Mathematical Structures in Computer Science (2020), **30**, pp. 664–682
doi:10.1017/S0960129520000183

CAMBRIDGE UNIVERSITY PRESS

PAPER

Bounded ACh unification

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

[1]Galois Inc, Portland, Oregon, 97204 and [2]Department of Computer Science, Clarkson University, Potsdam, NY 13699, USA
*Corresponding author. Email: aeeralla@galois.com

(Received 27 February 2019; revised 22 May 2020; accepted 4 August 2020; first published online 16 September 2020)

Abstract

We consider the problem of the unification modulo an equational theory associativity and commutativity (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.

Keywords: Equational unification, cryptographic protocol analysis, rewriting, homomorphism, commutativity, Maude

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 $\{x \mapsto b,\ y \mapsto a\}$; 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 Escobar et al. (2011); Kapur et al. (2003); 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 (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) 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 Tiden and Arnborg 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 heavily 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 (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 (TRSs) 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 $\sigma=\{x_1 \mapsto t_1,\ldots,x_n \mapsto t_n\}$, $x_i \neq t_i$, where the domain of σ is $Dom(\sigma):=\{x_1,\ldots,x_n\}$. The range of σ, denoted as $Range(\sigma)$, is 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
- $f(t_1\sigma,\ldots,t_n\sigma)$, where $t=f(t_1,\ldots,t_n)$ 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, that is, 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,*

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

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 is 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 write $\theta \models_E \Gamma$ when θ is an E-unifier of Γ. 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. *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 $\sigma \lesssim_E^{\mathcal{X}} \theta$.*

Definition 5 (Complete set of *E*-unifiers). *Let Γ be an 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 an E-unifier of Γ, that is, $S \subseteq \mathcal{U}_E(\Gamma)$, 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, that is, $\sigma \lesssim_E^{Var(\Gamma)} \theta$.*

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, that is, $\sigma \lesssim_E^{Var(\Gamma)} \theta$ 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 $l \to r$, where l is not a variable and $Var(r) \subseteq Var(l)$. A 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 $u \to_R v$ (or simply $u \to v$), if there exists a position p of u, $l \to r \in R$, 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 $u_0 \to_R u_1 \to_R \ldots$. A TRS R is *confluent* if, whenever $u \to_R^* s_1$ and $u \to_R^* s_2$, there exists a term v such that $s_1 \to_R^* v$ and $s_2 \to_R^* v$. A TRS R is convergent if it is both confluent and terminating.

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

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 ACh. We abbreviate this theory as ACh. The signature $\mathcal{F}$ includes a unary symbol h, 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 ACh.

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

2.4 h-depth set

For convenience, we assume that our unification problem is in *flattened* form, that is, 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 and y are variables, y_is and x_is are pairwise distinct variables, and f is a free symbol with $n \geq 0$. The first kind of equations is called *VarVar equations*. The second kind is called *h-equations*. The third kind is called *+-equations*. The fourth kind is 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 $y \stackrel{?}{=} f(x_1, \ldots, x_n)$, where f is a symbol with $n \geq 1$, is in Γ, then the graph $\mathbb{G}(\Gamma)$ contains n edges $y \stackrel{f}{\to} x_1, \ldots, y \stackrel{f}{\to} x_n$. 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 $h_{dh}(x, \Gamma) := \max\{1 + h_d(y, \Gamma) \mid y \stackrel{h}{\to} x$ is an edge in $\mathbb{G}(\Gamma)\}$ and $h_{df}(x, \Gamma) := \max\{h_d(y, \Gamma) \mid$ there exists $f \neq h$ such that $y \stackrel{f}{\to} x$ is in $\mathbb{G}(\Gamma)\}$.

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

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

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

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

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 $h_{ds}(\Gamma) := \{(x, h_d(x, \Gamma)) \mid x$ 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 occurs 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 $MaxVal(\triangle)$, that is, $MaxVal(\triangle) := \max\{c \mid (x,\ c) \in \triangle$ for some $x\}$.

Definition 11 (*ACh*-unification problem, bounded *ACh*-unifier). *An ACh-unification problem over $\mathcal{F}$ is a finite set of equations $\Gamma = \{s_1 \stackrel{?}{=}_{ACh} t_1, \ldots, s_n \stackrel{?}{=}_{ACh} t_n\}$, $s_i, t_i \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, 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 $\Gamma||\triangle||\sigma$ similar to the format presented in Liu and Lynch (2011), where Γ is a unification problem modulo the ACh theory, $\triangle$ 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 $\Gamma||\triangle||\sigma$ if θ satisfies σ and every equation in Γ, $MaxVal(\triangle) \leq \kappa$, and we write that relation as $\theta \models \Gamma||\triangle||\sigma$. $\Gamma||\triangle||\sigma$ is said to be in solved form if Γ is empty and $MaxVal(\triangle) \leq \kappa$. 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* nondeterminism when there is no priority, that is, any rule applied from a set of rules without priority. Initially, Γ is the non-empty set of equations to solve, $\triangle$ 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 $\frac{\Gamma||\triangle||\sigma}{\Gamma'||\triangle'||\sigma'}$. This means that if something matches the top of this rule, then it is to be replaced with the bottom of the rule.

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 $w, t, t_1, \ldots, t_n, s, s_1, \ldots, s_n$ in $\mathcal{T}(\mathcal{F}, \mathcal{V})$, and f and g are uninterpreted function symbols. 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 $\triangle$ 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:

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

Flatten Both Sides (FBS)

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

Flatten Left + (FL)

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

Flatten Right + (FR)

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

Flatten Under h (FU)

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

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.

