in \mathcal{NV} , and terms $w, t, t_1, \dots, t_n, s, s_1, \dots, s_n$ in $\mathcal{T}(\mathcal{F}, \mathcal{V})$, and f and g are uninterpreted function symbols. 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 flattened forms (see Section 2.4). 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 Δ with the h-depth values for each variable during the inference procedure.

3.2. Inference Rules

We present a set of inference rules to solve a unification problem modulo associativity, commutativity, and homomorphism theory. We also present some examples that illustrate the applicability of these rules.

3.2.1. Flattening

Firstly, we present a set of inference rules for flattening the given set of equations. The variable v represents a fresh variable in the following rules.

Flatten Both Sides (FBS) $\frac{\{t_1 \stackrel{?}{=} t_2\} \cup \Gamma \Delta \sigma}{\{v \stackrel{?}{=} t_1, v \stackrel{?}{=} t_2\} \cup \Gamma \{(v, 0)\} \cup \Delta \sigma} \quad \text{if } t_1 \text{ and } t_2 \notin \mathcal{V}$
Flatten Left + (FL) $\frac{\{t \stackrel{?}{=} t_1 + t_2\} \cup \Gamma \Delta \sigma}{\{t \stackrel{?}{=} v + t_2, v \stackrel{?}{=} t_1\} \cup \Gamma \{(v, 0)\} \cup \Delta \sigma} \quad \text{if } t_1 \notin \mathcal{V}$
Flatten Right + (FR) $\frac{\{t \stackrel{?}{=} t_1 + t_2\} \cup \Gamma \Delta \sigma}{\{t \stackrel{?}{=} t_1 + v, v \stackrel{?}{=} t_2\} \cup \Gamma \{(v, 0)\} \cup \Delta \sigma} \quad \text{if } t_2 \notin \mathcal{V}$
Flatten Under h (FU) $\frac{\{t_1 \stackrel{?}{=} h(t)\} \cup \Gamma \Delta \sigma}{\{t_1 \stackrel{?}{=} h(v), v \stackrel{?}{=} t\} \cup \Gamma \{(v, 0)\} \cup \Delta \sigma} \quad \text{if } t \notin \mathcal{V}$

We demonstrate the applicability of these rules using the example below.

Example 1. Solve the unification problem $\{h(h(x)) \stackrel{?}{=} (s + w) + (y + z)\}$.

We only consider the set of equations Γ here, not the full triple.

$$\begin{aligned}
& \{h(h(x)) \stackrel{?}{=} (s + w) + (y + z)\} \xRightarrow{FBS} \\
& \{v \stackrel{?}{=} h(h(x)), v \stackrel{?}{=} (s + w) + (y + z)\} \xRightarrow{FL} \\
& \{v \stackrel{?}{=} h(h(x)), v \stackrel{?}{=} v_1 + (y + z), v_1 \stackrel{?}{=} s + w\} \xRightarrow{FL} \\
& \{v \stackrel{?}{=} h(h(x)), v \stackrel{?}{=} v_1 + (y + z), v_1 \stackrel{?}{=} v_2 + w, v_2 \stackrel{?}{=} s\} \xRightarrow{FR} \\
& \{v \stackrel{?}{=} h(h(x)), v \stackrel{?}{=} v_1 + v_3, v_1 \stackrel{?}{=} v_2 + w, v_3 \stackrel{?}{=} y + z, v_2 \stackrel{?}{=} s\} \xRightarrow{FR} \\
& \{v \stackrel{?}{=} h(h(x)), v \stackrel{?}{=} v_1 + v_3, v_1 \stackrel{?}{=} v_2 + v_4, v_3 \stackrel{?}{=} y + z, v_2 \stackrel{?}{=} s, v_4 \stackrel{?}{=} w\} \xRightarrow{FU} \\
& \{v \stackrel{?}{=} h(v_5), v \stackrel{?}{=} v_1 + v_3, v_1 \stackrel{?}{=} v_2 + v_4, v_3 \stackrel{?}{=} y + z, v_2 \stackrel{?}{=} s, v_4 \stackrel{?}{=} w, v_5 \stackrel{?}{=} h(x)\}.
\end{aligned}$$

We see that each equation in the set $\{v \stackrel{?}{=} h(v_5), v \stackrel{?}{=} v_1 + v_3, v_1 \stackrel{?}{=} v_2 + v_4, v_3 \stackrel{?}{=} y + z, v_2 \stackrel{?}{=} s, v_4 \stackrel{?}{=} w, v_5 \stackrel{?}{=} h(x)\}$ is in the flattened form.

3.2.2. Update h -Depth Set

We also present a set of inference rules to update the h -depth set. These rules are performed eagerly.

Update h (Uh)

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

Example 2. Solve the unification problem: $\{x \stackrel{?}{=} h(h(h(y)))\}$.

We only consider the pair $\Gamma || \Delta$ since σ does not change at this step.

$$\begin{aligned} & \{x \stackrel{?}{=} h(h(h(y)))\} || \{(x, 0), (y, 0)\} \xRightarrow{FU^+} \\ & \{x \stackrel{?}{=} h(v), v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} h(y)\} || \{(x, 0), (y, 0), (v, 0), (v_1, 0)\} \xRightarrow{Uh} \\ & \{x \stackrel{?}{=} h(v), v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} h(y)\} || \{(x, 0), (y, 0), (v, 1), (v_1, 0)\} \xRightarrow{Uh} \\ & \{x \stackrel{?}{=} h(v), v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} h(y)\} || \{(x, 0), (y, 0), (v, 1), (v_1, 1)\} \xRightarrow{Uh} \\ & \{x \stackrel{?}{=} h(v), v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} h(y)\} || \{(x, 0), (y, 2), (v, 1), (v_1, 1)\} \xRightarrow{Uh} \\ & \{x \stackrel{?}{=} h(v), v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} h(y)\} || \{(x, 0), (y, 2), (v, 1), (v_1, 2)\} \xRightarrow{Uh} \\ & \{x \stackrel{?}{=} h(v), v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} h(y)\} || \{(x, 0), (y, 3), (v, 1), (v_1, 2)\}, \end{aligned}$$

where $\xRightarrow{FU^+}$ represents the application of FU rule once or more than once.

It is true that the h-Depth of y is 3 since there are three edges labeled h from x to y , in the graph $\mathbb{G}(\Gamma)$.

Update +**1 Update Left + (UL)**

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

2 Update Right + (UR)

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

Example 3. Solve the unification problem $\{z \stackrel{?}{=} x + y, x_1 \stackrel{?}{=} h(h(z))\}$.

