Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Um estudo em unificação e desunificação módulo (A Study on Unification and DisUnification Modulo)

Mehwish Arshid

Dissertação apresentado como requisito parcial para
conclusão do Mestrado em Informática

Orientador
Prof. Dr. Mauricio Ayala Rincón

Brasília
2020

Ficha Catalográfica de Teses e Dissertações

Está página existe apenas para indicar onde a ficha catalográfica gerada para dissertações de mestrado e teses de doutorado defendidas na UnB. A Biblioteca Central é responsável pela ficha, mais informações nos sítios:

http://www.bce.unb.br
http://www.bce.unb.br/elaboracao-de-fichas-catalograficas-de-teses-e-dissertacoes

Esta página não deve ser inclusa na versão final do texto.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Um estudo em unificação e desunificação módulo (A Study on Unification and DisUnification Modulo)

Mehwish Arshid

Dissertação apresentado como requisito parcial para
conclusão do Mestrado em Informática

Prof. Dr. Mauricio Ayala Rincón (Orientador)
CIC/UnB

Prof. Dr. Vander Ramos Alves
Universidade de Brasília

Prof.a Dr.a Daniele Nantes Sobrinho
Universidade de Brasília

Prof. Dr. Edward Hermann Haeusler
PUC-Rio

Prof.a Dr.a Genaina Nunes Rodrigues
Coordenador do Programa de Pós-graduação em Informática

Brasília, 30 de Setembro de 2020

Acknowledgment

All praises for Almighty Allah the most compassionate, the most beneficent, and ever merciful, who gives me the power to do, the sight to observe and mind to think and judge. Peace and blessings of Almighty Allah be upon His Last Prophet Hazrat Muhammad (P.B.U.H) who exhorted his followers to seek knowledge from cradle to grave.

I want to express my deepest gratitude to my supervisor, Prof. Dr. Mauricio Ayala Rincon, for his kind supervision, practical suggestions, consistent encouragement, valuable advice, sustained support, friendly behavior, and dynamic supervision which enabled me to complete my dissertation that would not have been possible without his support.

I am grateful to CAPES for financial support.

I am also grateful to my Husband Jamal Nasir for his kind support and countless help during my study, and my both daughters Ifra Nasir and Isbah Nasir for unconditional love and patience.

Furthermore, I would like to extend my sincere thanks to my lab fellows at Laforce especially to Gabriel Silva for his helpful suggestions, and others Bruno de Assis Delboni, Thiago Mendonca Ferreira Ramos, Ariane Alves Almeida, Kaliana Dias de Freitas, Lucas Angelo Silveira, Daniel Saad Nogueira Nunes for their co-operation and sympathetic attitude during my study.

Finally, I would like to express my deepest thanks to my family for continuous support and love during these years. I would like to express my special thanks to my dear parents especially to my father Arshid Mehmood for his love, encouragement, and support in every part of my life. I would like to express my special thanks to my dear parents-in-law especially to my father-in-law Muhammad Tariq for his trust in me and help during my study. May Almighty Allah shower His blessings and prosperity on all those who assisted me in any way during my work.

Resumo

Estuda-se a comparação entre unificação assimétrica e desunificação módulo teorias equacionais em relação às suas complexidades, como desenvolvida por Ravishankar, Narendran e Gero. A unificação assimétrica é um tipo de unificação equacional em que as soluções devem fornecer o lado direito dos problemas apresentados na forma normal. E a desunificação é resolver problemas com equações e "disequações" em relação à uma teoria equacional dada. As soluções para os problemas de desunificação são substituições que tornam os dois termos de cada equação iguais, mas os dois termos de cada "disequação" diferentes. Unificação e desunificação equacional foram comparadas por os autores mencionados com relação as suas complexidades de tempo para duas teorias equacionais: a primeira associativa (A), comutativa (C), com unidade (U) e nilpotente (N), como ($ACUN$) e a segunda com tais propriedades, mas adicionando um homomorfismo (h), como ($ACUNh$), mostrando que desunificação pode ser resolvida em tempo polinomial enquanto unificação assimétrica é NP-difícil para ambas as teorias equacionais.

Além disso, foi estudada a abordagem introduzidas por Zhiqiang Liu, em sua dissertação de doutorado, para converter os unificadores módulo $ACUN$ em assimétricos, com símbolos de função não interpretados, usando as regras de inferência.

Para a teoria associativa comutativa com homomorfismo (ACh), estudou-se a prova de que unificação módulo ACh é indecidível, assim como o algoritmo de semi-decisão, recentemente introduzido por Ajay Kumar Eeralla e Christopher Lynch, que apresenta um conjunto de regras de inferência para resolver o problema com limitações.

Palavras-chave: Unificação, Desunificação, Unificação Assimétrica, Teorias Equacionais Módulo Associatividade-Comutatividade

Abstract

Comparisons between asymmetric unification and disunification modulo AC concerning their complexities, as developed by Ravishankar, Narendran and Gero are studied. Asymmetric unification is a type of equational unification problem in which the solutions must give as right-hand sides of the input problem, normal forms regarding some rewriting system. And disunification problems require solving equations and "disequations" for a given equational theory. Solutions to the disunification problems are substitutions that make the two terms of each equation equal, but the two terms of each "disequation" different. These authors compared the complexity of the unification and disunification problems for two equational theories. The properties of the first equational theory are associativity (A), commutativity (C), the existence of unity (U), and nilpotence (N), abbreviated as ACUN. And, the second equational theory has the same properties but adds a homomorphism (h), for short, ACUNh. For such equational theories, details of the proof that disunification can be solved in polynomial time while the asymmetric unification is NP-hard have been studied. Besides, the approach for converting ACUN unifiers to asymmetric ones, with uninterpreted function symbols using the inference rules introduced by Zhiqiang Liu, in his Ph.D. dissertation, was studied. Narendran's proof of the undecidability of the unification problem modulo the associative commutative theory with homomorphism ACh is studied. Also, the semi-decision algorithm, recently introduced by Ajay Kumar Eeralla and Christopher Lynch, is studied, which presents a set of inference rules for solving a bounded version of ACh unification.

Keywords: Unification, Disunification, Asymmetric Unification, Equational Theories Modulo Associativity-Commutativity

Contents

Chapter 1

Introduction

1.1 Motivation

Computers use a set of rules called protocol to communicate with each other across a network, and the cryptographic protocols are used to provide security and called security protocols. Cryptographic protocols need the properties of secrecy and authentication [1]. There are many kinds of algorithms that are used in cryptographic protocols in [2].

In their seminal work [3], Dolev and Yao use formal algebraic tools to analyze cryptographic protocols [1]. At the beginning of the analysis, cryptographic primitives were taken as black boxes. The other method to analyze cryptographic protocol is to consider the protocol as an algebraic system, similar to the way in which Dolev and Yao model protocols. This method is known as the free algebra method. In this method of analysis, the term that is representing the received message will be compared against the term that is representing a sent message, if the corresponding terms are the same then these messages will be the same. Syntactic unification is used to compare them. But this approach ignores the fact that there may be some algebraic properties about the terms representing the messages. To solve this problem, modern approaches are taken into account unification modulo the algebraic properties.

Several examples of such kind of protocol analysis can be found in the literature, from which we discuss only three. In [4] Lowe analyzed the Needham-Schroeder Public-Key Protocol using the so called *Failures Divergences Refinement Checker* (FDR), a model checker for Communicating Sequential Process (CSP) to discover intruder attacks upon a given protocol and to make sure that the communication actors are talking to each other but not to an intruder, and so to guarantee the security of the protocol. Lowe modeled a general intruder who can interact with the protocol to observe and intercept messages to learn information and then use this information to introduce fake messages into the system. Lowe shown the existence of an attack to the Needham-Schroeder Public-Key

Protocol, and his method has been applied to analyze different protocols. In [5] Ayala-Rincón et al presented an algorithm to decide the intruder deduction problem (IDP) for associative commutative *(AC)* equational theories, using the initial knowledge of the intruder and modeling the algebraic capabilities of an intruder by a deduction system. The IDP consists in detecting whether a passive eavesdropper can obtain a certain information from the messages observed. In [6] Escobar et al discussed formal analysis of cryptographic protocols solving (equality and) disequality constraints, which also arise in cryptographic protocol analysis, and are used to reduce the size of the search space by protocol analysis tools.

The algebraic properties that express as an equational theory E, and to find an E-unification algorithm is a way to reduce the search space and generate a minimum complete set of unifiers.

Unification procedures for theories such as *AC, ACh, ACU, ACUh, ACUN* and *ACUNh* are essential for cryptographic protocol analysis. But since all these problems are not decidable, in many cases semi-decision algorithms are required. They must be efficient and allow the combination with uninterpreted function symbols. For the case of the theories *ACUN* and *ACUNh*, when uninterpreted function symbols occur, the complete set of unifiers is not always a singleton, but it is finite. Also, it is important that the unification algorithm creates a complete set of unifiers that is as small as possible. For the case of the *ACh*-unification problem, which is undecidable, it is possible to bound the set of computed solutions. To build an equational unification (semi-decision) algorithm for these theories that is efficient and compute small (complete) set of unifiers is an important application of an equational unification.

1.2 Equational problems under consideration

In this document, we discuss the time complexity of (ground) asymmetric unification and (ground) disunification for the theory of *ACUN* and *ACUNh* introduced by Veena Ravishankar, Paliath Narendran and Kimberly A. Gero in [7]. As the name indicates, the theory of *ACUN* consists of associativity, commutativity, unity and nilpotence. This theory is also called the theory of boolean *XOR* operator. Asymmetric unification is an extension of the equational unification problem. In asymmetric unification, the equational theory is divided into a set of rewriting rules R and the set of equations E, then we call (R, E) decomposition of the equational theory. Sometimes $R \cup E$ can be used to denote the equational theory.

We study for the theory of *ACUN* that the complexity of the asymmetric unification problem is NP-hard, while the disunification problem is solvable in polynomial time, as

proved in [7].

We also study the complexity of ground asymmetric unification and of ground disunification, as given in [7], for the theory of *ACUN* with homomorphism, *ACUNh*. Ground asymmetric unification modulo *ACUNh* is NP-hard, while ground disunification modulo *ACUNh* can be solved in polynomial time.