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

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

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

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

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

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

where $\stackrel{FU^+}{\Rightarrow}$ 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)**

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

2 **Update Right + (UR)**

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

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 $\Gamma||\triangle$,

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

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.

$$\{v \stackrel{?}{=} h(v_3),\ v_3 \stackrel{?}{=} h(x),\ v \stackrel{?}{=} v_1 + v_2,\ v_1 \stackrel{?}{=} s + w,\ v_2 \stackrel{?}{=} y + z\}||$$
$$\{(x,\ 0), (y,\ 0), (z,\ 0), (s,\ 0), (w,\ 0), (v,\ 0), (v_1,\ 0), (v_2,\ 0), (v_3,\ 0)\} \stackrel{Uh}{\Rightarrow}$$

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

$\{v \stackrel{?}{=} h(v_3),\ v_3 \stackrel{?}{=} h(x),\ v \stackrel{?}{=} v_1 + v_2,\ v_1 \stackrel{?}{=} s + w,\ v_2 \stackrel{?}{=} y + z\}||$
$\{(x,\ 1), (y,\ 0), (z,\ 0), (s,\ 0), (w,\ 0), (v,\ 0), (v_1,\ 0), (v_2,\ 0), (v_3,\ 0)\} \stackrel{Uh}{\Rightarrow}$
$\{v \stackrel{?}{=} h(v_3),\ v_3 \stackrel{?}{=} h(x),\ v \stackrel{?}{=} v_1 + v_2,\ v_1 \stackrel{?}{=} s + w,\ v_2 \stackrel{?}{=} y + z\}||$
$\{(x,\ 1), (y,\ 0), (z,\ 0), (s,\ 0), (w,\ 0), (v,\ 0), (v_1,\ 0), (v_2,\ 0), (v_3,\ 1)\} \stackrel{Uh}{\Rightarrow}$
$\{v \stackrel{?}{=} h(v_3),\ v_3 \stackrel{?}{=} h(x),\ v \stackrel{?}{=} v_1 + v_2,\ v_1 \stackrel{?}{=} s + w,\ v_2 \stackrel{?}{=} y + z\}||$
$\{(x,\ 2), (y,\ 0), (z,\ 0), (s,\ 0), (w,\ 0), (v,\ 0), (v_1,\ 0), (v_2,\ 0), (v_3,\ 1)\}.$

3.2.3 Splitting rule

This rule takes the homomorphism theory into account. In this theory, we cannot 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 $x_1, \ldots, x_n$.

Splitting

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

where $n > 1$, $x \neq y$ and $x \neq x_i$ for any i, $\triangle' = \{(v_1,\ 0), \ldots, (v_n,\ 0)\} \cup \triangle$, and $v_1, \ldots, v_n$ 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 $\Gamma||\triangle$, since rules modifying σ are not introduced yet.

$\{h(h(x)) \stackrel{?}{=} y_1 + y_2\}|||\{(x,\ 0), (y_1,\ 0), (y_2,\ 0)\} \stackrel{FBS^+}{\Rightarrow}$
$\{v \stackrel{?}{=} h(v_1),\ v_1 \stackrel{?}{=} h(x),\ v \stackrel{?}{=} y_1 + y_2\}|||\{(x,\ 0), (y_1,\ 0), (y_2,\ 0), (v,\ 0), (v_1,\ 0)\} \stackrel{Uh^+}{\Rightarrow}$
$\{v \stackrel{?}{=} h(v_1),\ v_1 \stackrel{?}{=} h(x),\ v \stackrel{?}{=} y_1 + y_2\}||||\{(x,\ 2), (y_1,\ 0), (y_2,\ 0), (v,\ 0), (v_1,\ 1)\}, \stackrel{Splitting}{\Rightarrow}$
$\{v \stackrel{?}{=} h(v_1),\ v_1 \stackrel{?}{=} v_{11} + v_{12},\ y_1 \stackrel{?}{=} h(v_{11}),\ y_2 \stackrel{?}{=} h(v_{12}),\ v_1 \stackrel{?}{=} h(x)\}||$
$\{(x,\ 2), (y_1,\ 0), (y_2,\ 0), (v,\ 0), (v_1,\ 1), (v_{11},\ 0), (v_{12},\ 0)\}, \stackrel{Uh^+}{\Rightarrow}$
$\{v \stackrel{?}{=} h(v_1),\ v_1 \stackrel{?}{=} v_{11} + v_{12},\ y_1 \stackrel{?}{=} h(v_{11}),\ y_2 \stackrel{?}{=} h(v_{12}),\ v_1 \stackrel{?}{=} h(x)\}||$
$\{(x,\ 2), (y_1,\ 0), (y_2,\ 0), (v,\ 0), (v_1,\ 1), (v_{11},\ 1), (v_{12},\ 1)\}, \stackrel{Splitting}{\Rightarrow}$
$\{v \stackrel{?}{=} h(v_1),\ y_1 \stackrel{?}{=} h(v_{11}),\ y_2 \stackrel{?}{=} h(v_{12}),\ v_1 \stackrel{?}{=} h(x),\ x \stackrel{?}{=} v_{13} + v_{14},\ v_{11} \stackrel{?}{=} h(v_{13}),$
$v_{12} \stackrel{?}{=} h(v_{14})\}|||\{(x,\ 2), (y_1,\ 0), (y_2,\ 0), (v,\ 0), (v_1,\ 1), (v_{11},\ 1), (v_{12},\ 1), (v_{13},\ 0), (v_{14},\ 0)\} \stackrel{Uh^+}{\Rightarrow}$
$\{v \stackrel{?}{=} h(v_1),\ y_1 \stackrel{?}{=} h(v_{11}),\ y_2 \stackrel{?}{=} h(v_{12}),\ v_1 \stackrel{?}{=} h(x),\ x \stackrel{?}{=} v_{13} + v_{14},\ v_{11} \stackrel{?}{=} h(v_{13}),$
$v_{12} \stackrel{?}{=} h(v_{14})\}|||\{(x,\ 2), (y_1,\ 0), (y_2,\ 0), (v,\ 0), (v_1,\ 1), (v_{11},\ 1), (v_{12},\ 1), (v_{13},\ 2), (v_{14},\ 2)\}.$

3.2.4 Trivial

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

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

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

3.2.5 Variable elimination

The variable elimination (VE) rule is to convert the equations into assignments. In other words, it is used to find the most general unifier.