Similar to the last example, we only consider the pair $\Gamma || \Delta$,

$$\begin{aligned} & \{z \stackrel{?}{=} x + y, x_1 \stackrel{?}{=} h(h(z))\} || \{(x, 0), (y, 0), (z, 0), (x_1, 0)\} \xRightarrow{FU} \\ & \{z \stackrel{?}{=} x + y, x_1 \stackrel{?}{=} h(v), v \stackrel{?}{=} h(z)\} || \{(x, 0), (y, 0), (z, 0), (x_1, 0), (v, 0)\} \xRightarrow{Uh^+} \\ & \{z \stackrel{?}{=} x + y, x_1 \stackrel{?}{=} h(v), v \stackrel{?}{=} h(z)\} || \{(x, 0), (y, 0), (z, 2), (x_1, 0), (v, 1)\} \xRightarrow{UL} \\ & \{z \stackrel{?}{=} x + y, x_1 \stackrel{?}{=} h(v), v \stackrel{?}{=} h(z)\} || \{(x, 2), (y, 0), (z, 2), (x_1, 0), (v, 1)\} \xRightarrow{UR} \\ & \{z \stackrel{?}{=} x + y, x_1 \stackrel{?}{=} h(v), v \stackrel{?}{=} h(z)\} || \{(x, 2), (y, 2), (z, 2), (x_1, 0), (v, 1)\}. \end{aligned}$$

Since there are two edges labeled h from x_1 to z in the graph $\mathbb{G}(\Gamma)$, the h-Depth of z is 2. The h-Depths of x and y are also updated accordingly.

Now, we resume the inference procedure for Example 1 and also we consider \triangle because it will be updated at this step.

$$\begin{aligned}
& \{v \stackrel{?}{=} h(v_3), v_3 \stackrel{?}{=} h(x), v \stackrel{?}{=} v_1 + v_2, v_1 \stackrel{?}{=} s + w, v_2 \stackrel{?}{=} y + z\} \\
& \{(x, 0), (y, 0), (z, 0), (s, 0), (w, 0), (v, 0), (v_1, 0), (v_2, 0), (v_3, 0)\} \xRightarrow{U_h} \\
& \{v \stackrel{?}{=} h(v_3), v_3 \stackrel{?}{=} h(x), v \stackrel{?}{=} v_1 + v_2, v_1 \stackrel{?}{=} s + w, v_2 \stackrel{?}{=} y + z\} \\
& \{(x, 1), (y, 0), (z, 0), (s, 0), (w, 0), (v, 0), (v_1, 0), (v_2, 0), (v_3, 0)\} \xRightarrow{U_h} \\
& \{v \stackrel{?}{=} h(v_3), v_3 \stackrel{?}{=} h(x), v \stackrel{?}{=} v_1 + v_2, v_1 \stackrel{?}{=} s + w, v_2 \stackrel{?}{=} y + z\} \\
& \{(x, 1), (y, 0), (z, 0), (s, 0), (w, 0), (v, 0), (v_1, 0), (v_2, 0), (v_3, 1)\} \xRightarrow{U_h} \\
& \{v \stackrel{?}{=} h(v_3), v_3 \stackrel{?}{=} h(x), v \stackrel{?}{=} v_1 + v_2, v_1 \stackrel{?}{=} s + w, v_2 \stackrel{?}{=} y + z\} \\
& \{(x, 2), (y, 0), (z, 0), (s, 0), (w, 0), (v, 0), (v_1, 0), (v_2, 0), (v_3, 1)\}.
\end{aligned}$$

3.2.3. 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, \dots, x_n .

Splitting

$$\frac{\{x \stackrel{?}{=} h(y), x \stackrel{?}{=} x_1 + \dots + x_n\} \cup \Gamma \parallel \triangle \parallel \sigma}{\{x \stackrel{?}{=} h(y), y \stackrel{?}{=} v_1 + \dots + v_n, x_1 \stackrel{?}{=} h(v_1), \dots, x_n \stackrel{?}{=} h(v_n)\} \cup \Gamma \parallel \triangle' \parallel \sigma}$$

where $n > 1$, $x \neq y$ and $x \neq x_i$ for any i , $\triangle' = \{(v_1, 0), \dots, (v_n, 0)\} \cup \triangle$, and v_1, \dots, v_n are fresh variables in \mathcal{NV} .

Example 4. Solve the unification problem $\{h(h(x)) \stackrel{?}{=} y_1 + y_2\}$.

Still we only consider pair $\Gamma \parallel \triangle$, since rules modifying σ are not introduced yet.

$$\begin{aligned}
& \{h(h(x)) \stackrel{?}{=} y_1 + y_2\} \parallel \{(x, 0), (y_1, 0), (y_2, 0)\} \xRightarrow{FBS^+} \\
& \{v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} h(x), v \stackrel{?}{=} y_1 + y_2\} \parallel \{(x, 0), (y_1, 0), (y_2, 0), (v, 0), (v_1, 0)\} \xRightarrow{U_h^+} \\
& \{v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} h(x), v \stackrel{?}{=} y_1 + y_2\} \parallel \{(x, 2), (y_1, 0), (y_2, 0), (v, 0), (v_1, 1)\}, \xRightarrow{Splitting} \\
& \{v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} v_{11} + v_{12}, y_1 \stackrel{?}{=} h(v_{11}), y_2 \stackrel{?}{=} h(v_{12}), v_1 \stackrel{?}{=} h(x)\} \parallel \\
& \{(x, 2), (y_1, 0), (y_2, 0), (v, 0), (v_1, 1), (v_{11}, 0), (v_{12}, 0)\}, \xRightarrow{U_h^+} \\
& \{v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} v_{11} + v_{12}, y_1 \stackrel{?}{=} h(v_{11}), y_2 \stackrel{?}{=} h(v_{12}), v_1 \stackrel{?}{=} h(x)\} \parallel \\
& \{(x, 2), (y_1, 0), (y_2, 0), (v, 0), (v_1, 1), (v_{11}, 1), (v_{12}, 1)\}, \xRightarrow{Splitting} \\
& \{v \stackrel{?}{=} h(v_1), y_1 \stackrel{?}{=} h(v_{11}), y_2 \stackrel{?}{=} h(v_{12}), v_1 \stackrel{?}{=} h(x), x \stackrel{?}{=} v_{13} + v_{14}, v_{11} \stackrel{?}{=} h(v_{13}), \\
& v_{12} \stackrel{?}{=} h(v_{14})\} \parallel \{(x, 2), (y_1, 0), (y_2, 0), (v, 0), (v_1, 1), (v_{11}, 1), (v_{12}, 1), (v_{13}, 0), (v_{14}, 0)\} \xRightarrow{U_h^+} \\
& \{v \stackrel{?}{=} h(v_1), y_1 \stackrel{?}{=} h(v_{11}), y_2 \stackrel{?}{=} h(v_{12}), v_1 \stackrel{?}{=} h(x), x \stackrel{?}{=} v_{13} + v_{14}, v_{11} \stackrel{?}{=} h(v_{13}), \\
& v_{12} \stackrel{?}{=} h(v_{14})\} \parallel \{(x, 2), (y_1, 0), (y_2, 0), (v, 0), (v_1, 1), (v_{11}, 1), (v_{12}, 1), (v_{13}, 2), (v_{14}, 2)\}.
\end{aligned}$$

3.2.4. Trivial

The Trivial inference rule is to remove trivial equations in the given problem Γ .