We also study the algorithm described by Zhiqiang Liu and Christopher Lynch in [8] and [1], called Asymmetric Unification algorithm for the theory of *ACUN* (or *XOR*) with uninterpreted function symbols. This algorithm is used to find asymmetric unifiers from symmetric ones. Indeed, if there exists a symmetric unifier one can check if this unifier is also an asymmetric unifier by using some inference rules.

In addition, we study and discuss Narendran's approach [9], about the undecidability of the unification problems modulo the two equational theories *ACUh* and *ACh*. Narendran considers the problems of linear equations over polynomial semirings and showed that the solvability of linear equations over polynomial semirings is undecidable by using Hilbert's tenth problem. After that, Narendran showed that the solvability of linear equations over polynomial semirings is reducible to the unification problems by applying Nutt's approach published in [10].

In addition, we discuss the algorithm for solving bounded *ACh* unification problems introduced by Ajay Kumar Eeralla and Christopher Lynch in [11]. They bound the application of the homomorphic function h to the terms, and allow only those solutions which satisfy the given bound. To use this theory *ACh* in cryptographic protocols, Ajay Kumar Eeralla uses the bounded version of this theory *ACh* that he shown to be decidable in [11]. They introduced a set of inference rules to solve the bounded *ACh* unification problems.

1.3 Contribution

This work presents the study of all topics mentioned before but adding details on the proofs that were not given in the source papers:

- the complexity of *ACUN* and *ACUNh* asymmetric unification and disunification and ground asymmetric unification and disunification, as given by Ravishankar, Narendran and Gero in [7];
- Zhiqiang Liu and Christopher Lynch's asymmetric unification algorithm, as given in [1];
- undecidability of *ACUh* and *ACh* unification, as given by Narendran in [9];

- Ajay Kumar Eeralla and Christoper Lynch's bounded *ACh* unification algorithm, as given in [11].

Specifically, this work presents details of the proof of the undecidability of the equational theory *ACh*, which is not explicitly given by Narendran in [9]. In his paper, Narendran discussed in detail the undecidability of the equational theory *ACUh* and the adaptation of the undecidability of the equational theory *ACh* by following the proof given for the undecidability of the equational theory *ACUh*. In the proof of the undecidability of the equational theory *ACUh*, he shown that the solvability of the linear equations over polynomial semiring $\mathbb{N}$ is undecidable by considering Hilbert's tenth problem and he illustrated how Hilbert's tenth problem reduces to the problem of solving linear equations over polynomial semiring $\mathbb{N}$. Further, he illustrated the reducibility of such linear equations over the polynomial semiring $\mathbb{N}$ to the ACUh unification problem by Nutt's approach in [10].

In this work, we add to Narendran's proof all details related to the adaptation of the proof of the undecidability of the equational theory *ACh*, which consist of proving that solvability of the linear equations over the structure $\mathbb{N}^+$ is also undecidable from Hilbert's tenth problem, and reducibility of such linear equations over the structure $\mathbb{N}^+$ to the *ACh* unification problem.

1.4 Organization

Here is the organization of the document.

- In Chapter 2, the basic notations are given, which are used in the following chapters and provide enough background for understanding the study. The unification theory and related notions and properties are discussed. Also, term rewriting systems are studied. A section on complexity theory and a section on automata theory are given. The definition of ring and semiring is also given.

- In Chapter 3, we study the complexities of (ground) asymmetric unification and disunification modulo two different equational theories i.e., *ACUN* and *ACUNh*.

- In Chapter 4, we study Zhiqiang Liu and Christopher Lynch's algorithm to solve the asymmetric unification problem for the theory of *ACUN* with uninterpreted function symbols.

- In Chapter 5, we discuss Narendran's proof on the undecidability of the unification problem modulo *ACUh* and *ACh*, and Ajay Kumar and Christoper Lynch's bounded *ACh* unification algorithm.

Chapter 2

Theoretical Referential

2.1 Unification Theory

Unification is an algorithmic procedure that is used to make two symbolic expressions(also called terms) equal with a suitable substitution i.e., by replacing certain sub-expressions (variables) with other expressions [12]. Different occurrences of the same variable in a unification problem must always be replaced by the same term.

In unification problems, it is not enough that we just decide whether two terms s and t are unifiable or not, but also if they are unifiable we have to construct its solutions, i.e. substitutions that identify s and t. In first-order unification, the solution is unique and is called a unifier of both the terms s and t. In general, a unification problem may have infinitely many solutions but it is sufficient to consider the so-called complete set of *most general unifiers*, i.e. a minimal set of unifiers such that every other unifier can be obtained by instantiation of one of the unifiers in the complete set.

Unification as described until now is called syntactic unification of first-order terms. Syntactic means that the terms must be made syntactically equal, whereas the first-order expresses the fact that we do not allow for higher-order variables, i.e. variables for functions. For example, the terms $f(x, a)$ and $g(a, x)$ obviously cannot be made syntactically equal by first-order unification.

2.1.1 Basic Notation

Signature

A finite or countably infinite set of function symbols with some defined arity, where a constant is a function symbol with an arity zero is called signature and usually denoted by Σ [12].

The set of terms over V and Σ is denoted by $T(\Sigma, V)$ where Σ is a set of signature function symbol and V is a countably infinite set of variables. Usually, the terms are denoted by s, t, u and v, whereas the variables are denoted by x, y and z and the function symbols are denoted by f and g and so on. The set of variables that appears in a term t is denoted by $Var(t)$. If this set $Var(t) = \phi$ then the term t is said to be a ground term.

Unification Problem

A unification problem for two terms s and t is represented by $s =^? t$. A unification problem is a problem of solving a finite set of equations between terms, usually denoted by Γ.

$$\Gamma = \{s_1 =^? t_1, s_2 =^? t_2, \ldots, s_n =^? t_n\}$$

if there exist a substituition σ such that $\sigma(s_i) = \sigma(t_i)\ \ i = 1, \ldots, n$. This σ is called the unifier of the unification problem. And if a substitution σ is the unifier of the set of equations Γ, then it is a solution of each equation in Γ.

Substitution

A substitution (also called solution) is usually denoted by σ is a mapping from variables V to terms $T(\Sigma, V)$ i.e. $\sigma : V \to T(\Sigma, V)$.

A substitution σ can be represented by a set of bindings of variables and is written as

$$\sigma = \{x_1 \mapsto t_1, x_2 \mapsto t_2, \ldots, x_n \mapsto t_n\}$$

which means x_i maps to t_i for $i = 1, 2, \ldots, n$.

If a substitution σ maps a variable to itself then it is called Identity substitution and usually denoted by Id. A substitution σ will be equal to some another substitution θ if for every variable x such that $\sigma(x) = \theta(x)$ holds. A substitution σ is more general than substitution θ if there exists a substitution η such that $\theta = \eta \circ \sigma$ (also written as prefix $\sigma\eta$) that is denoted by $\sigma \leq \theta$.

Unifier (Most General Unifier)

If some substitution σ is a solution of two terms s and t i.e., if it satisfies $\sigma(s) = \sigma(t)$ then it is called a unifier or solution of these two terms. It is called most general unifier (m.g.u) if, for every unifier σ of s and t, it satisfies $\sigma \leq \theta$.

Example 2.1.1. *Consider the unification problem:*

$$f(a, x) =^? f(y, z)$$

The signature of this problem is given by $\Sigma = \{f, a\}$*, where* f *is a unary function symbol and* a *is a constant.*

A unifier of this problem is given by the substitution below.

$$\delta = \{x \mapsto a, y \mapsto a, z \mapsto a\}$$

But the m.g.u. is the unifier

$$\sigma = \{x \mapsto z, y \mapsto a\}$$

Indeed, notice that $\gamma\sigma = \delta$ *for* $\gamma = \{z \mapsto a\}$*.*

2.1.2 E-Unification Problem

For a given signature Σ and equational theory E, the unification problem modulo equational theory E is a problem of solving a finite set of equations between terms i.e

$$\Gamma = \{s_1 =_E^? t_1, \ldots, s_n =_E^? t_n\}$$

If $s =_E t$, then we can say that the terms s and t are equal with respect to equational theory E. A unification problem modulo an equational theory for two terms s and t is represented by $s =_E^? t$.

E-unifier

Two terms s and t are unifiable modulo an equational theory E iff there exists a substitution σ such that $\sigma(s) =_E \sigma(t)$; then, such a substitution σ is called E-unifier or E-solution of these terms. An E-unifier of a unification problem modulo an equational theory Γ is a substitution σ such that $\sigma(s_i) =_E \sigma(t_i)$ for $i = 1, 2, \ldots, n$

Example 2.1.2. *Consider the unification problem below modulo commutativity (C).*

$$f(x, y) =_C^? f(a, b)$$

This problem has two solutions:

$$\sigma_1 = \{x \mapsto a, y \mapsto b\}, \text{ and } \sigma_2 = \{x \mapsto b, y \mapsto a\}$$

Notice that in the syntactic case the only possible solution is σ_1*.*

Example 2.1.3. *Consider the unification problem modulo associativity (A).*

$$f(a,x) =^{?}_{A} f(x,a)$$

This problem as an infinite set of solutions:

$$\sigma_1 = \{x \mapsto a\}, \sigma_2 = \mapsto \{f(a,a)\}, \sigma_3 \mapsto \{x \mapsto f(a,f(a,a))\}, \ldots$$

For the case of first-order syntactic unification, $\emptyset$-unification, considered in the previous subsection, it is proved that if a unification problem is unifiable then there exists a unique m.g.u., but this is different for the case of E-unification in which depending on the theory E, E-unifiability may be undecidable, and if it is decidable then we do not need to have a unique m.g.u., as seen above for the cases of C and A unification. According to theory E, the problems are classified as problems of type unitary, finitary, infinitary, or of type zero. The case C is of type finitary (and not unitary) and the case A is infinitary. For such theories, solutions are built as so called *minimal complete sets of m.g.u.'s*. Theories for which may not exist such sets are called type zero. An example of a type zero theory is the theory with an associative and idempotent function symbol, AI.

Example 2.1.4. *Consider the theory AI, with a function symbol f that is associative and idempotent: $f(x,x) = x$.*

Franz Baader proved that the problem below has not a minimal complete set of m.g.u.'s [13].

$$f(x,f(y,x)) =^{?}_{AI} f(x,f(z,x))$$

Manfred Schmidt-Schauß proved simultaneously the same in [14].