1 **VE1**

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

2 **VE2**

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

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

$\{x \stackrel{?}{=} y,\ x \stackrel{?}{=} h(z)\}||\{(x,\ 0),\ (y,\ 0),\ (z,\ 0)\}||\emptyset \stackrel{Uh}{\Rightarrow}$
$\{x \stackrel{?}{=} y,\ x \stackrel{?}{=} h(z)\}||\{(x,\ 0),\ (y,\ 0),\ (z,\ 1)\}||\emptyset \stackrel{VE1}{\Rightarrow}$
$\{y \stackrel{?}{=} h(z)\}||\{(x,\ 0),\ (y,\ 0),\ (z,\ 1)\}||\{x \mapsto y\} \stackrel{VE2}{\Rightarrow}$
$\emptyset||\{(x,\ 0),\ (y,\ 0),\ (z,\ 1)\}||\{x \mapsto h(z),\ y \mapsto h(z)\}$.

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

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

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

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

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

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

AC unification

$$\frac{\Psi \cup \Gamma || \triangle || \sigma}{GetEqs(\theta_1) \cup \Gamma || \triangle || \sigma \vee \ldots \vee GetEqs(\theta_n) \cup \Gamma || \triangle || \sigma}$$

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

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.

$$\{x + y \stackrel{?}{=} z + y_1,\ x_1 \stackrel{?}{=} x_2\} \stackrel{FBS}{\Rightarrow} \{v \stackrel{?}{=} x + y,\ v \stackrel{?}{=} z + y_1\} \cup \{x_1 \stackrel{?}{=} x_2\} \stackrel{AC\ Unification}{\Rightarrow}$$
$$\{v \stackrel{?}{=} v_1 + v_2 + v_3 + v_4,\ x \stackrel{?}{=} v_1 + v_2,\ y \stackrel{?}{=} v_3 + v_4, z \stackrel{?}{=} v_1 + v_3,\ y_1 \stackrel{?}{=} v_2 + v_4\} \cup \{x_1 \stackrel{?}{=} x_2\} \vee$$
$$\{v \stackrel{?}{=} v_5 + z + y,\ x \stackrel{?}{=} v_5 + z,\ y_1 \stackrel{?}{=} v_5 + y\} \cup \{x_1 \stackrel{?}{=} x_2\} \vee$$
$$\{v \stackrel{?}{=} z + v_5 + y,\ x \stackrel{?}{=} z + v_5,\ y_1 \stackrel{?}{=} v_5 + y\} \cup \{x_1 \stackrel{?}{=} x_2\} \vee$$
$$\{v \stackrel{?}{=} x + v_5 + z,\ y \stackrel{?}{=} v_5 + z,\ y_1 \stackrel{?}{=} x + v_5\} \cup \{x_1 \stackrel{?}{=} x_2\} \vee$$
$$\{v \stackrel{?}{=} x + z + v_5,\ y \stackrel{?}{=} z + v_5,\ y_1 \stackrel{?}{=} x + v_5\} \cup \{x_1 \stackrel{?}{=} x_2\} \vee$$
$$\{v \stackrel{?}{=} z + y_1,\ x \stackrel{?}{=} z,\ y \stackrel{?}{=} y_1\} \cup \{x_1 \stackrel{?}{=} x_2\} \vee$$
$$\{v \stackrel{?}{=} y_1 + z,\ x \stackrel{?}{=} y_1,\ y \stackrel{?}{=} z\} \cup \{x_1 \stackrel{?}{=} x_2\},$$

where v_1, v_2, v_3, v_4, and v_5 are new variables.

3.2.8 Occur check

Occur check (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.

OC

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

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

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

$$\{x \stackrel{?}{=} y,\ y \stackrel{?}{=} z + x\} || \{(x,\ 0), (y,\ 0), (z,\ 0)\} || \emptyset \stackrel{VE1}{\Rightarrow}$$

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

$\{y \stackrel{?}{=} z + y\}||\{(x,\ 0), (y,\ 0), (z,\ 0)\}||\{x \mapsto y\} \stackrel{OC}{\Rightarrow}$ *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 $+$.

Clash

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

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

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

3.2.10 Bound check
The bound check (BC) 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. We apply this rule immediately after the rules of update h-depth set.

BC

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

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

Let the bound be $\kappa = 2$.