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

3.2.5. 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.

1 VE1

$$\frac{\{x \stackrel{?}{=} y\} \cup \Gamma || \Delta || \sigma}{\Gamma \{x \mapsto y\} || \Delta || \sigma \{x \mapsto y\} \cup \{x \mapsto y\}} \quad \text{if } x \text{ and } y \text{ are distinct variables}$$

2 VE2

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

The rule VE2 is performed last after all other inference rules have been performed. The rule VE1 is performed eagerly.

Example 5. Solve unification problem $\{x \stackrel{?}{=} y, x \stackrel{?}{=} h(z)\}$.

$$\begin{aligned} & \{x \stackrel{?}{=} y, x \stackrel{?}{=} h(z)\} || \{(x, 0), (y, 0), (z, 0)\} || \emptyset \xRightarrow{Uh} \\ & \{x \stackrel{?}{=} y, x \stackrel{?}{=} h(z)\} || \{(x, 0), (y, 0), (z, 1)\} || \emptyset \xRightarrow{VE1} \\ & \{y \stackrel{?}{=} h(z)\} || \{(x, 0), (y, 0), (z, 1)\} || \{x \mapsto y\} \xRightarrow{VE2} \\ & \emptyset || \{(x, 0), (y, 0), (z, 1)\} || \{x \mapsto h(z), y \mapsto h(z)\}. \end{aligned}$$

The substitution $\{x \mapsto h(z), y \mapsto h(z)\}$ is the most general unifier of the given problem $\{x \stackrel{?}{=} y, x \stackrel{?}{=} h(z)\}$.

3.2.6. Decomposition (Decomp)

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

Decomp

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

Example 6. Solve the unification problem $\{h(h(x)) \stackrel{?}{=} h(h(y))\}$.

$$\begin{aligned}
& \{h(h(x)) \stackrel{?}{=} h(h(y))\} \parallel \{(x, 0), (y, 0)\} \parallel \emptyset \xRightarrow{Flatten^+} \\
& \{v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} h(x), v \stackrel{?}{=} h(v_2), v_2 \stackrel{?}{=} h(y)\} \parallel \{(x, 0), (y, 0), (v, 0), (v_1, 0), (v_2, 0)\} \parallel \emptyset \xRightarrow{Uh^+} \\
& \{v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} h(x), v \stackrel{?}{=} h(v_2), v_2 \stackrel{?}{=} h(y)\} \parallel \{(x, 2), (y, 2), (v, 0), (v_1, 1), (v_2, 1)\} \parallel \emptyset \xRightarrow{Decomp} \\
& \{v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} v_2, v_1 \stackrel{?}{=} h(x), v_2 \stackrel{?}{=} h(y)\} \parallel \{(x, 2), (y, 2), (v, 0), (v_1, 1), (v_2, 1)\} \parallel \emptyset \xRightarrow{VE1} \\
& \{v \stackrel{?}{=} h(v_2), v_2 \stackrel{?}{=} h(x), v_2 \stackrel{?}{=} h(y)\} \parallel \{(x, 2), (y, 2), (v, 0), (v_1, 1), (v_2, 1)\} \parallel \{v_1 \mapsto v_2\} \xRightarrow{Decomp} \\
& \{v \stackrel{?}{=} h(v_2), v_2 \stackrel{?}{=} h(x), x \stackrel{?}{=} y\} \parallel \{(x, 2), (y, 2), (v, 0), (v_1, 1), (v_2, 1)\} \parallel \{v_1 \mapsto v_2\} \xRightarrow{VE2^+} \\
& \emptyset \parallel \{(x, 2), (y, 2), (v, 0), (v_1, 1), (v_2, 1)\} \parallel \{v_1 \mapsto h(y), x \mapsto y, v \mapsto h(h(y)), v_2 \mapsto h(y)\}, \\
& \text{where } \{x \mapsto y\} \text{ is the most general unifier of the problem } \{h(h(x)) \stackrel{?}{=} h(h(y))\}.
\end{aligned}$$

3.2.7. 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, $GetEqs(\{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}) = \{x_1 \stackrel{?}{=} t_1, \dots, x_n \stackrel{?}{=} t_n\}$.

AC Unification

$$\frac{\Psi \cup \Gamma \parallel \Delta \parallel \sigma}{GetEqs(\theta_1) \cup \Gamma \parallel \Delta \parallel \sigma \vee \dots \vee GetEqs(\theta_n) \cup \Gamma \parallel \Delta \parallel \sigma}$$

where $Unify(\Psi) = \{\theta_1, \dots, \theta_n\}$.

We illustrate the applicability of the AC unification rule using the example below. For convenience, we only consider Γ from the problem.

Example 7. Solve the unification problem $\{x+y \stackrel{?}{=} z+y_1, x_1 \stackrel{?}{=} x_2\}$, where x, y, z, x_1, x_2 , and y_1 are pairwise distinct.

$$\begin{aligned}
& \{x+y \stackrel{?}{=} z+y_1, x_1 \stackrel{?}{=} x_2\} \xRightarrow{FBS} \{v \stackrel{?}{=} x+y, v \stackrel{?}{=} z+y_1\} \cup \{x_1 \stackrel{?}{=} x_2\} \xRightarrow{AC\ Unification} \\
& \{v \stackrel{?}{=} c_1+c_2+c_3+c_4, x \stackrel{?}{=} c_1+c_2, y \stackrel{?}{=} c_3+c_4, z \stackrel{?}{=} c_1+c_3, y_1 \stackrel{?}{=} c_2+c_4\} \cup \{x_1 \stackrel{?}{=} x_2\} \vee \\
& \{v \stackrel{?}{=} c+z+y, x \stackrel{?}{=} c+z, y_1 \stackrel{?}{=} c+y\} \cup \{x_1 \stackrel{?}{=} x_2\} \vee \\
& \{v \stackrel{?}{=} z+c+y, x \stackrel{?}{=} z+c, y_1 \stackrel{?}{=} c+y\} \cup \{x_1 \stackrel{?}{=} x_2\} \vee \\
& \{v \stackrel{?}{=} x+c+z, y \stackrel{?}{=} c+z, y_1 \stackrel{?}{=} x+c\} \cup \{x_1 \stackrel{?}{=} x_2\} \vee \\
& \{v \stackrel{?}{=} x+z+c, y \stackrel{?}{=} z+c, y_1 \stackrel{?}{=} x+c\} \cup \{x_1 \stackrel{?}{=} x_2\} \vee \\
& \{v \stackrel{?}{=} z+y_1, x \stackrel{?}{=} z, y \stackrel{?}{=} y_1\} \cup \{x_1 \stackrel{?}{=} x_2\} \vee \\
& \{v \stackrel{?}{=} y_1+z, x \stackrel{?}{=} y_1, y \stackrel{?}{=} z\} \cup \{x_1 \stackrel{?}{=} x_2\}, \\
& \text{where } c, c_1, c_2, c_3, \text{ and } c_4 \text{ are constant symbols.}
\end{aligned}$$

3.2.8. Occur Check (OC)

OC checks if a variable on the left-hand side of an equation occurs on the other side of the equation. If it does, then the problem has no solution. This rule has the highest priority.