Definition 1 [12] Let E be an equational theory, the E-unification problem has type unitary, finitary, infinitary iff all unifiable problems have a minimal complete set of E-unifiers of respective cardinality at most one, finite, and infinite. If the E-unification problem does not have a minimal complete set of E-unifiers, then it is said to be of type zero.

2.1.3 Term Rewriting System

Definition 2 [Term Rewriting System [15]] A rewrite rule is an identity $l \approx r$ such that l and r are terms containing variables and l is not a variable and $Var(l) \supseteq$

$Var(r)$. It is often written or denoted as $l \to r$. A term rewriting system (TRS) is a set of rewrite rules. Often TRS is written as R

A ***redex*** (reducible expression) is an instance of the left-hand side of a rewrite rule. Contracting the redex means replacing it with the corresponding instance of the right-hand side of the rule, i.e. we can say that a term t is reducible modulo R if and only if there is a rule $l \to r$ in R, a subterm t' at position p of the term t, and a substitution σ such that $\sigma(l) = t'$. The term $t[\sigma(r)]p$ is the result of reducing t by $l \to r$ at position p. The rewrite relation induced by R is written as $\to_R$.

A term is said to be in ***normal form*** with respect to a TRS if and only if no rule can be applied to it.

A TRS is ***terminating*** if and only if there is no infinite reduction chain/sequence (no infinite derivations are possible).

A term rewriting system R is said to be ***confluent*** if and only if the following (diamond) property holds: If t, u, v and w are any terms then

$$\forall t \forall u \forall v[(t \to_R^* u \wedge t \to_R^* v) \Rightarrow \exists w(u \to_R^* w \wedge v \to_R^* w)]$$

R is ***convergent*** if and only if it is both terminating and confluent.

Example 2.1.5. *Consider the term rewriting system given by the four rewrite rules below.*

$$\begin{aligned} &x + x \to 0 \\ &x + 0 \to x \\ &h(0) \to 0 \\ &h(x + y) \to h(x) + h(y) \end{aligned}$$

The term $h(x + 0) + x$ has the redex $x + 0$, then $h(x + 0) + x \to h(x) + x$, which is a normal form. But also $h(x + 0)$ is a redex of the term $h(x + 0) + x$, and thus, $h(x + 0) + x \to (h(x) + h(0)) + x \to (h(x) + 0) + x \to h(x) + x$.

That the rewriting system is terminating can be proved using the lexicographic ordering given by the total number of zeros and additions in arguments of the operator h, and the size of terms. Notice that the first measure decreases after applying the third and fourth rule, and case it does not happens applying the first and the second rule, their application decreases the size of terms. Confluence of the rewriting system results from application of the Knuth-Bendix critical pair theorem (see e.g. [15]).

2.1.4 Asymmetric Unification

When an equational theory is divided into a set of rewrite rules R and a set of equations E, this division is called decomposition.

An asymmetric unification problem for two terms s and t is represented by $s =^?_\downarrow t$.

Definition 3 Given a decomposition (Σ, E, R) of an equational theory, a substitution σ is an asymmetric R, E-unifier of a set Γ of asymmetric equations $\{s_1 =^?_\downarrow t_1, \ldots, s_n =^?_\downarrow t_n\}$ iff for each asymmetric equation denoted as $s_i =^?_\downarrow t_i, \sigma$ is an $(E \cup R)$-unifier of the equation $s_i =^?_\downarrow t_i$, and $\sigma(t_i)$ is in R, E-normal form. [7].

Important examples of asymmetric unification modulo equational theories *ACUN* and *ACUNh* that will be considered in the next chapter.

2.1.5 Disunification

Disunification deals with solving a set of equations and disequations with respect to a given equational theory E.

Definition 4 For an equational theory E, a disunification problem is a set of equations and disequations

$$\Gamma = \{s_1 =^?_E, \ldots, s_n =^?_E t_n\} \cup \{s_{n+1} \neq^?_E t_{n+1}, \ldots, s_{n+m} \neq^?_E t_{n+m}\}$$

A solution to this problem is a substitution σ such that:

$$\sigma(s_i) =_E \sigma(t_i) \qquad (i = 1, \ldots, n)$$

and

$$\sigma(s_{n+j}) \neq_E \sigma(t_{n+j}) \qquad (j = 1, \ldots, m).$$

Observation An equation is an expression of the form $s = t$ where s and t are terms and a disequation is an expression of the form $s \neq t$.

Definition 5 A system is an equation, a disequation or an expression of the form $P_1 \wedge \cdots \wedge P_n$ where $P_1 \ldots P_n$ are systems or an expression $P_1 \vee \cdots \vee P_n$ where $P_1 \ldots P_n$ are systems [16].

Definition 6 An equational problem is an expression of the form

$$\exists w_1, \ldots, w_m \forall y_1, \ldots, y_n : P$$

where P is the system and $w_1, \ldots, w_m, y_1, \ldots y_n$ are distinct variables [16].

Important examples of disunification modulo equational theories *ACUN* and *ACUNh* that will be considered in the next chapter.

2.2 Complexity Theory

Definition 7 [Decision problems] Decision problems are those problems in which the answer to the problem is either yes or no.

Definition 8 [Optimization problems] Many problems are optimization problems, in which each feasible solution has an associated value, and we wish to find a feasible solution with the best value.

In computer science, computational complexity theory is the branch of the theory of computation that studies the resources, or cost, of the computation required to solve a given computational problem. Complexity theory analyzes the difficulty of computational problems in terms of many different computational resources. The complexity class P is the set of decision problems that can be solved by a deterministic machine in polynomial time. This class corresponds to an intuitive idea of the problems which can be effectively solved in the worst cases. The complexity class NP is the set of decision problems that can be solved by a non-deterministic machine in polynomial time. This class contains many problems that people would like to be able to solve effectively. All the problems in this class have the property that their solutions can be checked deterministically in polynomial time.

Definition 9 A problem X is called an NP-hard problem if every problem in NP is polynomially reducible to X [17].

Definition 10 A problem X is called an NP-complete problem if (i) X belongs to NP and (ii) X is NP-hard [17].

For proving that the problem A is NP-hard we will use the technique of reducing a known NP-complete problem B to it. If we can show that any input of the problem B can be reduced polynomially to an input of problem A, such that the corresponding decision questions over these inputs have the same answer (yes or no), then one can conclude that problem A is NP-hard. And if we also can prove that problem A belongs to NP, then A is NP-complete. The following problems are well known to be NP-complete: vertex cover, dominating set, SAT, 3SAT, 3-coloring, and clique. Each of these problems is described in more detail in [17]. Many other problems are NP-complete problems for example hamiltonian cycle, hamiltonian path, independent set, 3-dimentional matching, knapsack, traveling-salesman and many more given in [17].
For instance, to prove the traveling-salesman problem is NP-complete, we first need to show that traveling-salesman problem belongs to NP. Then we need to show any other known NP-complete problem that reduces polynomially to the traveling salesman problem. The hamiltonian cycle problem is reduced to the traveling salesman problem in polynomial time [18].
And some other problems are NP-hard problems because they do not belong to NP-complete problems. For instance, halting problem is NP-hard but not NP-complete [19].

2.3 Automata Theory

Automata theory is very useful in the study of the theory of computation. In particular, **finite automata** (FA) are a useful model for several important applications in hardware and software design. In this work, a variant of this model will be useful for proving the NP-hardness of a class of unification problems.

Definition 11 [Finite automata] A finite automaton is a 5-tuple of the form $(Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set, called the states;
- Σ is a finite set, called the alphabet;
- $\delta : Q \times \Sigma \times Q$ is a relation, called transition function;

- $q_0 \in Q$ is the start state (initial state); and
- $F \subset Q$ is the set of accepting states, also called final states.

An automaton receives an input string from its alphabet Σ, starts from the initial state q_0 and processes symbol by symbol of the input string, from left to right, and moves to a next state according to the transition function. When the transition is indeed a function, it is said that the automaton is *deterministic* (DFA), otherwise, it is said to be *non-deterministic* (NDFA).

The language accepted by an automaton is the set of strings of the alphabet that are processed by it, starting from the initial state and (may) finishing in an accepting state. The problem of deciding whether a string $w \in \Sigma^*$ belongs or not to the language accepted by an automaton is known as the automata language accepting problem.

Definition 12 [Transition Diagram (State Diagram)] A transition diagram for a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is a graph defined as follows:

- For each state in Q there is a node.
- For each state q in Q and each input symbol a in Σ, let $\delta(q, a) = p$. Then the transition diagram has an edge from node q to node p, labeled a. If there are several input symbols that cause transitions from q to p, then the transition diagram can have one edge that is labeled by the list of these symbols.
- There is an arrow to the start state q_0, labeled Start. This arrow does not originate at any node.
- Nodes corresponding to accepting states (those in F) are marked by a double circle. States not in F have a single circle.

Example 2.3.1. *Consider the finite automaton* $A_1 = (Q, \Sigma, \delta, q_1, F)$*, where*

- $Q = \{q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- δ *is given as* $\delta(q_1, 0) = \{q_1\}, \delta(q_1, 1) = \{q_2\}, \delta(q_2, 0) = \{q_3\}, \delta(q_2, 1) = \{q_2\}, \delta(q_3, 0) = \{q_2\}$*, and* $\delta(q_3, 1) = \{q_3, q_2\}$
- q_1 *is the start state, and*
- $F = \{q_2\}$.

A_1 can be described by the transition diagram in Figure 2.1.

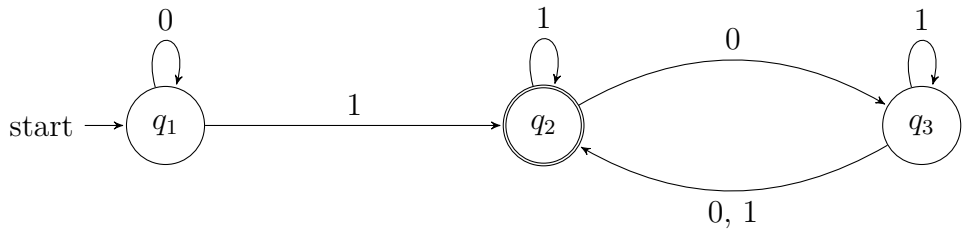


Figure 2.1: The finite automaton A_1

If δ is redefined such that $\delta(q_3, 1) = \{q_2\}$, the automaton is deterministic. The language accepted for this automaton is the language of binary words in $\{0, 1\}$ that have at least one 1, then some occurrence of 01 or finish in 1 or 00.