$\{h(y) \stackrel{?}{=} y + x\}||\{(x,\ 0), (y,\ 0)\}||\emptyset \stackrel{FBS}{\Rightarrow}$
$\{v \stackrel{?}{=} h(y),\ v \stackrel{?}{=} y + x\}||\{(x,\ 0), (y,\ 0), (v,\ 0)\}||\emptyset \stackrel{Uh}{\Rightarrow}$
$\{v \stackrel{?}{=} h(y),\ v \stackrel{?}{=} y + x\}||\{(x,\ 0), (y,\ 1), (v,\ 0)\}||\emptyset \stackrel{Splitting}{\Rightarrow}$
$\{v \stackrel{?}{=} h(y),\ y \stackrel{?}{=} v_{11} + v_{12},\ y \stackrel{?}{=} h(v_{11}),\ x \stackrel{?}{=} h(v_{12})||\{(x,\ 0), (y,\ 1), (v,\ 0), (v_{11},\ 0), (v_{12},\ 0)\}||\emptyset \stackrel{Uh^+}{\Rightarrow}$
$\{v \stackrel{?}{=} h(y),\ y \stackrel{?}{=} v_{11} + v_{12},\ y \stackrel{?}{=} h(v_{11}),\ x \stackrel{?}{=} h(v_{12})||\{(x,\ 0), (y,\ 1), (v,\ 0), (v_{11},\ 2), (v_{12},\ 1)\}||\emptyset \stackrel{Splitting}{\Rightarrow}$
$\{v \stackrel{?}{=} h(y),\ v_{11} \stackrel{?}{=} v_{13} + v_{14},\ v_{11} \stackrel{?}{=} h(v_{13}),\ v_{12} \stackrel{?}{=} h(v_{14}),\ y \stackrel{?}{=} h(v_{11}),\ x \stackrel{?}{=} h(v_{12})||$
$\{(x,\ 0), (y,\ 1), (v,\ 0), (v_{11},\ 2), (v_{12},\ 1), (v_{13},\ 0), (v_{14},\ 0)\}||\emptyset \stackrel{Uh^+}{\Rightarrow}$
$\{v \stackrel{?}{=} h(y),\ v_{11} \stackrel{?}{=} v_{13} + v_{14},\ v_{11} \stackrel{?}{=} h(v_{13}),\ v_{12} \stackrel{?}{=} h(v_{14}),\ y \stackrel{?}{=} h(v_{11}),\ x \stackrel{?}{=} h(v_{12})||$
$\{(x,\ 0), (y,\ 1), (v,\ 0), (v_{11},\ 2), (v_{12},\ 1), (v_{13},\ 3), (v_{14},\ 2)\}||\emptyset \stackrel{BC}{\Rightarrow}$ *Fail.*

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

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

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

$$\frac{\{t \stackrel{?}{=} x\} \cup \Gamma || \triangle || \sigma}{\{x \stackrel{?}{=} t\} \cup \Gamma || \triangle || \sigma} \qquad \text{If } t \text{ is not a variable}$$

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 a few notations which will be used in the subsequent sections. For two set triples, $\Gamma || \triangle || \sigma$ and $\Gamma' || \triangle' || \sigma'$,

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

As we notice, AC unification divides $\Gamma || \triangle || \sigma$ into the finite number of branches $\Gamma_1 || \triangle_1 || \sigma_1$ and so on $\Gamma_n || \triangle_n || \sigma_n$. Hence, for a triple $\Gamma || \triangle || \sigma$, after applying some inference rules, the result is a disjunction of set triples $\bigvee_i (\Gamma_i || \triangle_i || \sigma_i)$. Accordingly, we introduce the following notation:

- $\Gamma || \triangle || \sigma \Longrightarrow_{\mathfrak{I}_{ACh}} \bigvee_i (\Gamma_i || \triangle_i || \sigma_i)$, where $\bigvee_i (\Gamma_i || \triangle_i || \sigma_i)$ is a disjunction of triples, which means that the set triple $\Gamma || \triangle || \sigma$ becomes $\bigvee_i (\Gamma_i || \triangle_i || \sigma_i)$ with an application of a rule once.
- $\Gamma || \triangle || \sigma \Longrightarrow^{+}_{\mathfrak{I}_{ACh}} \bigvee_i (\Gamma_i || \triangle_i || \sigma_i)$ which represents that $\Gamma || \triangle || \sigma$ becomes $\bigvee_i (\Gamma_i || \triangle_i || \sigma_i)$ after applying some inference rules once or more than once.
- $\Gamma || \triangle || \sigma \Longrightarrow^{*}_{\mathfrak{I}_{ACh}} \bigvee_i (\Gamma_i || \triangle_i || \sigma_i)$ which indicates that $\Gamma || \triangle || \sigma$ becomes $\bigvee_i (\Gamma_i || \triangle_i || \sigma_i)$ after applying some inference rules zero or more times.

4.1.1 Flattening

Here we prove the termination of the flattening procedure.

Consider a multi-set $\mathcal{F}(\Gamma)$, where each element of it is the number of function symbols of an equation in Γ. In other words, for every equation E in the unification problem Γ, there is a number k, the total number of function symbols in E, in the multi-set $\mathcal{F}(\Gamma)$. We define a measure of $\Gamma || \triangle || \sigma$ as the multi-set ordering $\mathcal{F}(\Gamma)_{mul}$ on $\mathcal{F}(\Gamma)$. Note that $\mathcal{F}(\Gamma)_{mul}$ is well-founded since $\mathcal{F}(\Gamma)$ is a well-ordered set.

Lemma 1. *Let* $\Gamma || \triangle || \sigma$ *and* $\Gamma' || \triangle' || \sigma'$ *be two triple sets such that* $\Gamma || \triangle || \sigma \stackrel{Flattening}{\Rightarrow} \Gamma' || \triangle' || \sigma'$. *Then,* $\mathcal{F}(\Gamma)_{mul} > \mathcal{F}(\Gamma')_{mul}$.

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

Algorithm 1 Flattening

Input:

— An equation set Γ.

Output:

— Equation set Γ' where each equation of it is in the *flattened* from (see Section 2.4).

Repeat until none of the flattening rules applied

1 Apply Flatten Both Sides
2 Apply Flatten Left +
3 Apply Flatten Right +
4 Apply Flatten Under h

End

Proof. We have to prove that $\mathcal{F}(\Gamma)_{mul}$ is reducing in all the cases.

Flatten Both Sides: Here the equation $\{t_1 \stackrel{?}{=} t_2\}$ is replaced by $\{v \stackrel{?}{=} t_1,\ v \stackrel{?}{=} t_2\}$, where t_1 and t_2 are terms but not variables. Let $k_1 \geq 0$ be the number of function symbols in t_1 and let $k_2 \geq 0$ be the number of function symbols in t_2. Now, the multi-set $\mathcal{F}(\{t_1 \stackrel{?}{=} t_2\}) = \{k_1 + k_2\}$ and the multi-set $\mathcal{F}(\{v \stackrel{?}{=} t_1,\ v \stackrel{?}{=} t_2\}) = \{k_1,\ k_2\}$. Hence, $\mathcal{F}(\Gamma)_{mul}$ is reduced since the number $(k_1 + k_2)$ is replaced by two smaller numbers k_1, and k_2.