<div style="display: flex; justify-content: space-between; align-items: center;"> <div style="text-align: center;"> OC $\frac{\{x \stackrel{?}{=} f(t_1, \dots, t_n)\} \cup \Gamma \Delta \sigma}{\perp}$ </div> <div style="text-align: center;"> If $x \in \text{Var}(f(t_1, \dots, t_n)\sigma)$ where $\text{Var}(f(t_1, \dots, t_n)\sigma)$ represents set of all variables that occur in $f(t_1, \dots, t_n)\sigma$. </div> </div>
--

Example 8. Solve the following unification problem $\{x \stackrel{?}{=} y, y \stackrel{?}{=} z + x\}$.

$\{x \stackrel{?}{=} y, y \stackrel{?}{=} z + x\} || \{(x, 0), (y, 0), (z, 0)\} || \emptyset \xrightarrow{VE1}$
 $\{y \stackrel{?}{=} z + y\} || \{(x, 0), (y, 0), (z, 0)\} || \{x \mapsto y\} \xrightarrow{OC} \text{Fail}.$
 Hence, the problem $\{x \stackrel{?}{=} y, y \stackrel{?}{=} z + x\}$ has no solution.

3.2.9. 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 $+$.

<div style="display: flex; justify-content: space-between; align-items: center;"> <div style="text-align: center;"> Clash $\frac{\{x \stackrel{?}{=} f(s_1, \dots, s_m), x \stackrel{?}{=} g(t_1, \dots, t_n)\} \cup \Gamma \Delta \sigma}{\perp}$ </div> <div style="text-align: center;"> If $f \notin \{h, +\}$ or $g \notin \{h, +\}$ </div> </div>
--

Example 9. Solve the unification problem $\{f(x, y) \stackrel{?}{=} g(h(z))\}$, where f and g are two distinct uninterpreted function symbols.

$\{f(x, y) \stackrel{?}{=} g(h(z))\} || \{(x, 0), (y, 0), (z, 0)\} || \emptyset \xrightarrow{Flatten^+}$
 $\{v \stackrel{?}{=} f(x, y), v \stackrel{?}{=} g(v_1), v_1 \stackrel{?}{=} h(z)\} || \{(x, 0), (y, 0), (z, 0), (v, 0), (v_1, 0)\} || \emptyset \xrightarrow{Uh^+}$
 $\{v \stackrel{?}{=} f(x, y), v \stackrel{?}{=} h(v_1), v_1 \stackrel{?}{=} h(z)\} || \{(x, 0), (y, 0), (z, 1), (v, 0), (v_1, 0)\} || \emptyset \xrightarrow{Clash} \text{Fail}.$
 Hence, the problem $\{f(x, y) \stackrel{?}{=} g(h(z))\}$ has no solution.

3.2.10. Bound Check (BC)

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 Δ exceeds the bound κ , then the problem has no solution. We apply this rule immediately after the rules of update h-depth set.

<div style="display: flex; justify-content: space-between; align-items: center;"> <div style="text-align: center;"> BC $\frac{\Gamma \Delta \sigma}{\perp}$ </div> <div style="text-align: center;"> If $\text{MaxVal}(\Delta) > \kappa$ </div> </div>

Example 10. Solve the following unification problem $\{h(y) \stackrel{?}{=} y + x\}$.

Let the bound be $\kappa = 2$.

$$\begin{aligned}
& \{h(y) \stackrel{?}{=} y + x\} \parallel \{(x, 0), (y, 0)\} \parallel \emptyset \xRightarrow{FBS} \\
& \{v \stackrel{?}{=} h(y), v \stackrel{?}{=} y + x\} \parallel \{(x, 0), (y, 0), (v, 0)\} \parallel \emptyset \xRightarrow{Uh} \\
& \{v \stackrel{?}{=} h(y), v \stackrel{?}{=} y + x\} \parallel \{(x, 0), (y, 1), (v, 0)\} \parallel \emptyset \xRightarrow{Splitting} \\
& \{v \stackrel{?}{=} h(y), y \stackrel{?}{=} v_{11} + v_{12}, y \stackrel{?}{=} h(v_{11}), x \stackrel{?}{=} h(v_{12})\} \parallel \{(x, 0), (y, 1), (v, 0), (v_{11}, 0), (v_{12}, 0)\} \parallel \emptyset \xRightarrow{Uh^+} \\
& \{v \stackrel{?}{=} h(y), y \stackrel{?}{=} v_{11} + v_{12}, y \stackrel{?}{=} h(v_{11}), x \stackrel{?}{=} h(v_{12})\} \parallel \{(x, 0), (y, 1), (v, 0), (v_{11}, 2), (v_{12}, 1)\} \parallel \emptyset \xRightarrow{Splitting} \\
& \{v \stackrel{?}{=} h(y), v_{11} \stackrel{?}{=} v_{13} + v_{14}, v_{11} \stackrel{?}{=} h(v_{13}), v_{12} \stackrel{?}{=} h(v_{14}), y \stackrel{?}{=} h(v_{11}), x \stackrel{?}{=} h(v_{12})\} \\
& \{(x, 0), (y, 1), (v, 0), (v_{11}, 2), (v_{12}, 1), (v_{13}, 0), (v_{14}, 0)\} \parallel \emptyset \xRightarrow{Uh^+} \\
& \{v \stackrel{?}{=} h(y), v_{11} \stackrel{?}{=} v_{13} + v_{14}, v_{11} \stackrel{?}{=} h(v_{13}), v_{12} \stackrel{?}{=} h(v_{14}), y \stackrel{?}{=} h(v_{11}), x \stackrel{?}{=} h(v_{12})\} \\
& \{(x, 0), (y, 1), (v, 0), (v_{11}, 2), (v_{12}, 1), (v_{13}, 3), (v_{14}, 2)\} \parallel \emptyset \xRightarrow{BC} Fail.
\end{aligned}$$

Since $MaxVal(\Delta) = 3 > \kappa$, the problem $\{h(y) \stackrel{?}{=} y + x\}$ has no solution within the given bound.