2.4 Ring Theory

Definition 13 [Ring] A ring is a nonempty set R together with two binary operations, called addition and multiplication, satisfying the following axioms.

- $(R, +)$ is an abelian group. i.e.,
  - $a + b = b + a$ for all $a, b \in R$.
  - $(a + b) + c = a + (b + c)$ for all $a, b, c \in R$.
  - There exists an element $0 \in R$ such that $a + 0 = a$ for all $a \in R$.
  - For every $a \in R$, there exists an element $(-a) \in R$ such that $a + (-a) = 0$.
- $(R, .)$ is monoid. i.e.,
  - $(ab)c = a(bc)$ for all $a, b, c \in R$.
  - There exists an element $1 \in R$ such that $1a = a1 = a$ for all $a \in R$.
- Multiplication is distributive over addition from both sides i.e., from left-side and also from right-side.
  - $a(b + c) = ab + ac$ and $(a + b)c = ac + bc$ for all $a, b, c \in R$.

  A ring R is said to be commutative ring if, in addition,

- $ab = ba$ for all $a, b \in R$.

To specify the ring we sometimes needs to write 0_R and 1_R for the elements 0 and 1.

Definition 14 [Semiring]

A semiring is a nonempty set S together with two binary operations, called addition and multiplication, satisfying the following axioms.

- $(S, +)$ is an abelian monoid. i.e.,
  - $a + b = b + a$ for all $a, b \in S$.
  - $(a + b) + c = a + (b + c)$ for all $a, b, c \in S$.
  - There exists an element $0 \in S$ such that $a + 0 = a$ for all $a \in S$.
- $(S, .)$ is monoid. i.e.,
  - $(ab)c = a(bc)$ for all $a, b, c \in S$.
  - There exists an element $1 \in S$ such that $1a = a1 = a$ for all $a \in S$.
- Multiplication is distributive over addition from both sides i.e., from left-side and also from right-side.
  - $a(b + c) = ab + ac$ and $(a + b)c = ac + bc$ for all $a, b, c \in S$.

A semiring is also called a ring without subtraction.

Definition 15 [Polynomial]

A polynomial is an expression consisting of variables (also called indeterminate) and coefficients, together with the binary operations that consist of addition, subtraction, multiplication, and non-negative integer exponentiation of variables.

Definition 16 [Polynomial ring over $\mathbb{N}$]

A polynomial ring $\mathbb{N}[X]$, in X over $\mathbb{N}$ can be defined as the set of expressions, called polynomials in X of the form

$$P = p_0 + p_1X + p_2X^2 + \ldots + p_{m-1}X^{m-1} + p_mX^m,$$

where, $p_0, p_1, \ldots, p_m$ are the coefficients of P, and are elements of the set $\mathbb{N}$. The symbol X is called an indeterminate or variable.

Chapter 3

Complexities of asymmetric unification and disunification modulo *ACUN* and *ACUNh*

In this chapter it is illustrated how complexities of asymmetric unification and disunification problems vary when equational theories change. This chapter and given examples are taken from [7] making small adjust and providing additional details when necessary.

3.1 Disunification modulo *ACUN* is in P whereas asymmetric unification is NP-hard

3.1.1 Disunification modulo *ACUN* is in P

We have the equational theory *ACUN*, over the signature $\Sigma = \{+, 0\}$ where $+$ is a binary symbol, 0 is a constant, and with the following properties below

$$\begin{aligned} (x + y) + z &= x + (y + z) && (\textbf{A}\text{ssociativity}) \\ x + y &= y + x && (\textbf{C}\text{ommutativity}) \\ x + 0 &= x && (\textbf{U}\text{nity}) \\ x + x &= 0 && (\textbf{N}\text{ilpotence}) \end{aligned}$$

This theory is also called theory of boolean *XOR* operator. Disunification modulo *ACUN* can be solved in polynomial time using Gaussian Elimination over $\mathbb{Z}_2$.

The procedure to solve this problem as given in [7] will be discussed.

Consider there is a set of variables $\{x_1, x_2, ..., x_m\}$, a set of constant symbols $\{c_1, c_2, ..., c_n\}$, and the given equations and disequations to be unified.

The variables and constants will be in the following order. i.e., $x_1 > x_2 > \cdots > x_m > c_1 > c_2 > \cdots > c_n$.

We can have different equations and disequations but we will start by picking an equation with the leading variable x_1 in it, then eliminate the variable x_1 from all other equations and disequations.

Repeat this process by picking the equation with leading variable x_2, then similarly do the same with the equation having leading variable x_3, until we have no variables to eliminate.

The problem has a solution iff

- we have no equations consisting of only constants, i.e., $c_1 + c_2 = c_3$, and
- no disequations of the form $0 \neq 0$ included in the given problem.

By following this process we can solve the disunification problem in polynomial time using Gaussian Elimination over $\mathbb{Z}_2$ [7], as illustrated by Example 3.1.1.

Example 3.1.1 (Taken from [7]). *Given the two equations*

$$x_1 + x_2 + x_3 + c_1 + c_2 =^? 0$$

$$x_1 + x_3 + c_2 + c_3 =^? 0,$$

and a disequation

$$x_2 \neq^? 0.$$

Solution:

By using the Gaussian Elimination method and the theory ACUN we have

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & c_1 + c_2 \\ 1 & 0 & 1 & c_2 + c_3 \\ 0 & 1 & 0 & 0 \end{array}\right]$$

Eliminating x_1 from the second row by adding row 1 and row 2 we get

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & c_1 + c_2 \\ 0 & 1 & 0 & c_1 + c_3 \\ 0 & 1 & 0 & 0 \end{array}\right]$$

Eliminate x_2 from the first row by adding row 2 and row 1 we get

$$\left[\begin{array}{ccc|c} 1 & 0 & 1 & c_2 + c_3 \\ 0 & 1 & 0 & c_1 + c_3 \\ 0 & 1 & 0 & 0 \end{array}\right]$$

Eliminate x_2 from the third row by adding row 3 and row 2 we get

$$\left[\begin{array}{ccc|c} 1 & 0 & 1 & c_2 + c_3 \\ 0 & 1 & 0 & c_1 + c_3 \\ 0 & 0 & 0 & c_1 + c_3 \end{array}\right]$$

Thus we get

$$x_1 + x_3 + c_2 + c_3 =^? 0$$
$$x_2 + c_1 + c_3 =^? 0$$
$$c_1 + c_3 \neq^? 0$$

So, we have

$$x_2 =^? c_1 + c_3$$
$$x_1 + x_3 =^? c_2 + c_3$$

And the solution set is:

$$\sigma = \{x_1 \mapsto c_2, \quad x_2 \mapsto c_1 + c_3, \quad x_3 \mapsto c_3\}$$

3.1.2 Asymmetric unification modulo *ACUN* is NP-hard

Asymmetric unification problem modulo theory of *ACUN* is NP-hard as given in [7].

As in the section of complexity theory, it is shown that if we can reduce any problem from any of the known NP problems and this reduction is in polynomial time then the reduced problem will be NP-hard.

In this case, we have a 3-colorability problem, so here it is shown that how we can build an instance of asymmetric unification problem in polynomial time from the 3-colorability graph problem.

For this, take the graph i.e., $G = (V, E)$, where V is the set of vertices $V = \{v_1, v_2, v_3, \ldots, v_n\}$, E is the set of edges $E = \{e_1, e_2, e_3, \ldots, e_m\}$ and C is the set of colours $C = \{c_1, c_2, c_3\}$ where $n \geq 3$.

The condition for the graph G is 3-colorable is that, if none of the adjacent vertices $\{v_i, v_j\} \in E$ have the same color assigned from the set C then this is called graph G is 3-colorable.

Now, an instance of asymmetric unification can be constructed as follows. We need to have variables for corresponding vertices and edges to build an instance of the asymmetric unification problem. In [7] it is given that for each vertex v_i, we can assign a variable x_i and for each edge e_k, we can assign a variable y_k. Now, we can create an equation by considering edges. For instance, for the edge $e_k = \{v_i, v_j\}$, we assign variable to the edge and corresponding vertices of that edge i.e., $EQ_k = v_i + v_j + e_k =^?_\downarrow x_i + x_j + y_k$. The variable y_k appears just one time in every equation. So, the set of asymmetric unification problem will be

$$S = \{EQ_1, EQ_2, \ldots, EQ_m\}$$

Now we need to assign colors to variables. If G is 3-colorable, then there is a color assignment $\theta : V \to C$ such that $\theta(v_i) \neq \theta(v_j)$ if $e_k = \{v_i, v_j\} \in E$. This can be converted into an asymmetric unifier α for S as follows: $\theta(v_i)$ to $x_i, \theta(v_j)$ to x_j, and the remaining color to y_k. Thus $\alpha(x_i + x_j + y_k) =_\downarrow c_1 + c_2 + c_3$ and therefore α is an asymmetric unifier of S since the term $c_1 + c_2 + c_3$ is irreducible modulo $ACUN$.

Suppose S has an asymmetric unifier β. Note that β cannot map x_i, x_j or y_k to 0, or to a term of the form $u + v$ since $\beta(x_i + x_j + y_k)$ has to be in normal form (or irreducible). Hence for each equation EQ_k, it must be that $\beta(x_i), \beta(x_j), \beta(y_k) \in \{c_1, c_2, c_3\}$ and $\beta(x_i) \neq \beta(x_j), \beta(x_j) \neq \beta(y_k)$ and $\beta(x_i) \neq \beta(y_k)$. Thus β is a 3-coloring of G.