Flatten Left +: The equation $\{t \stackrel{?}{=} t_1 + t_2\}$ is replaced by $\{t \stackrel{?}{=} v + t_2,\ v \stackrel{?}{=} t_1\}$, where t_1 is not a variable. Let k_1 be the number of function symbols in t, k_2 be the number of function symbols in t_1, and k_3 be the number of function symbols in t_2. Then $k_2 \geq 1$ as t_1 is not a variable. The set $\mathcal{F}(\{t \stackrel{?}{=} t_1 + t_2\}) = k_1 + k_2 + k_3 + 1$. But the set $\mathcal{F}(\{t \stackrel{?}{=} v + t_2,\ v \stackrel{?}{=} t_1\}) = \{k_1 + k_3,\ k_2\}$.
Hence, $\mathcal{F}(\Gamma)_{mul} > \mathcal{F}(\Gamma')_{mul}$.

Flatten Right +: In this case, the equation $\{t \stackrel{?}{=} t_1 + t_2\}$ is replaced by $\{t \stackrel{?}{=} t_1 + v,\ v \stackrel{?}{=} t_2\}$, where t_2 is not a variable. The argument is same as above except t_2 is not a variable instead of t_1.

Flatten Under h: The equation $\{t \stackrel{?}{=} h(t_1)\}$ is replaced by $\{t \stackrel{?}{=} h(v),\ v \stackrel{?}{=} t_1\}$, where t_1 is not a variable. Let k_1 be the number of function symbols in t and let $k_2 \geq 1$ be the number of number of function symbols in t_1. Then the multi-set $\mathcal{F}(\{t \stackrel{?}{=} h(t_1)\}) = k_1 + k_2 + 1$ since there is h symbol on top of t_1, where the multi-set $\mathcal{F}(\{t \stackrel{?}{=} h(v),\ v \stackrel{?}{=} t_1\}) = \{k_1 + 1,\ k_2\}$.
Hence, $\mathcal{F}(\{t \stackrel{?}{=} h(t_1)\})_{mul} > \mathcal{F}(\{t \stackrel{?}{=} h(v),\ v \stackrel{?}{=} t_1\})_{mul}$. □

4.1.2 AChUnify

Here we show termination of the AChUnify (see Algorithm 2). First, we define a measure of $\Gamma||\triangle||\sigma$ for proving termination of Algorithm 2.

— 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 bound given by the user. 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 k be the number of applications of the AC unification.

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

Algorithm 2 AChUnify

Input:

— An equation set Γ, a bound $\kappa \in \mathbb{N}$, an empty set σ, and an empty h-depth set $\triangle$.

Output:

— A complete set of κ-bounded *ACh* unifiers $\{\sigma_1, \ldots, \sigma_n\}$ or $\perp$ indicating that the problem has no solution.

Begin

1 Apply flattening (Algorithm 1) on Γ.
2 **Repeat**
(Apply VE1 exhaustively after each of the following rule applications)

(a) Apply *Trivial* exhaustively to eliminate equations of the form $t \stackrel{?}{=} t$.
(b) Apply the *OC*, i.e., **If** any variable on the left side occurs on the right **then** return $\perp$.
(c) Apply the *BC*, i.e., **if** $MaxVal(\triangle) > \kappa$ **then** return $\perp$.
(d) **If** at least one of the h-depth updation rules, Uh, UL, or UR, is applicable **then** apply the rule and go to (c) **else** go to next step.
(e) Apply the *Orient* exhaustively.
(f) **If** *Splitting* is applicable **then** apply the rule and go to (a).
(g) Apply the *Clash*, i.e., **If** the top symbols of the left and right sides of an equation do not match **then** return $\perp$.
(h) **If** Decomposition applicable **then** apply the rule and go to (a).
(i) **If** there is at least one variable x that occurs left side in at least two equations, $x \stackrel{?}{=} y_1 + \cdots + y_n$ and $x \stackrel{?}{=} z_1 + \cdots + z_n$, **then** apply the *AC Unification* and then go to (d) **else** go to next step.
(j) Apply VE2 exhaustively and return the output.

End

End

— Let p be the 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, that is, $x = x_1 + \cdots + x_n$.

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

$$\mathbb{M}_{\mathfrak{I}_{ACh}}(\Gamma, \triangle, \sigma) = (\kappa - k, n, |Sym(\Gamma)|, p, m, |\Gamma|, \overline{h_d}(\Gamma)).$$

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

Lemma 2. *Let $\Gamma||\triangle||\sigma$ be a set triple, and κ be a natural number (bound) given as an input to the algorithm. Then, the maximum number of times the AC unification applied is κ.*

Proof. The only time the AC unification is invoked on the problem is when there is at least one non-solved variable in the problem. A variable x occurs at least in two equations as $x \stackrel{?}{=} y_1 + \cdots +$

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

y_n and $x \stackrel{?}{=} z_1 + \cdots + z_m$. On each application of the AC unification, the lowest depth of non-solved variables get solved, and there is no other rule that makes these variables non-solved again. Hence, the maximum number of times the AC unification could be applied is the κ. □

Lemma 3. *Let $\Gamma||\triangle||\sigma$ and $\Gamma'||\triangle'||\sigma'$ be two set triples, where Γ and Γ' are in flattened form such that $\Gamma||\triangle||\sigma \Rightarrow_{\mathfrak{I}_{ACh}} \Gamma'||\triangle'||\sigma'$. Then $\mathbb{M}_{\mathfrak{I}_{ACh}}(\Gamma, \triangle, \sigma) > \mathbb{M}_{\mathfrak{I}_{ACh}}(\Gamma', \triangle', \sigma')$.*

Proof. We prove that on each of the inference rules decrease the measure.