3.2.11. Orient

The Orient rule swaps the left side term of an equation with the right side term. In particular, when the left side term is a variable but not the right side term.

Orient

$$\frac{\{t \stackrel{?}{=} x\} \cup \Gamma \parallel \Delta \parallel \sigma}{\{x \stackrel{?}{=} t\} \cup \Gamma \parallel \Delta \parallel \sigma} \quad \text{If } t \text{ is not a variable}$$

4. Proof of Correctness

We prove that the proposed inference system is terminating, sound, and complete.

4.1. Termination

Before going to present the proof of termination, we shall introduce few notation which will be used in the subsequent sections. For two set triples, $\Gamma \parallel \Delta \parallel \sigma$ and $\Gamma' \parallel \Delta' \parallel \sigma'$,

- $\Gamma \parallel \Delta \parallel \sigma \Rightarrow_{\mathcal{I}_{ACh}} \Gamma' \parallel \Delta' \parallel \sigma'$, means that the set triple $\Gamma' \parallel \Delta' \parallel \sigma'$ is deduced from $\Gamma \parallel \Delta \parallel \sigma$ by applying a rule from \mathcal{I}_{ACh} once. We call it as one step.
- $\Gamma \parallel \Delta \parallel \sigma \Rightarrow_{\mathcal{I}_{ACh}}^* \Gamma' \parallel \Delta' \parallel \sigma'$, means that the set triple $\Gamma' \parallel \Delta' \parallel \sigma'$ is deduced from $\Gamma \parallel \Delta \parallel \sigma$ by zero or more steps
- $\Gamma \parallel \Delta \parallel \sigma \Rightarrow_{\mathcal{I}_{ACh}}^+ \Gamma' \parallel \Delta' \parallel \sigma'$, means that the set triple $\Gamma' \parallel \Delta' \parallel \sigma'$ is deduced from $\Gamma \parallel \Delta \parallel \sigma$ by one or more steps

As we notice, AC unification divides $\Gamma \parallel \Delta \parallel \sigma$ into finite number of branches $\Gamma_1 \parallel \Delta_1 \parallel \sigma_1$ and so on $\Gamma_n \parallel \Delta_n \parallel \sigma_n$. Hence, for a triple $\Gamma \parallel \Delta \parallel \sigma$, after applying some inference rules, the result is a disjunction of triples $\bigvee_i (\Gamma_i \parallel \Delta_i \parallel \sigma_i)$. Accordingly, we introduce the following notation:

Algorithm 1 AChUnify

Input:

- An equation set Γ , a bound κ , an empty set σ , and an empty h-depth set Δ .

Output:

- A complete set of κ -bounded *AC*h unifiers $\{\sigma_1, \dots, \sigma_n\}$ or \perp indicating that the problem has no solution.
 - 1: Apply *Trivial* to eliminate equations of the form $t \stackrel{?}{=} t$.
 - 2: Apply *OC* to see if any variable on the left side occurs on the right. If yes, then return \perp .
 - 3: Flatten the set of equations Γ using the flattening rules.
 - 4: Update the h-depth set Δ .
 - 5: Apply *BC* to see if $MaxVal(\Delta) > \kappa$. If yes, then return \perp .
 - 6: Apply the *Orient* rule.
 - 7: Apply the *Splitting* rule.
 - 8: Apply the *Clash* rule.
 - 9: Apply the *Decomposition* rule.
 - 10: Apply the *AC Unification* rule.
 - 11: Finally, apply the *Variable Elimination* rule and get the output.
-

- $\Gamma || \Delta || \sigma \Rightarrow_{\mathcal{I}_{AC^h}} \bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$, where $\bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$ is a disjunction of triples, means that the set triple $\Gamma || \Delta || \sigma$ becomes $\bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$ with an application of a rule once.
- $\Gamma || \Delta || \sigma \Rightarrow_{\mathcal{I}_{AC^h}}^+ \bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$ means that $\Gamma || \Delta || \sigma$ becomes $\bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$ after applying some inference rules once or more than once.
- $\Gamma || \Delta || \sigma \Rightarrow_{\mathcal{I}_{AC^h}}^* \bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$ means that $\Gamma || \Delta || \sigma$ becomes $\bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$ after applying some inference rules zero or more times.

Here, we define a measure of $\Gamma || \Delta || \sigma$ for proving termination:

- Let $Sym(\Gamma)$ be a multi-set of non-variable symbols occurring in Γ . The standard ordering of $|Sym(\Gamma)|$ based on natural numbers is a well-founded ordering on the set of equations.
- Let κ be a natural number. Let $\overline{h_d}(\Gamma) := \{(\kappa + 1) - h_d(x, \Gamma) \mid (x, h_d(x, \Gamma)) \in h_d(\Gamma)\}$ be a multi-set. Since every element of the set is a natural number, the multi-set order for $\overline{h_d}(\Gamma)$ is a well-founded ordering.
- Let p be a number of non-solved variables in Γ .
- Let m be the number of equations of the form $f(t) \stackrel{?}{=} x$ in Γ .
- Let n be the number of $+$ -equations with x occurring on the left side, i.e, $x = x_1 + \dots + x_n$.

Then we define the measure of $\Gamma || \Delta || \sigma$ as the following:

$$\mathbb{M}_{\mathcal{I}_{AC^h}}(\Gamma, \Delta, \sigma) = (n, |Sym(\Gamma)|, p, m, |\Gamma|, \overline{h_d}(\Gamma)).$$

Since each element in this tuple with its corresponding order is well-founded, the lexicographic order on this tuple is well-founded as well.

We show that $\mathbb{M}_{\mathcal{J}_{ACh}}(\Gamma, \Delta, \sigma)$ decreases with the application of each rule of the inference system \mathcal{J}_{ACh} except AC unification. The reader can see the proof of termination of the AC unification in (François Fages 1984).

Lemma 1. Let $\Gamma||\Delta||\sigma$ and $\Gamma'||\Delta'||\sigma'$ be two set triples, where Γ and Γ' are in flattened form, such that $\Gamma||\Delta||\sigma \Rightarrow_{\mathcal{J}_{ACh}} \Gamma'||\Delta'||\sigma'$. Then $\mathbb{M}_{\mathcal{J}_{ACh}}(\Gamma, \Delta, \sigma) > \mathbb{M}_{\mathcal{J}_{ACh}}(\Gamma', \Delta', \sigma')$.

Proof. Trivial. The cardinality of Γ , $|\Gamma|$, decreases while other components of the measure either stays the same or decreases. Hence, $\mathbb{M}_{\mathcal{J}_{ACh}}(\Gamma, \Delta, \sigma) > \mathbb{M}_{\mathcal{J}_{ACh}}(\Gamma', \Delta', \sigma')$.