Example 3.1.2. *Taken from [7]. Given the graph $G = (V, E)$, whose set of vertices is $V = \{v_1, v_2, v_3, v_4\}$, and set of edges is $E = \{e_1, e_2, e_3, e_4\}$, where*
$e_1 = \{v_1, v_3\}, e_2 = \{v_1, v_2\}, e_3 = \{v_2, v_3\}, e_4 = \{v_3, v_4\}$ and the colour set is $C = \{c_1, c_2, c_3\}$,
For each vertex v_i we assign a variable x_i and for each edge e_j we assign a variable y_j. Then instance of asymmetric unification will be constructed as

$$EQ_1 = c_1 + c_2 + c_3 =^?_\downarrow x_1 + x_3 + y_1$$

$$EQ_2 = c_1 + c_2 + c_3 =^?_\downarrow x_1 + x_2 + y_2$$

$$EQ_3 = c_1 + c_2 + c_3 =^?_\downarrow x_2 + x_3 + y_3$$

$$EQ_4 = c_1 + c_2 + c_3 =^?_\downarrow x_3 + x_4 + y_4$$

Now suppose the vertices in the graph G are given this color assignment:

$$\theta = \{v_1 \to c_1, v_2 \to c_2, v_3 \to c_3, v_4 \to c_1\}$$

Now create an asymmetric unifier based on this θ by mapping each x_i to $\theta(v_i)$ and for each edge e_j mapping y_j to the remaining color from $\{c_1, c_2, c_3\}$ after both its vertices are assigned. For instance, for $e_1 = \{v_1, v_3\}$ since x_1 is mapped to c_1 and x_3 is mapped to c_3, we have to map y_1 to c_2. We will do this in a similar way for all given edges,
Thus the asymmetric unifier is

$$\sigma = \{x_1 \mapsto c_1, x_2 \mapsto c_3, x_3 \mapsto c_2, y_1 \mapsto c_3, y_2 \mapsto c_2, y_3 \mapsto c_1, y_4 \mapsto c_3\}$$

According to [7] membership in NP is an open problem.

3.2 Ground disunification modulo *ACUN*h is in P whereas ground asymmetric unification is NP-hard

This theory is the same as *ACUN* but including in the signature a unary function symbol h for the homomorphism. We will call it *ACUNh* and it will be specified by the decomposition consisting of the *AC* equations and the set of rewriting rules below.

$$\left\{ \begin{array}{cc} (x+y)+z = x+(y+z) & (\textbf{A}\text{ssociativity}) \\ x+y = y+x & (\textbf{C}\text{ommutativity}) \end{array} \right\}$$

$$\left\{ \begin{array}{cc} x+0 \to x & (\textbf{U}\text{nity}) \\ x+x \to 0 & (\textbf{N}\text{ilpotence}) \\ x+(y+x) \to y & (\text{Auxiliary Nilpotence rule}) \\ h(x+y) \to h(x)+h(y) & (\textbf{H}\text{omomorphism over binary AC operator}) \\ h(0) \to 0 & (\textbf{H}\text{omomorphism over unity}) \end{array} \right\}$$

3.2.1 Ground disunification modulo *ACUN*h is in P

This section is taken from [7]. The ground disunification problem refers to check for ground solutions for a set of disequations and equations. The restriction is that only the set of constants provided in the input are allowed, no new constants can be introduced [20].

It is shown that ground disunifiability modulo *ACUNh* can be solved in polynomial time, by reducing the problem to the system of linear equations. This gives us a general solution to all the variables or unknowns.

Suppose there are m equations in our ground disunifiability problem. And disequations are of the form $z \neq 0$. For example, if we have disequations of the form $e_1 \neq e_2$, we introduce a new variable z and set $z = e_1 + e_2$ and $z \neq 0$.

Let there are n variables or unknowns for which we have to find a solution.

For each constant in our ground disunifiability problem, we follow the approach similar to [21], forming a set of linear equations and solving them to find ground solutions.

We use $h^k x$ to represent the term $h(h(\dots h(x)\dots))$ and

$$H^k[x] = h^{k_1}x + h^{k_2}x + \cdots + h^{k_n}x$$

is a polynomial over $\mathbb{Z}_2[h]$

Example 3.2.1.

$$h(x_1) + x_2 =^? h^2(a) + a$$

$$x_1 + h(x_2) =^? h^2(b) + b$$

$$x_1 + x_2 + z =^? 0 \quad (z \neq 0, \quad z = x_1 + x_2 \quad \Rightarrow x_1 + x_2 + z = 0)$$

For constant a :

$$\begin{bmatrix} h & 1 & 0 \\ 1 & h & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1^a \\ x_2^a \\ z^a \end{bmatrix} = \begin{bmatrix} h^2(a) + a \\ 0 \\ 0 \end{bmatrix}$$

$$\Rightarrow h(x_1^a) + x_2^a + 0 = h^2(a) + a$$

$$\Rightarrow x_1^a + h(x_2^a) + 0 = 0$$

$$\Rightarrow x_1^a + x_2^a + z^a = 0$$

Then from the second equation

$$x_1^a + h(x_2^a) + 0 = 0$$

$$x_1^a + h(x_2^a) = 0$$

we get

$$x_1^a = h(x_2^a)$$

put in the first equation

$$\Rightarrow h(h(x_2^a)) + x_2^a = h^2(a) + a$$

$$\Rightarrow h^2(x_2^a) + x_2^a = h^2(a) + a$$

The term with h^2 *becomes* 0 *because we have a coefficient from the polynomial ring* $\mathbb{Z}_2[h]$. *We get*

$$\Rightarrow x_2^a = a$$

By using the value of x_2^a *we can get* x_1^a

$$\Rightarrow x_1^a = h(a)$$

By using the value of x_1^a *and* x_2^a *we get*

$$z^a = x_1^a + x_2^a$$

$$\Rightarrow z^a = a + h(a)$$

$$\Rightarrow \{x_1^a = h(a), \quad x_2^a = a, \quad z^a = h(a) + a\}$$

For constant b :

$$\begin{bmatrix} h & 1 & 0 \\ 1 & h & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1^b \\ x_2^b \\ z^b \end{bmatrix} \begin{bmatrix} 0 \\ h^2(b) + b \\ 0 \end{bmatrix}$$

$$\Rightarrow h(x_1^b) + x_2^b + 0 = 0$$

$$x_1^b + h(x_2^b) + 0 = h^2(b) + b$$

$$x_1^b + x_2^b + z^b = 0$$

From the first equation we get

$$\Rightarrow x_2^b = h(x_1^b)$$

By using the value of x_2^b *we get*

$$\Rightarrow x_1^b + h(h(x_1^b)) + 0 = h^2(b) + b$$

$$\Rightarrow x_1^b + h^2(x_1^b) = h^2(b) + b$$

The term with h^2 *becomes* 0 *because we have a coefficient from the polynomial ring* $\mathbb{Z}_2[h]$. *We get*

$$\Rightarrow x_1^b = b$$

By using the value of x_1^b *we can get* x_2^b

$$\Rightarrow x_2^b = h(b)$$

By using the value of x_1^b and x_2^b we get

$$z^b = x_1^b + x_2^b$$

$$z^b = b + h(b)$$

$$\Rightarrow \{x_1^b = b, \quad x_2^b = h(b), \quad z^b = h(b) + b\}$$

$$z = z^a + z^b$$

$$z = h(a) + a + h(b) + b \neq 0$$

Thus, the solution set will be

$$\sigma = \{x_1^a \mapsto h(a),\ x_2^a \mapsto a,\ x_1^b \mapsto b,\ x_2^b \mapsto h(b),\ z^a \mapsto h(a) + a,\ z^b \mapsto h(b) + b\}$$

3.2.2 Ground Asymmetric Unification modulo $ACUNh$ is NP-hard

Here, we use the technique in [7] that applies automata theory to prove that asymmetric unification modulo $ACUNh$ is NP-hard. The technique is not the usual one but uses the equivalence relation between decidability of weak second-order theory of one successor (WS1S) and membership in the language accepted by finite automata. The formulas of WS1S are built from atomic formulas of the form $such(x, y)$ for first-order variables x and y, where $x \in X$ for a first-order variable x and a set of variables X that is a monadic second-order variable, using boolean connectives, first- and second-order existential quantification, the latter restricted to finite sets [22]. The key result in this methodology is Büchi's theorem that states that interpretations of formula in WS1S are related to the language of variable assignments that satisfy the formula and are equivalent to the language accepted by a (Büchi) finite automaton. Since the Boolean satisfiability problem reduces to satisfiability in WS1S, by proving the correspondence between solvability of ground asymmetric $ACUNh$ unification and language acceptation in automata.

We will construct the automata for the equation and we will see how this automata problem will be transformed into the ground asymmetric unification problem. If an automaton can be reduced in polynomial time to the ground asymmetric unification problem then we have the desired result.

To construct the automata we have the set of alphabets consist of column vectors of bits with length equal to the number of variables in the problem. Thus, the inputs to the automata are finite sequences of column vectors of bits. First, we see the case for one constant a, then after we will see for the case when we have more than one constant.

The examples illustrate how automata are constructed for different problems. To avoid the mess up of the diagrams, a so-called *dead state* has been included only for the automaton in the first example. When the transition is undefined, a transition to the dead state is included. So in all automata, all missing transitions lead to the dead state by default.

Example 3.2.2 (Taken from [7])**.** *Consider the problem given by equation* $P =^{?} Q + R$.

Solution: *Now we will construct the automata for the given problem. For the problem* $P =^{?} Q + R$ *the alphabets of the automaton, consists of 3-bit vectors for variables* P, Q, *and* R, *respectively. The ordering of variables is* $\begin{pmatrix} P \\ Q \\ R \end{pmatrix}$. *The corresponding alphabet symbol* $\begin{pmatrix} P \\ Q \\ R \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$ *means* P *has a value* 1, *when either* Q *or* R *have value* 1. *For example if* $R = 0, Q = 1$, *then* $P = 1$

Whenever input symbols of the alphabet that are solutions of the problem are processed, the transitions maintain the control in the start state of the automaton that is also the final state. Other symbols move the control to the dead state.