Trivial. The cardinality of Γ, $|\Gamma|$, decreases while other components of the measure either stay the same or decrease. Hence, $\mathbb{M}_{\mathfrak{I}_{ACh}}(\Gamma, \triangle, \sigma) > \mathbb{M}_{\mathfrak{I}_{ACh}}(\Gamma', \triangle', \sigma')$.

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

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, $\kappa\text{-}n > \kappa\text{-}(n+1)$. Which means that $\overline{h_d}(\Gamma)$ decreases while the other components stay the same. Hence, $\mathbb{M}_{\mathfrak{I}_{ACh}}(\Gamma, \triangle, \sigma) > \mathbb{M}_{\mathfrak{I}_{ACh}}(\Gamma', \triangle', \sigma')$.

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

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.

AC Unification. The measure decreases as the first component of it does. □

Theorem 4 (Termination). *For any set triple $\Gamma||\triangle||\sigma$, there is a set triple $\Gamma'||\triangle'||\sigma'$ such that $\Gamma||\triangle||\sigma \Rightarrow^*_{\mathfrak{I}_{ACh}} \bigvee_i (\Gamma_i||\triangle_i||\sigma_i)$ and none of the rules $\mathfrak{I}_{ACh}$ can be applied on $\Gamma_i||\triangle_i||\sigma_i$.*

Proof. By induction on Lemma 3 this theorem can be proved. □

4.2 Soundness

In this section, we show that our inference system $\mathfrak{I}_{ACh}$ is truth-preserving.

Lemma 5. *Let $\Gamma||\triangle||\sigma$ and $\Gamma'||\triangle'||\sigma'$ be two set triples such that $\Gamma||\triangle||\sigma \Rightarrow_{\mathfrak{I}_{ACh}} \Gamma'||\triangle'||\sigma'$ via all the rules of $\mathfrak{I}_{ACh}$ except AC unification. Let θ be a substitution such that $\theta \models \Gamma'||\triangle'||\sigma'$. Then $\theta \models \Gamma||\triangle||\sigma$.*

Proof. **Trivial.** It is trivially true.

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

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

By the homomorphism theory, we write that $h(v_1\theta) + \cdots + h(v_n\theta) \stackrel{?}{=} h(v_1\theta + \cdots + v_n\theta)$. Then $h(v_1\theta + \cdots + v_n\theta) \stackrel{?}{=} h(y\theta) \stackrel{?}{=} w\theta$. Hence, θ satisfies $w \stackrel{?}{=} x_1 + \cdots + x_n$.

Variable Elimination.
VE1. Assume that $\theta \models \Gamma\{x \mapsto y\}||\triangle||\sigma\{x \mapsto y\} \cup \{x \mapsto y\}$. 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 $\{x \stackrel{?}{=} y\}$, Γ, 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 $\theta \models \{x \stackrel{?}{=} f(t_1, \ldots, t_n),\ s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\} \cup \Gamma||\triangle||\sigma$. This means that θ satisfies $\{x \stackrel{?}{=} f(t_1, \ldots, t_n),\ s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\} \cup \Gamma$. Now we have to prove that θ satisfies $\{x \stackrel{?}{=} f(s_1, \ldots, s_n), x \stackrel{?}{=} f(t_1, \ldots, t_n)\} \cup \Gamma$. Given that θ satisfies $x \stackrel{?}{=} f(t_1, \ldots, t_n)$ and it is enough to show that θ also satisfies $x \stackrel{?}{=} f(s_1, \ldots, s_n)$. We write $x\theta \stackrel{?}{=} f(t_1, \ldots, t_n)\theta \stackrel{?}{=} f(t_1\theta, t_2\theta, \ldots, t_n\theta) \stackrel{?}{=} f(s_1\theta, s_2\theta, \ldots, s_n\theta)$ since $s_1\theta \stackrel{?}{=} t_1\theta, \ldots, s_n\theta \stackrel{?}{=} t_n\theta$. So, θ satisfies $x \stackrel{?}{=} f(t_1, \ldots, t_n)$ and $x \stackrel{?}{=} f(s_1, \ldots, s_n)$. Hence, $\theta \models \{x \stackrel{?}{=} f(s_1, \ldots, s_n),\ x \stackrel{?}{=} f(t_1, \ldots, t_n)\}$. □

Lemma 6. *Let $\Gamma||\triangle||\sigma$ and $\Gamma'||\triangle'||\sigma'$ be two set triples such that $\Gamma||\triangle||\sigma \Longrightarrow_{\mathfrak{I}_{ACh}} \bigvee_i (\Gamma_i||\triangle_i||\sigma_i)$ via AC unification. Let θ be a substitution such that $\theta \models \Gamma_i||\triangle_i||\sigma_i$. Then $\theta \models \Gamma||\triangle||\sigma$.*

Proof. **AC Unification.**

$$\frac{\Psi \cup \Gamma||\triangle||\sigma}{GetEqs(\theta_1) \cup \Gamma||\triangle||\sigma \vee \ldots \vee GetEqs(\theta_n) \cup \Gamma||\triangle||\sigma}$$

Given that $\theta \models GetEqs(\theta_1) \cup \Gamma||\triangle||\sigma \vee \ldots \vee GetEqs(\theta_n) \cup \Gamma||\triangle||\sigma$. This means that θ satisfies $GetEqs(\theta_1) \cup \Gamma||\triangle||\sigma, \ldots, GetEqs(\theta_n) \cup \Gamma||\triangle||\sigma$. Which implies that θ also satisfies Ψ. □