Decomposition. The number of f symbols decreased by one, and hence $|Sym(\Gamma)|$ decreases while p stays the same. Hence, $\mathbb{M}_{\mathcal{J}_{ACh}}(\Gamma, \Delta, \sigma) > \mathbb{M}_{\mathcal{J}_{ACh}}(\Gamma', \Delta', \sigma')$.

Update h-Depth Set. On application of one of the update rules, increases h-depth of a variable x from n to $n+1$. However, $\kappa \cdot n > \kappa \cdot (n+1)$. Which means that $\overline{h_d}(\Gamma)$ decreases while the other components stay the same. Hence, $\mathbb{M}_{\mathcal{J}_{ACh}}(\Gamma, \Delta, \sigma) > \mathbb{M}_{\mathcal{J}_{ACh}}(\Gamma', \Delta', \sigma')$.

Splitting. On the application of the Splitting rule, n , the number of $+$ -equations with x on the left side decreased by one. So, $\mathbb{M}_{\mathcal{J}_{ACh}}(\Gamma, \Delta, \sigma) > \mathbb{M}_{\mathcal{J}_{ACh}}(\Gamma', \Delta', \sigma')$.

Orient. It is not difficult to see the fact that m decreases.

Variable elimination. Of course, the number of non-solved variables decreases in the application of this rule.

□

Theorem 2 (Termination). For any set triple $\Gamma||\Delta||\sigma$, there is a set triple $\Gamma'||\Delta'||\sigma'$ such that $\Gamma||\Delta||\sigma \Rightarrow_{\mathcal{J}_{ACh}}^* \Gamma'||\Delta'||\sigma'$ and none of the rules \mathcal{J}_{ACh} can be applied on $\Gamma'||\Delta'||\sigma'$.

Proof. By induction on Lemma 1, this theorem can be proved.

□

4.2. Soundness

In this Section, we show that our inference system \mathcal{J}_{ACh} is truth-preserving.

Lemma 3. Let $\Gamma||\Delta||\sigma$ and $\Gamma'||\Delta'||\sigma'$ be two set triples such that $\Gamma||\Delta||\sigma \Rightarrow_{\mathcal{J}_{ACh}} \Gamma'||\Delta'||\sigma'$ via all the rules of \mathcal{J}_{ACh} except AC unification. Let θ be a substitution such that $\theta \models \Gamma'||\Delta'||\sigma'$. Then $\theta \models \Gamma||\Delta||\sigma$.

Proof. Trivial. It is trivially true.

Splitting. Let θ be a substitution. Assume that θ satisfies $\{w \stackrel{?}{=} h(y), y \stackrel{?}{=} v_1 + \dots + v_n, x_1 \stackrel{?}{=} h(v_1), \dots, x_n \stackrel{?}{=} h(v_n)\} \cup \Gamma$. Then we have that $w\theta \stackrel{?}{=} h(y)\theta$, $y\theta \stackrel{?}{=} (v_1 + \dots + v_n)\theta$, $x_1\theta \stackrel{?}{=} h(v_1)\theta$, \dots , $x_n\theta \stackrel{?}{=} h(v_n)\theta$. This implies that $w\theta \stackrel{?}{=} h(y\theta)$, $y\theta \stackrel{?}{=} v_1\theta + \dots + v_n\theta$, $x_1\theta \stackrel{?}{=} h(v_1\theta) \dots x_n\theta \stackrel{?}{=} h(v_n\theta)$. In order to prove that θ satisfies $\{w \stackrel{?}{=} h(y), w \stackrel{?}{=} x_1 + \dots + x_n\}$, it is enough to prove θ satisfies the equation $w \stackrel{?}{=} x_1 + \dots + x_n$. By considering the right side term $x_1 + \dots + x_n$ and after applying the substitution, we get $(x_1 + \dots + x_n)\theta \stackrel{?}{=} x_1\theta + \dots + x_n\theta \stackrel{?}{=} h(v_1\theta) + \dots + h(v_n\theta)$. By the homomorphism theory, we write that

$h(v_1\theta) + \dots + h(v_n\theta) \stackrel{?}{=} h(v_1\theta + \dots + v_n\theta)$. Then $h(v_1\theta + \dots + v_n\theta) \stackrel{?}{=} h(y\theta) \stackrel{?}{=} w\theta$. Hence, θ satisfies $w \stackrel{?}{=} x_1 + \dots + x_n$.

Variable Elimination.

VE1. Assume that $\theta \models \Gamma\{x \mapsto y\} \parallel \Delta \parallel \sigma\{x \mapsto y\} \cup \{x \mapsto y\}$. This means that θ satisfies $\Gamma\{x \mapsto y\}$ and $\sigma\{x \mapsto y\} \cup \{x \mapsto y\}$. Now, we have to prove that θ satisfies $\{x \stackrel{?}{=} y\}, \Gamma$, and σ . But θ satisfies $x \mapsto y$ means that $x\theta \stackrel{?}{=} y\theta$. Γ is $\Gamma\{x \mapsto y\}$ but without replacing x with y . Since $y\theta \stackrel{?}{=} x\theta$, the substitution θ satisfies $y \mapsto x$. Hence, we conclude that θ satisfies Γ and σ .

VE2. We have that θ satisfies Γ and $\sigma\{x \mapsto t\} \cup \{x \mapsto t\}$. Now, we have to prove that θ satisfies $\{x \stackrel{?}{=} t\}$ and σ . By the definition of $\theta \models \Gamma$, we have $x\theta \stackrel{?}{=} t\theta$ and it is enough to prove that θ satisfies σ . Let $w \mapsto s[x]$ be an assignment in σ . After applying $x \mapsto t$ on σ , the assignment $y \mapsto s$ with $s|_p = x$, where p is a position, becomes $y \mapsto s[t]_p$. We also know that θ satisfies $\sigma\{x \mapsto t\}$ implies that θ also satisfies $w \mapsto s[t]_p$. Then by the definition, we write that $y\theta \stackrel{?}{=} s[t\theta]_p \stackrel{?}{=} s[x\theta]_p$. This means that θ satisfies the assignment $w \mapsto s[x]$. Hence, θ satisfies σ .