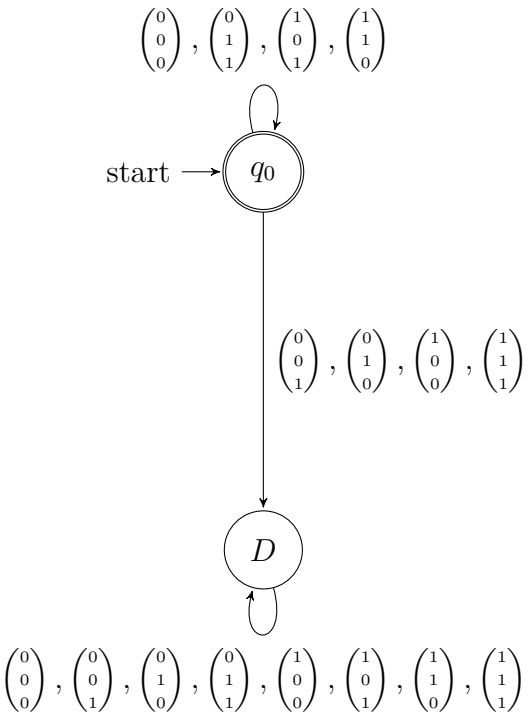


Figure 3.1: Automaton for the problem $P =^{?} Q + R$

Hence, only strings with all their alphabet symbols in

$$\left\{\begin{pmatrix}0\\0\\0\end{pmatrix},\begin{pmatrix}0\\1\\1\end{pmatrix},\begin{pmatrix}1\\0\\1\end{pmatrix},\begin{pmatrix}1\\1\\0\end{pmatrix}\right\}$$

are accepted by this automaton, while all other strings with at least one input symbol in the set

$$\left\{\begin{pmatrix}0\\0\\1\end{pmatrix},\begin{pmatrix}0\\1\\0\end{pmatrix},\begin{pmatrix}1\\0\\0\end{pmatrix},\begin{pmatrix}1\\1\\1\end{pmatrix}\right\}$$

go to the dead state D as they violate the XOR property.

Now we pick strings from the set of accepted strings by this automaton and get the solution for the input problem $P =^{?} Q + R$. *For instance, we pick*

$$\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}1\\1\\0\end{pmatrix} \text{ and } \begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}1\\1\\0\end{pmatrix}\begin{pmatrix}0\\1\\1\end{pmatrix}$$

These accepted strings correspond respectively to the solutions

$$P = a + h(a), Q = h(a) \text{ and } R = a, \text{ and}$$

$$P = a + h(a), Q = h(a) + h^2(a) \text{ and } R = a + h^2(a).$$

Finally, notice that if we pick any string from the set of strings that are not being accepted by this automaton then we will not get the solution. For instance, we pick

$$\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}1\\1\\1\end{pmatrix} \text{ and } \begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}1\\0\\0\end{pmatrix}\begin{pmatrix}0\\1\\1\end{pmatrix}$$

that correspond to the following instances of P, Q *and* R *that are not solutions of the input problem* $P =^{?} Q + R$*:*

$$P = a + h(a), Q = h(a) \text{ and } R = a + h(a), \text{ and}$$

$$P = a + h(a), Q = h^2(a) \text{ and } R = a + h^2(a).$$

Example 3.2.3 (Taken from [7])**.** *Consider the asymmetric equation* $P =^{?}_{\downarrow} Q + R$.

Solution:

To preserve asymmetry on the right-hand side of this equation, $Q + R$ should be irreducible. If either Q or R is empty, or if they have any term in common, then a reduction will occur and the input should be rejected. The dead state is omitted in this automaton.

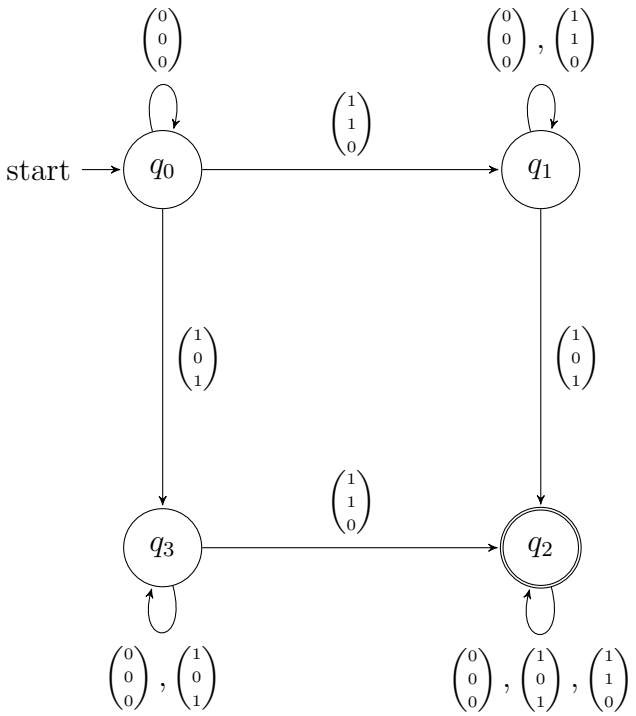


Figure 3.2: Automaton for the problem $P =_{\downarrow} Q + R$

For example, if $Q = h(a)$ and $R = h(a) + a$, there is a reduction, whereas if $R = h(a)$ and $Q = a$, irreducibility is preserved, since there is no common term and neither one is empty. Since neither Q nor R can be empty, any accepted string should have one occurrence of $\begin{pmatrix}1\\0\\1\end{pmatrix}$ *and one occurrence of* $\begin{pmatrix}1\\1\\0\end{pmatrix}$.

Note that the string

$$\begin{pmatrix}0\\0\\0\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}1\\1\\0\end{pmatrix}$$

is accepted by the automaton. This string corresponds to the solution

$$P = h(a) + h^2(a), Q = h^2(a) \text{ and } R = h(a),$$

Finally, notice that the string

$$\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}1\\1\\0\end{pmatrix}\begin{pmatrix}0\\1\\1\end{pmatrix}$$

corresponds to the following instance of P, Q and R that is not a solution of the input problem $P =_{\downarrow} Q + R$, because it cause a reduction.

$$P = a + h(a), Q = h(a) + h^2(a) \text{ and } R = a + h^2(a).$$

For a given problem, an automaton is built for each formula, and the solution consists of the intersection of sets of sequences that are accepted by each automaton. If the intersection is not empty, then we have a solution or an asymmetric unifier for the input set of formulas.

Example 3.2.4 (Taken from [7])**.** *Consider the problem given by equation $X =^? h(Y)$.*

Solution:

Now we will construct the automata for the given problem. For the problem $X =^? h(Y)$ the alphabets of the automaton consist of 2-bit vectors for variables X and Y respectively. The ordering of variables for this automata will be $\begin{pmatrix}Y\\X\end{pmatrix}$. Here, q_0 is the only accepting state and h acts like the successor functions. Here transition occurs with the strings $\binom{1}{0}, \binom{0}{1}$. If $Y = 1$ in current state then $X = 1$ in the next state.

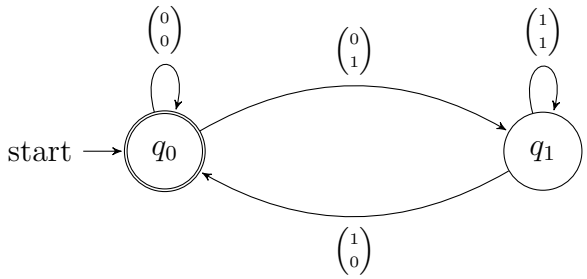


Figure 3.3: Automaton for the problem $X = h(Y)$

Notice that the string

$$\begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}$$

is accepted. This corresponds to the solution

$$X = h^2(a) + h^3(a), Y = h(a) + h^2(a)$$

While the string below is rejected.

$$\binom{0}{1}\binom{0}{0}\binom{1}{0}$$

The last string corresponds to the instantiation

$$X = h^2(a), Y = a$$

Example 3.2.5 (Taken from [7])**.** *Consider the asymmetric equation* $X =^?_{\downarrow} h(Y)$.

Solution:

In this equation, $h(Y)$ *must not be reducible to preserve the asymmetry. So,* Y *cannot be either* 0 *or of the form* $u + v$ *to avoid the reduction. The automaton is similar to the previous one, but the start state may not be the initial state in order to avoid acceptation of* $X = 0, Y = 0$*, and only a value is possible for avoiding instantiation of* Y *with an addition. See below.*

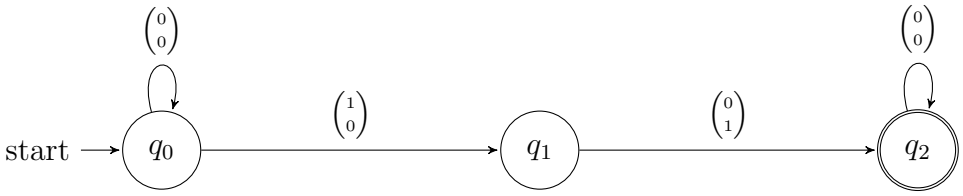


Figure 3.4: Automaton for the problem $X =_{\downarrow} h(Y)$

Equations with more than one constant:

What happens when we have more than one constant? We will follow the approach given in [7]. Suppose there are k constants, say $c_1, \ldots, c_k$. We express each variable X in terms of the constants as follows:

$$X = X^{c_1} + \cdots + X^{c_k}$$

Thus if $X = h(c_1) + c_2 + h^2(c_3)$, then $X^{c_1} = h(c_1), X^{c_2} = c_2$, and $X^{c_3} = h^2(c_3)$. If the variables are $X_1, \ldots, X_m$, then we set

$$X_1 = X_1^{c_1} + \cdots + X_1^{c_k}$$

$$X_2 = X_2^{c_1} + \cdots + X_2^{c_k}$$

$$\vdots$$

$$X_m = X_m^{c_1} + \cdots + X_m^{c_k}$$

For example, if Y and Z are variables and a, b,and c are constants, then we can write $Y = Y^a + Y^b + Y^c$ and $Z = Z^a + Z^b + Z^c$. The examples given below show the equations having more than one constant.

Example 3.2.6. *For the equation $X =_\downarrow h(Y)$ with two constants, a and b we have equations $X^a =_\downarrow h(Y^a)$, and $X^b =_\downarrow h(Y^b)$. However, since the equation is asymmetric, Y must not be equal to 0, and Y has to be a single term of the form $h^i(d)$ where d is either a or b and $i \geq 0$. All components in an equation except any one component have to be 0. Remember that the ordering of variables for this automata is $\begin{pmatrix} Y \\ X \end{pmatrix}$.*

$$X = X^a + X^b$$

and

$$Y = Y^a + Y^b$$

***For constant** a**:** the string $\begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ corresponds to the solution*

$$X = 0 + h(a), \text{ that is } X^a = h(a)$$
$$Y = a + 0, \text{ that is } Y^a = a$$

***For constant b:** the string $\begin{pmatrix} 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ corresponds to the solution*

$$X = 0 + 0 + h^2(b), \text{ that is } X^b = h^2(b)$$
$$Y = 0 + h(b) + 0, \text{ that is } Y^b = h(b)$$

Notice, that in this case ($X =_\downarrow h(Y)$) only one of the solutions may be non-zero in order to be able to build an asymmetric solution. That is, $X = X^a + X^b = h(a) + h^2(b), Y = Y^a + Y^b = a + h(b)$ is not a solution since the term $h(a + h(b))$ is reducible.