By combining Lemmas 5 and 6, we have

Lemma 7. *Let $\Gamma||\triangle||\sigma$ and $\Gamma'||\triangle'||\sigma'$ be two set triples such that $\Gamma||\triangle||\sigma \Longrightarrow_{\mathfrak{I}_{ACh}} \bigvee_i (\Gamma_i||\triangle_i||\sigma_i)$. Let θ be a substitution such that $\theta \models \Gamma_i||\triangle_i||\sigma_i$. Then $\theta \models \Gamma||\triangle||\sigma$.*

Then by induction on Lemma 7, we get the following theorem:

Theorem 8. *Let $\Gamma||\triangle||\sigma$ and $\Gamma'||\triangle'||\sigma'$ be two set triples such that $\Gamma||\triangle||\sigma \Longrightarrow^*_{\mathfrak{I}_{ACh}} \bigvee_i (\Gamma_i||\triangle_i||\sigma_i)$. Let θ be a substitution such that $\theta \models \Gamma_i||\triangle_i||\sigma_i$. Then $\theta \models \Gamma||\triangle||\sigma$.*

We have the following corollary from Theorem 8:

Theorem 9 (Soundness). *Let σ be a set of equations. Suppose that we get $\bigvee_i (\Gamma_i||\triangle_i||\sigma_i)$ after exhaustively applying the rules from $\mathfrak{I}_{ACh}$ to $\Gamma||\triangle||\sigma$, that is, $\Gamma||\triangle||\sigma \Longrightarrow^*_{\mathfrak{I}_{ACh}} \bigvee_i (\Gamma_i||\triangle_i||\sigma_i)$,*

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

where for each i, no rules applicable to $\Gamma_i||\triangle_i||\sigma_i$. *Let* $\Sigma = \{\sigma_i \mid \Gamma_i = \emptyset\}$. *Then any member of* Σ *is an ACh-unifier of* Γ.

4.3 Completeness

Here, we prove that our inference system never loses any solution.

Lemma 10. *Let* $\Gamma||\triangle||\sigma$ *be a set triple which is not in solved form, and* θ *be a substitution such that* $\theta \models \Gamma||\triangle||\sigma$. *Then there exists an inference* $\Gamma||\triangle||\sigma \Rightarrow_{\mathfrak{I}_{ACh}} \bigvee_i (\Gamma_i||\triangle_i||\sigma_i)$ *and an i and* θ_0 *whose domain is the variables in* $Var(\Gamma_i) \setminus Var(\Gamma)$, *such that* $\theta\theta_0 \models \Gamma_i||\triangle_i||\sigma_i$.

Proof. Assume that $\Gamma||\triangle||\sigma$ be a set triple that is not in solved form and θ be a substitution such that $\theta \models \Gamma||\triangle||\sigma$. Now, we consider various possible forms of Γ and prove that there is an inference rule that can be applied and the solution θ can be extended.

1. $\{x \stackrel{?}{=} f(t_1, \ldots, t_n),\ x \stackrel{?}{=} f(s_1, \ldots, s_n)\} \cup \Gamma'||\triangle||\sigma, f \neq +$.
   Here, we can apply decomposition to get $\{x \stackrel{?}{=} f(t_1, \ldots, t_n),\ s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\} \cup \Gamma'||\triangle||\sigma$.
2. $\{x \stackrel{?}{=} h(y),\ x \stackrel{?}{=} x_1 + \cdots + x_n\} \cup \Gamma'||\triangle||\sigma$.
   In this case, we can apply splitting to transform the set triple to $\{x \stackrel{?}{=} h(y), y \stackrel{?}{=} v_1 + \cdots + v_n, x_1 \stackrel{?}{=} h(v_1), \ldots, x_n \stackrel{?}{=} h(v_n)\} \cup \Gamma'||\triangle||\sigma$. Since θ is a solution of $\{x \stackrel{?}{=} h(y),\ x \stackrel{?}{=} x_1 + \cdots + x_n\} \cup \Gamma'||\triangle||\sigma$, we have that θ satisfies $h(y)\theta \stackrel{?}{=} (x_1 + \cdots + x_n)\theta$. Now, considering the rewrite system R1 which has the rewrite rule $h(x + y) \to h(x) + h(y)$ and setting $y \stackrel{?}{=} v_1 + \cdots + v_n$, we have $(x_1 + \cdots + x_n)\theta \stackrel{?}{=} (h(v_1) + \cdots + h(v_n))\theta$. Hence, θ can be extended to a substitution giving values for the new variables.
3. $\{x \stackrel{?}{=} x_1 + \cdots + x_n,\ x \stackrel{?}{=} y_1 + \cdots + y_m\} \cup \Gamma'||\triangle||\sigma$.
   We apply AC unification on $\{x \stackrel{?}{=} x_1 + \cdots + x_n,\ x \stackrel{?}{=} y_1 + \cdots + y_m\}$ and get a complete set of AC unifiers and one of them corresponds to the substitution θ. The substitution θ can be extended to a substitution giving values for new variables.
4. $\{t \stackrel{?}{=} x)\} \cup \Gamma'||\triangle||\sigma$.
   In this case, we apply the orient, and θ becomes the solution to this problem.
5. $\{t \stackrel{?}{=} t)\} \cup \Gamma'||\triangle||\sigma$.
   Here, we apply the trivial rule to eliminate the equation $t \stackrel{?}{=} t$. Of course, θ becomes solution of $\Gamma'||\triangle||\sigma$.
6. $\{x \stackrel{?}{=} y)\} \cup \Gamma'||\triangle||\sigma$.
   We apply VE1 and θ becomes the solution to the resulting problem.
7. $\{x \stackrel{?}{=} t)\} \cup \Gamma'||\triangle||\sigma$, where $t \notin \mathcal{V}$, x does not occur in t, and none of the other conditions apply.
   Here, we apply a sequence of VE2 steps until the set of equations empty. □

By induction on Lemma 10, we get

Theorem 11. *Let* $\Gamma||\triangle||\sigma$ *be a set triple which is not in solved form, and* θ *be a substitution such that* $\theta \models \Gamma||\triangle||\sigma$. *Then there exists a sequence of inferences* $\Gamma||\triangle||\sigma \Longrightarrow^{+}_{\mathfrak{I}_{ACh}} \bigvee_i (\Gamma_i||\triangle_i||\sigma_i)$ *and an i and* θ_0 *whose domain is the variables in* $Var(\Gamma_i) \setminus Var(\Gamma)$, *such that* $\theta\theta_0 \models \Gamma_i||\triangle_i||\sigma_i$.