Decomposition. Assume that $\theta \models \{x \stackrel{?}{=} f(t_1, \dots, t_n), s_1 \stackrel{?}{=} t_1, \dots, s_n \stackrel{?}{=} t_n\} \cup \Gamma \parallel \Delta \parallel \sigma$. This means that θ satisfies $\{x \stackrel{?}{=} f(t_1, \dots, t_n), s_1 \stackrel{?}{=} t_1, \dots, s_n \stackrel{?}{=} t_n\} \cup \Gamma$. Now we have to prove that θ satisfies $\{x \stackrel{?}{=} f(s_1, \dots, s_n), x \stackrel{?}{=} f(t_1, \dots, t_n)\} \cup \Gamma$. Given that θ satisfies $x \stackrel{?}{=} f(t_1, \dots, t_n)$ and it is enough to show that θ also satisfies $x \stackrel{?}{=} f(s_1, \dots, s_n)$. We write $x\theta \stackrel{?}{=} f(t_1, \dots, t_n)\theta \stackrel{?}{=} f(t_1\theta, t_2\theta, \dots, t_n\theta) \stackrel{?}{=} f(s_1\theta, s_2\theta, \dots, s_n\theta)$ since $s_1\theta \stackrel{?}{=} t_1\theta, \dots, s_n\theta \stackrel{?}{=} t_n\theta$. So, θ satisfies $x \stackrel{?}{=} f(t_1, \dots, t_n)$ and $x \stackrel{?}{=} f(s_1, \dots, s_n)$. Hence, $\theta \models \{x \stackrel{?}{=} f(s_1, \dots, s_n), x \stackrel{?}{=} f(t_1, \dots, t_n)\}$. □

Lemma 4. Let $\Gamma \parallel \Delta \parallel \sigma$ and $\Gamma' \parallel \Delta' \parallel \sigma'$ be two set triples such that $\Gamma \parallel \Delta \parallel \sigma \Rightarrow_{\text{AC}} \bigvee_i (\Gamma_i \parallel \Delta_i \parallel \sigma_i)$ via AC unification. Let θ be a substitution such that $\theta \models \Gamma_i \parallel \Delta_i \parallel \sigma_i$. Then $\theta \models \Gamma \parallel \Delta \parallel \sigma$.

Proof. **AC Unification.**

$$\frac{\Psi \cup \Gamma \parallel \Delta \parallel \sigma}{\text{GetEqs}(\theta_1) \cup \Gamma \parallel \Delta \parallel \sigma \vee \dots \vee \text{GetEqs}(\theta_n) \cup \Gamma \parallel \Delta \parallel \sigma}$$

Given that $\theta \models \text{GetEqs}(\theta_1) \cup \Gamma \parallel \Delta \parallel \sigma \vee \dots \vee \text{GetEqs}(\theta_n) \cup \Gamma \parallel \Delta \parallel \sigma$. This means that θ satisfies $\text{GetEqs}(\theta_1) \cup \Gamma \parallel \Delta \parallel \sigma, \dots, \text{GetEqs}(\theta_n) \cup \Gamma \parallel \Delta \parallel \sigma$. Which implies that θ also satisfies Ψ . □

By combining Lemma 3 and Lemma 4, we have:

Lemma 5. Let $\Gamma \parallel \Delta \parallel \sigma$ and $\Gamma' \parallel \Delta' \parallel \sigma'$ be two set triples such that $\Gamma \parallel \Delta \parallel \sigma \Rightarrow_{\text{AC}} \bigvee_i (\Gamma_i \parallel \Delta_i \parallel \sigma_i)$. Let θ be a substitution such that $\theta \models \Gamma_i \parallel \Delta_i \parallel \sigma_i$. Then $\theta \models \Gamma \parallel \Delta \parallel \sigma$.

Then by induction on Lemma 5, we get the following theorem:

Theorem 6. Let $\Gamma \parallel \Delta \parallel \sigma$ and $\Gamma' \parallel \Delta' \parallel \sigma'$ be two set triples such that

$\Gamma \parallel \Delta \parallel \sigma \Rightarrow^*_{\mathfrak{J}_{ACh}} \bigvee_i (\Gamma_i \parallel \Delta_i \parallel \sigma_i)$. Let θ be a substitution such that $\theta \models \Gamma_i \parallel \Delta_i \parallel \sigma_i$. Then $\theta \models \Gamma \parallel \Delta \parallel \sigma$.

We have the following corollary from Theorem 6:

Theorem 7 (Soundness). Let σ be a set of equations. Suppose that we get $\bigvee_i (\Gamma_i \parallel \Delta_i \parallel \sigma_i)$ after exhaustively applying the rules from \mathfrak{J}_{ACh} to $\Gamma \parallel \Delta \parallel \sigma$, i.e., $\Gamma \parallel \Delta \parallel \sigma \Rightarrow^*_{\mathfrak{J}_{ACh}} \bigvee_i (\Gamma_i \parallel \Delta_i \parallel \sigma_i)$, where for each i , no rules applicable to $\Gamma_i \parallel \Delta_i \parallel \sigma_i$. Let $\Sigma = \{\sigma_i \mid \Gamma_i = \emptyset\}$. Then any member of Σ is an ACh-unifier of Γ .

4.3. Completeness

Before going to prove the completeness of our inference system, we present a definition below:

Definition 12 (Directed conservative extension). Let $\bigvee_i (\Gamma_i \parallel \Delta_i \parallel \sigma_i)$ and $\bigvee_i (\Gamma'_i \parallel \Delta'_i \parallel \sigma'_i)$ be two set triples. $\bigvee_i (\Gamma'_i \parallel \Delta'_i \parallel \sigma'_i)$ is called a directed conservative extension of $\bigvee_i (\Gamma_i \parallel \Delta_i \parallel \sigma_i)$, if for any substitution θ , such that $\theta \models \Gamma_i \parallel \Delta_i \parallel \sigma_i$, then there exists k and σ , whose domain is the variables in $Var(\Gamma'_k) \setminus Var(\Gamma_k)$, such that $\theta\sigma \models \Gamma'_i \parallel \Delta'_i \parallel \sigma'_i$. If $\bigvee_i (\Gamma_i \parallel \Delta_i \parallel \sigma_i)$ (resp. $\bigvee_i (\Gamma'_i \parallel \Delta'_i \parallel \sigma'_i)$) only contains one set triple $\Gamma \parallel \Delta \parallel \sigma$ (resp. $\Gamma' \parallel \Delta' \parallel \sigma'$), we say $\bigvee_i (\Gamma'_i \parallel \Delta'_i \parallel \sigma'_i)$ (resp. $\Gamma' \parallel \Delta' \parallel \sigma'$) is a directed conservative extension of $\Gamma \parallel \Delta \parallel \sigma$.

Next, we show that our inference procedure never loses any solution.

Lemma 8. Let $\Gamma \parallel \Delta \parallel \sigma$ be a set triple. If there exists a set triple $\Gamma' \parallel \Delta' \parallel \sigma'$ such that $\Gamma \parallel \Delta \parallel \sigma \Rightarrow_{\mathfrak{J}_{ACh}} \Gamma' \parallel \Delta' \parallel \sigma'$ via all the rules of \mathfrak{J}_{ACh} except AC unification, then $\Gamma' \parallel \Delta' \parallel \sigma'$ is a directed conservative extension of $\Gamma \parallel \Delta \parallel \sigma$.