The automaton for this problem with two constants a and b consists of a combination of two copies of the automaton in Figure 3.4, see Figure 3.5, where the alphabet was changed to 4-bit vectors, being the first two components associated with non-zero solutions for a and the last two bits associated with non-zero solutions for b.

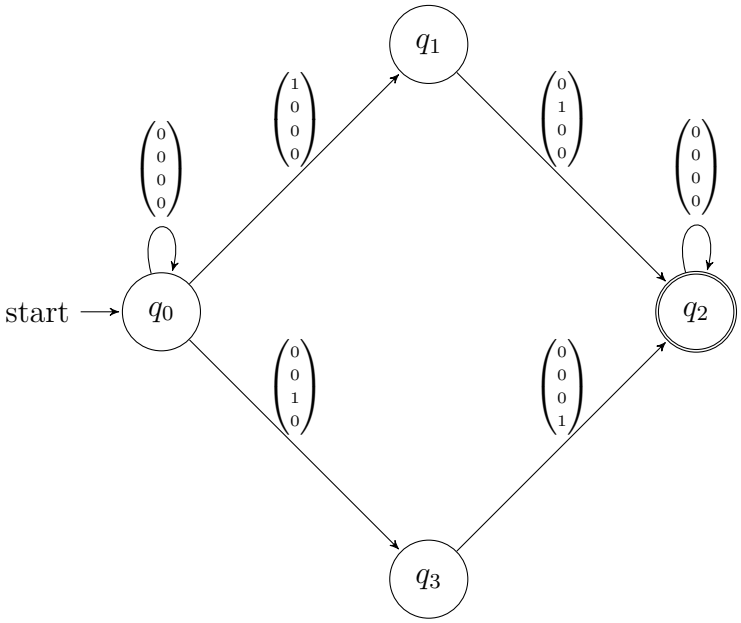


Figure 3.5: Automaton for the problem $X =_{\downarrow} h(Y)$ with two constants a and b

Example 3.2.7. *Now consider the problem below, For constants, a, b and c.*

$$X =_{\downarrow} W + Z$$

The decomposition of the problem is given by the equations below.

$$\begin{aligned} X &= X^a + X^b + X^c \\ W &= W^a + W^b + W^c \\ Z &= Z^a + Z^b + Z^c \end{aligned}$$

And the corresponding automaton is built combining the automaton built for only one constant in Example 3.2.3.

***For constant** a**:** an automaton as the one of Example 3.2.3 is built providing solutions such as:*

$$\{X^a = a + h(a), W^a = h(a), Z^a = a\}$$

***For constant** b**:** also through an automaton as the one of Example 3.2.3, solutions as those below are found.*

$$\{X^b = b + h(b) + h^2(b), W^b = h(b) + h^2(b), Z^b = b\}$$

***For constant** c**:** in the same manner, solutions as the one below can be obtained.*

$$\{X^c = c + h(c), W^c = 0, Z^c = c + h(c)\}$$

Combining such solutions one obtains solutions for the given problem as below.

$$\underbrace{X^a + X^b + X^c}_{X} \;=_{\downarrow}\; \underbrace{W^a + W^b + W^c}_{W} \;+\; \underbrace{Z^a + Z^b + Z^c}_{Z}$$

That is,

$$(a + h(a)) + (b + h(b) + h^2(b)) + (c + h(c)) =_{\downarrow} (h(a) + h(b) + h^2(b)) + (a + b + c + h(c))$$

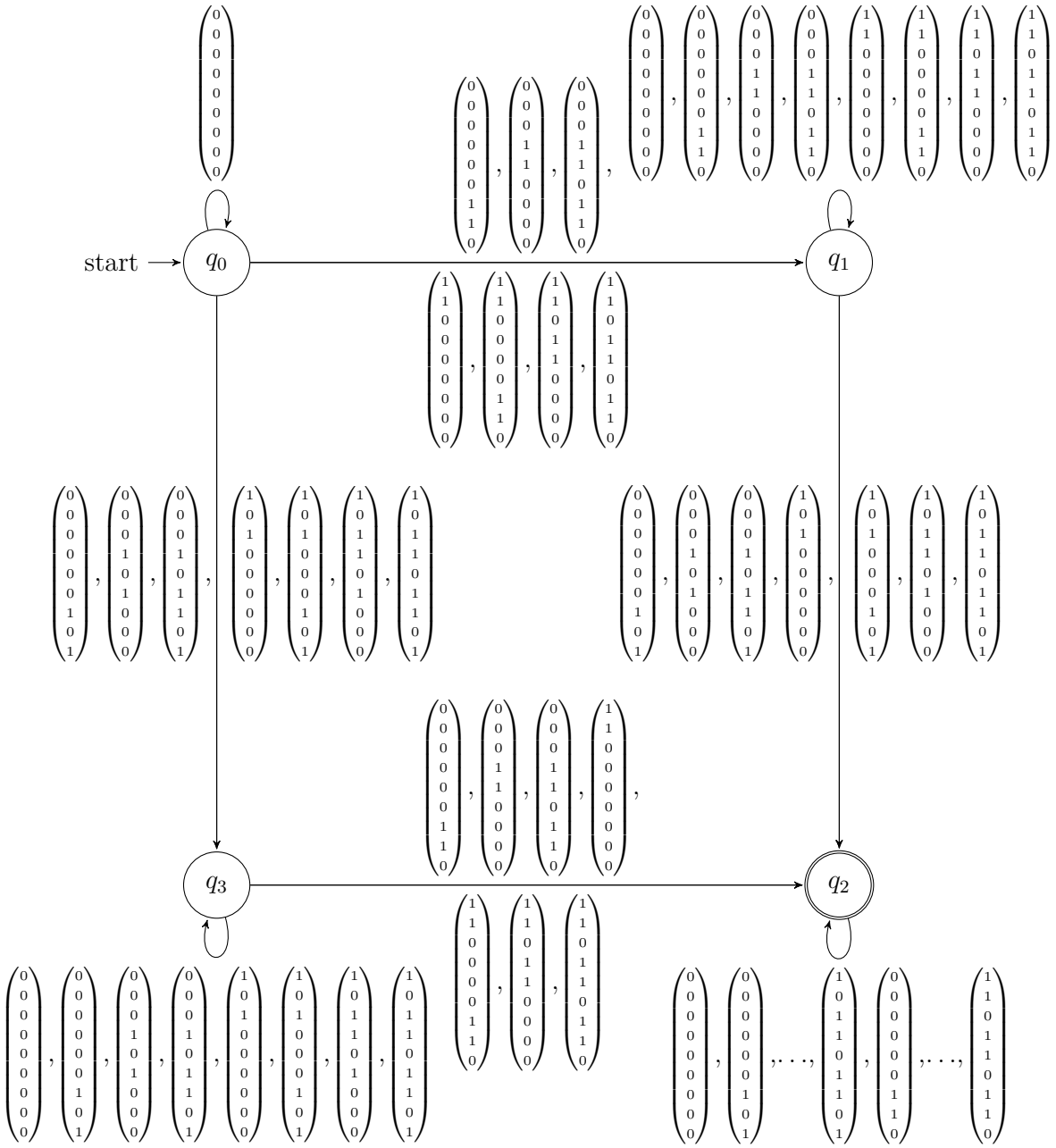


Figure 3.6: Automaton for the problem $P =_{\downarrow} Q + R$

All of the above given examples help to understand the reduction from the automata problem to the ground asymmetric unification problem in polynomial time. So, this is the NP-hard problem by [7].

Chapter 4

Asymmetric Unification modulo *XOR* (*ACUN*) with Uninterpreted Function Symbols

This chapter studies an algorithm that is taken from the Ph.D. dissertation of Zhiqiang Lui [1] and from its related paper [23]. They introduced an algorithm that is based on inference rules to find asymmetric unifiers for the *XOR* theory. The given examples illustrate the application of the inference rules.

4.1 *XOR* theory

The equational theory *XOR* that has the equational properties of *ACUN* over a signature Σ with a binary function symbol $\oplus$ and constant unity 0:

$$\begin{aligned} x \oplus (y \oplus z) &= (x \oplus y) \oplus z && \text{(Associativity)} \\ x \oplus y &= y \oplus x && \text{(Commutativity)} \\ x \oplus 0 &= x && \text{(Unity)} \\ x \oplus x &= 0 && \text{(Nilpotence)} \end{aligned}$$

The theory *XOR* is decomposed into the union of the *AC* equational theory denoted as E, and the convergent rewriting system R for the theory *UN*, given below.

$$E = \begin{cases} x \oplus (y \oplus z) = (x \oplus y) \oplus z & \text{(Associativity)} \\ x \oplus y = y \oplus x & \text{(Commutativity)} \end{cases}$$

$$R = \begin{cases} x \oplus 0 \to x & \text{(Unity)} \\ x \oplus x \to 0 & \text{(Nilpotence)} \end{cases}$$

The algorithm starts with the computation of a standard *XOR*-unifier of the input unification problem, and then search for corresponding asymmetric unifiers. From a standard *XOR*-unifier, the inference rules will be used to search for equivalent asymmetric unifiers.

Given a decomposition (Σ, E, R), and an asymmetric unification problem

$$\Gamma = \{t_1 =^?_\downarrow t'_1, \ldots, t_n =^?_\downarrow t'_n\}$$

then we will follow these steps given in [23]:

1. Find the complete finite set S of standard unifiers of the given problem by using any unification algorithm. If the set of standard unifiers is empty, then there are no asymmetric unifiers.

2. For each unifier σ from the set S, check whether every $\sigma(t'_i)$ is in R, E-normal form. If yes, then all unifiers in S will be asymmetric unifiers also.

3. For a unifier σ such that some $\sigma(t'_i)$ is not in R, E-normal form, then try to find an equivalent asymmetric unifier.

4. If step 2 and step 3 did not give any asymmetric unifier, this implies that σ or its equivalents cannot be the asymmetric unifiers. We can find the instances of the given unifier by instantiating variables in it, which can be an asymmetric unifier. So, the complete set of instances of a given unifier σ can be generated by instantiating suitable variables. For each instance of σ, the above steps are repeated.