We get the following corollary from the above theorem:

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

Table 1. Tested results with bounded ACh-unification algorithm

Unification problem	Real time	Solution	# Sol.	Bound
$\{h(y) \stackrel{?}{=} y + x\}$	674 ms	$\perp$	0	10
$\{h(y) \stackrel{?}{=} y + x\}$	15 880 ms	$\perp$	0	20
$\{h(y) \stackrel{?}{=} x_1 + x_2\}$	5 ms	Yes	1	10
$\{h(h(x)) \stackrel{?}{=} h(h(y))\}$	2 ms	Yes	1	10
$\{x + y_1 \stackrel{?}{=} x + y_2\}$	3 ms	Yes	1	10
$\{v \stackrel{?}{=} x + y, v \stackrel{?}{=} w + z, s \stackrel{?}{=} h(t)\}$	46 ms	Yes	10	10
$\{v \stackrel{?}{=} x_1 + x_2, v \stackrel{?}{=} x_3 + x_4, x_1 \stackrel{?}{=} h(y), x_2 \stackrel{?}{=} h(y)\}$	100 ms	Yes	6	10
$\{h(h(x)) \stackrel{?}{=} v + w + y + z\}$	224 ms	Yes	1	10
$\{v \stackrel{?}{=} (h(x) + y), v \stackrel{?}{=} w + z\}$	55 ms	Yes	7	10
$\{f(x, y) \stackrel{?}{=} h(x_1)\}$	0 ms	$\perp$	0	10
$\{f(x_1, y_1) \stackrel{?}{=} f(x_2, y_2)\}$	1 ms	Yes	1	10
$\{v \stackrel{?}{=} x_1 + x_2, v \stackrel{?}{=} x_3 + x_4\}$	17 ms	Yes	7	10
$\{f(x_1, y_1) \stackrel{?}{=} g(x_2, y_2)\}$	0 ms	$\perp$	0	10
$\{h(y) \stackrel{?}{=} x, y \stackrel{?}{=} h(x)\}$	0 ms	$\perp$	0	10

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

5. Implementation

We have implemented the algorithm in the Maude programming language.[1] The implementation of this inference system is available.[2] 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 GB 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), 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. (2010, 2012); Escobar et al. (2011). 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.

Acknowledgements. We thank anonymous reviewers who have provided useful comments. Part of the work of this paper has been done during the first author's stay at the University of Missouri, partially supported by NSF Grant CNS 1314338, and Clarkson University.

https://doi.org/10.1017/S0960129520000183 Published online by Cambridge University Press

Notes

1 http://maude.cs.illinois.edu/w/index.php/The_Maude_System.
2 https://github.com/ajayeeralla/Unification_ACh.

References

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

Anantharaman, S., Lin, H., Lynch, C., Narendran, P. and Rusinowitch, M. (2012). Unification modulo homomorphic encryption. In: *Journal of Automated Reasoning*, Springer, 135–158.

Baader, F. and Nipkow, T. (1998). *Term Rewriting and All that*, Cambridge University Press, Cambridge, UK.

Baader, F. and Snyder, W. (2001). Unification theory. In: *Handbook of Automated Reasoning*, Elsevier, 447–533.

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

Escobar, S., Kapur, D., Lynch, C., Meadows, C., Meseguer, J., Narendran, P. and Sasse, R. (2011). 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*, ACM, 65–76.

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

Fages, F. (1984). Associative-commutative unification. In: *7th International Conference on Automated Deduction*, Springer, 194–208.

Kapur, D., Narendran, P. and Wang, L. (2003). An E-unification algorithm for analyzing protocols that use modular exponentiation. In: *Rewriting Techniques and Applications*, Springer, 165–179.

Kremer, S., Ryan, M. and Smyth, B. (2010). Election verifiability in electronic voting protocols. In: *Computer Security – ESORICS*, Springer, 389–404.

Liu, Z. and Lynch, C. (2011). Efficient general unification for XOR with homomorphism. In: *23rd International Conference on Automated Deduction*, CADE-23, Springer, 407–421.

Liu, Z. and Lynch, C. (2014). Efficient general AGH-unification. In: *Information and Computation*, vol. 238, Elsevier, 128–156.

Marshall, A. M., Meadows, C. A. and Narendran, P. (2015). On unification modulo one-sided distributivity: algorithms, variants and asymmetry. *Logical Methods in Computer Science* **11** (2) 1–39.

Narendran, P. (1996). Solving linear equations over polynomial semirings. In: *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society, 466–472.

Schmidt-Schauß, M. (1998). A decision algorithm for distributive unification. In: *Journal of Theoretical Computer Science*, Elsevier, 111–148.

Tidén, E. and Arnborg, S. (1987). Unification problems with one-sided distributivity. In: *Journal of Symbolic Computation*, Springer, 183–202.

Cite this article: Eeralla AK and Lynch C (2020). Bounded ACh unification. *Mathematical Structures in Computer Science* **30**, 664–682. https://doi.org/10.1017/S0960129520000183