Proof. Trivial. It is trivially true.

Occur Check. In the homomorphism theory, no term can be equal to a subterm of itself. This is because the number of $+$ symbols and h-depth of each variable stay the same with the application of the homomorphism equation $h(x_1 + \dots + x_n) \stackrel{?}{=} h(x_1) + \dots + h(x_n)$. So, the given problem has no solution in the homomorphism theory.

Bound Check. We see that there exists a variable y with the h-depth $\kappa + 1$ in the graph, that is, there is a variable x above y with $\kappa + 1$ h-symbols below it. Let θ be a solution of the unification problem Γ . Then the term $x\theta$ has the h-height $\kappa + 1$, but the term $x\theta$ is also a subterm of some $s_i\theta$ or $t_i\theta$ in the original unification problem. Hence, the unification problem Γ has no solution within the given bound κ .

Clash. We don't have a rewrite rule that deals with the uninterpreted function symbols, i.e., the function symbols which are not in $\{h, +\}$. So the given problem has to have no solution.

Splitting. We have to make sure that we never lose any solution with this rule. Here we consider the rewrite system R_1 which has the rewrite rule $h(x_1 + \dots + x_n) \rightarrow h(x_1) + \dots + h(x_n)$. In order to apply this rule the term under the h should be the sum of n variables. The problem $\{h(y) \stackrel{?}{=} x_1 + \dots + x_n\}$ is replaced by the set $\{h(v_1 + \dots + v_n) \stackrel{?}{=} x_1 + \dots + x_n\}$ with the substitution $\{y \mapsto v_1 + \dots + v_n\}$. Then we have the equation with the reduced term in R_1 is the equation $h(v_1) + \dots + h(v_n) \stackrel{?}{=} x_1 + \dots + x_n$, and the substitution $\{y \mapsto v_1 + \dots + v_n, x_1 \mapsto h(v_1), \dots, x_n \mapsto h(v_n)\}$. Hence, we never lose any solution here.

Decomposition. If f is the top symbol on both sides of an equation then there is no other rule to solve it except the Decomposition rule, where $f \neq h$ and $f \neq +$. So, we never lose any solution.

To cover the case where the top symbol is h for the terms on both sides of an equation, we consider the rewrite system R_2 which has the rewrite rule $h(x_1) + h(x_2) \rightarrow h(x_1 + x_2)$. In the homomorphism theory with the rewrite system R_2 , we cannot reduce the term $h(t)$. So, we solve the equation of the form $h(t_1) \stackrel{?}{=} h(t_2)$ only with the Decomposition rule. Hence, we never lose any solution here too. \square

Lemma 9. Let $\Gamma || \Delta || \sigma$ be a set triple. If there exists a set triple $\Gamma' || \Delta' || \sigma'$ such that $\Gamma || \Delta || \sigma \Rightarrow_{\mathcal{I}_{AC^h}} \bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$ via AC unification, then $\bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$ is a directed conservative extension of $\Gamma || \Delta || \sigma$.

Proof. Since the built-in AC unification algorithm is complete, we never lose any solutions on the application of this rule. \square

By combining Lemma 8 and Lemma 9, we have:

Lemma 10. Let $\Gamma || \Delta || \sigma$ be a set triple. If there exists a set of set triples $\bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$ such that $\Gamma || \Delta || \sigma \Rightarrow_{\mathcal{I}_{AC^h}} \bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$, then $\bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$ is a directed conservative extension of $\Gamma || \Delta || \sigma$.

By induction on Lemma 10, we get:

Theorem 11. Let $\Gamma || \Delta || \sigma$ be a set triple. If there exists a set of set triples $\bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$ such that $\Gamma || \Delta || \sigma \Rightarrow_{\mathcal{I}_{AC^h}}^+ \bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$, then $\bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$ is a directed conservative extension of $\Gamma || \Delta || \sigma$.

We get the following corollary from the above theorem:

Theorem 12 (Completeness). Let Γ be a set of equations. Suppose that we get $\bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$ after applying the rules from \mathcal{I}_{AC^h} to $\Gamma || \Delta || \sigma$ exhaustively, that is, $\Gamma || \Delta || \sigma \Rightarrow_{\mathcal{I}_{AC^h}}^* \bigvee_i (\Gamma_i || \Delta_i || \sigma_i)$, where for each i , none of the rules applicable on $\Gamma_i || \Delta_i || \sigma_i$. Let $\Sigma = \{\sigma_i \mid \Gamma_i = \emptyset\}$. Then for any AC^h -unifier θ of Γ , there exists a $\sigma \in \Sigma$, such that $\sigma \lesssim_{AC^h}^{Var(\Gamma)} \theta$.

5. Implementation

We have implemented the algorithm in the Maude programming language[†]. The implementation of this inference system is available[‡]. We chose the Maude language because it provides a nice environment for expressing inference rules of this algorithm. The system specifications of this implementation 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 (milliseconds),

[†] http://maude.cs.illinois.edu/w/index.php/The_Maude_System

[‡] https://github.com/ajayeeralla/Unification_ACh

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*

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 a threshold κ on the h-depth of any variable. Homomorphism is a property that 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 (Anantharaman *et al.* 2012; Escobar *et al.* 2011; Anantharaman *et al.* 2010). However, none of those results perform ACh unification because that is undecidable. 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.

Acknowledgments. We thank anonymous reviewers who have provided useful comments. Ajay Kumar Eeralla was partially supported by NSF Grant CNS 1314338.

References

- 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.
- 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.
- F. Baader and T. Nipkow. *Term Rewriting and All that*. Cambridge University Press, 1998.
- F. Baader and W. Snyder. Unification Theory. In *Handbook of Automated Reasoning*, pages 447–533. Elsevier, 2001.
- 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.
- 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.
- 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.
- 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.
- S. Kremer, M. Ryan, and B. Smyth. Election Verifiability in Electronic Voting Protocols. In *Computer Security – ESORICS*, pages 389–404. Springer, 2010.
- 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.
- A. M. Marshall, C. A. Meadows, and P. Narendran. On Unification Modulo One-Sided Distributivity: Algorithms, Variants and Asymmetry. In *Logical Methods in Computer Science*, **11** (2), pages 1–39, 2015.
- M. Schmidt-Schauß. A Decision Algorithm for Distributive Unification. In *Theoretical Computer Science*, pages 111–148. Elsevier, 1998.
- E. Tidén and S. Arnborg. Unification Problems with One-Sided Distributivity. In *booktitle of Symbolic Computation*, pages 183–202. Springer, 1987.
- F. Fages. Associative-Commutative Unification In *7th International Conference on Automated Deduction* pages 194–208. Springer, 1984.
- Z. Liu and C. Lynch. Efficient General Unification for XOR with Homomorphism. In *Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction*, pages 407–421. Springer, 2011.