4.2 Notations

- The variables which are in the unification problem Γ are called **original variables**.
- The variables which are not in the unification problem Γ but in the range of σ are called **support variables**.
- A term S is called a **sum term** if its normal form has the form $s_1 \oplus s_2 \oplus \cdots \oplus s_n, n > 1$
- A term s is called a **simple term** if it does not have $\oplus$ as its outermost symbol.
- A variable x is said to be in **conflict** with a simple term s if both x and s appear in some t'_i in Γ.

- A substitution δ **satisfies** Υ iff δ satisfies every constraint pair in Υ i.e. for a pair $(v, s) \in \Upsilon$, δ satisfies (v, s) iff $\delta(v) \oplus \delta(s)$ is irreducible with respect to the theory taken.
- A substitution δ **violates** Υ if it does not satisfy Υ.
- A substitution δ **satisfies** Δ iff δ satisfies every disequation in Δ, otherwise, we say δ violates Δ.

4.3 Inference System

The algorithm presented by Zhiqiang Lui in [1] is a collection of inference rules on a triple of sets:

$$\frac{\sigma \parallel \Upsilon \parallel \Delta}{\sigma' \parallel \Upsilon' \parallel \Delta'}$$

- where σ (sigma) is a standard XOR unifier of Γ,
- Υ (Upsilon) is called a constraint set, which is a set of pairs of the form (v, s), where v is a variable and s is a simple term or a variable. The pair (v, s) tells that v is in conflict with s.
- and Δ (Delta) is the set of disequations having members of the form $s \oplus t \neq^? 0$, where s and t with the same uninterpreted function symbol.

4.3.1 Inference Rules

The set of inference rules transforms members of a complete set of standard *XOR* unifiers into a complete set of asymmetric *XOR* unifiers. Before discussing rules here is a sketch of the whole procedure that will help to understand the application of rules.

Sketch

All these inference rules can be divided into three main groups ***Splitting, Branching*** and ***Instantiation***. Then again we can divide these rules further into two groups: In the first group, we have the ***Splitting*** and ***Branching*** rules, and in the second group we have ***Instantiation*** rules. The application of the rules to any given problem will be according to the grouping. We need to follow the sequence given below.

The ***Splitting*** rule is applied first to move all of the original variables out from the range of an XOR unifier.

Then, all ***Branching*** rules will be applied to eliminate conflicts that occur after the application of the splitting rule. These conflicts can occur between original variables with other support variables or other non-variable subterms.

The the ***Non-Variable Branching*** rule eliminates a conflict between the original variable x and a non-variable simple term s or a. This rule will be applied first until there is no application of this rule remains possible.

Then the ***Variable Branching*** rule and the ***Useless Branching*** rule will be applied to remove conflict between two original variables i.e. two original variables sharing the common part in their substitution.

At last the ***Instantiation*** rules will be applied to generate instances of equivalent *XOR* unifiers. The ***Decomposition Instantiation*** rule generates instances of an *XOR* unifier, whereas the ***Elimination Instantiation*** rule generates instances by mapping support variables to 0. If the result is not an asymmetric *XOR* unifier, then the algorithm will repeat this complete process again.

Now we will discuss the inference rules introduced by Zhiqiang Lui one by one with examples.

Splitting Rule (S)

This rule introduced in [23] is used to transform all the top variables in the given unifier σ into the support variables.

Splitting Rule (S)

$$\boxed{\frac{[x \mapsto y \oplus S \oplus T] \cup \sigma \parallel \Upsilon \parallel \Delta}{([x \mapsto y \oplus S \oplus T] \cup \sigma) \circ \theta \parallel \Upsilon\theta \parallel \Delta\theta}}$$

where $\theta = \{y \mapsto v \oplus S\}$ and v is a fresh support variable.

- $x, y \in Vars(\Gamma)$
- $y \notin Vars(S)$

After Splitting there will be no original variables in the range of σ, all the top variables which appear in the range of σ will be a support variables.

Example 4.3.1 (Taken from [1])**.** *Given an asymmetric unification problem.*

$$\Gamma = \{x \oplus z =^?_\downarrow y \oplus a, \quad x \oplus a =^?_\downarrow x \oplus a, \quad x \oplus z =^?_\downarrow x \oplus z\}$$

Solution:

$$\sigma = [x \mapsto y \oplus z \oplus a] \overset{S}{\Rightarrow}$$

$$\sigma_1 = [x \mapsto v_1 \oplus a, \quad y \mapsto v_1 \oplus z] \overset{S}{\Rightarrow}$$

$$\sigma_2 = [x \mapsto v_1 \oplus a, \quad y \mapsto v_1 \oplus v_2, \quad z \mapsto v_2]$$

After applying Splitting, there will be no original variables in the range of σ.

Branching Rules

There are three rules in this part. The main idea as introduced in [23] is to try to transform a unifier into an equivalent unifier without conflicts.

Non-Variable Branching (NVB)

The rule ***Non-Variable Branching*** introduced in [23] will be applied when we have a conflict between some original variable x and some non-variable simple term s by dividing the triple into two triples: the substitutions in the first triple assume some variable v can cancel s and the substitutions in the second triple assume v cannot cancel s.

Non-Variable Branching (NVB)

$$\frac{\sigma \parallel \Upsilon \parallel \Delta}{\sigma \circ \theta \parallel (\Upsilon[v'/v] \cup (v', s))\theta \parallel \Delta\theta \quad \bigvee \quad \sigma \parallel \Upsilon\{(v, s)\} \parallel \Delta\theta}$$

where there exist an assignment $[x \mapsto v \oplus s \oplus S] \in \sigma$ and

$$\theta = [v \mapsto v' \oplus s]$$

with v' being a fresh support variable, under the conditions that x has a conflict at a simple non-variable terms s in Γ where $(i) \quad v \notin Vars(s)$ and $(ii) \quad (v, s) \notin \Upsilon$.
Above, $\Upsilon[v'/v]$ means, replace all occurrences of the variable v in the first component of every pair in Υ by the variable v'. The first branch is used when the conflict between x and s is successfully resolved using v by introducing a new support variable v'; the second branch is used when that is not possible, thus leading to an additional constraint (v, s) implying that v and s are in conflict.

Example 4.3.2. *Continuing Example 4.3.1.*

$$\Gamma = \{x \oplus z =^?_\downarrow y \oplus a, \quad x \oplus a =^?_\downarrow x \oplus a, \quad x \oplus z =^?_\downarrow x \oplus z\}$$

$$\sigma_2 = [x \to v_1 \oplus a, \quad y \to v_1 \oplus v_2, \quad z \to v_2]$$

Solution:

Now, we can apply Non-Variable Branching to σ_2 to solve the case that x has a conflict at a in Γ and get:

Branch 1:

$$\sigma_2 = [x \to v_1 \oplus a, \quad y \to v_1 \oplus v_2, \quad z \to v_2] \overset{NVB}{\Rightarrow}$$

$$\sigma_{2.1} = [x \to v_3, \quad y \to v_3 \oplus a \oplus v_2, \quad z \to v_2]$$

$$\Upsilon_1 = \{(v_3, a)\}$$

where $\theta = [v_1 \to v_3 \oplus a]$ and

Branch 2:

$$\sigma_{2.2} = [x \to v_1 \oplus a, \quad y \to v_1 \oplus v_2, \quad z \to v_2]$$

$$\Upsilon_2 = \{(v_1, a)\}$$

In Branch 1, y has a conflict at a and $(v_3, a) \in \Upsilon_1$, so apply Non-Variable Branching, Let $v_2 \to v_4 \oplus a$ and get:

Branch 1.1

$$\sigma_{2.1} = [x \to v_3, \quad y \to v_3 \oplus a \oplus v_2, \quad z \to v_2] \overset{NVB}{\Rightarrow}$$

$$\sigma_{2.1.1} = [x \to v_3, \quad y \to v_3 \oplus v_4, \quad z \to v_4 \oplus a]$$

$$\Upsilon_{1.1} = \{(v_3, a), (v_4, a)\}$$

where $\theta = [v_2 \to v_4 \oplus a]$ and

Branch 1.2

$$\sigma_{2.1.2} = [x \to v_3, \quad y \to v_3 \oplus a \oplus v_2, \quad z \to v_2]$$

$$\Upsilon_{1.2} = \{(v_3, a), (v_2, a)\}$$

We can see $\sigma_{2.1.1}$ is an asymmetric unifier of Γ.

Here $\sigma_{2.1.1}$ is equivalent to σ. Non-Variable Branching is not applied to Branch 2 and 1.2 because the conditions for applying Non-Variable Branching are not satisfied.

Variable Branching (VB)

The rule ***Variable Branching*** introduced in [23] will be applied when there is a conflict between some original variable x and some support variable v. But it will not solve the conflict directly. It will prepare the problem for the instantiation part.

Variable Branching (VB)

$$\frac{\sigma \parallel \Upsilon \parallel \Delta}{\sigma \circ \theta \parallel \Upsilon'\theta \parallel \Delta\theta \quad \bigvee \quad \sigma \parallel \Upsilon \cup \{(v_1, v_2)\} \parallel \Delta}$$

where

$$\theta = [v_1 \to v_{12} \oplus v_1', v_2 \to v_{12} \oplus v_2']$$

There exist an assignment $[x \to v_1 \oplus v_2 \oplus S, y \to v_1 \oplus S']$ in σ

$$\Upsilon' = \Upsilon[v_{12}/(v_1, v_2)] \cup \Upsilon[v_1'/v_1] \cup \Upsilon[v_2'/v_2] \cup$$

$$\{(v_{12}, v_1'), (v_{12}, v_2'), (v_1', v_2'), (v_1', v_{12}), (v_2', v_{12}), (v_2', v_1')\}$$

Here v_{12}, v_1' and v_2' are fresh support variables.

This rule is applied only if (i) x and y have a conflict in Γ and (ii) $(v_1, v_2) \notin \Upsilon$

The first branch is the case when v_1 and v_2 have a common part, whereas the second branch is the case when v_1 and v_2 have nothing in common.

- Here in the first branch, we assume $v_1 \to v \oplus v_1'$ and $v_2 \to v \oplus v_2'$, which means in each instance δ of $\sigma\theta$, $v_1\delta$ and $v_2\delta$ should have the common part $v\delta$ at the same time $v_1'\delta$ and $v_2'\delta$ should not have any common part.
- In the second branch, we assume for any instance δ of σ, $v_1\delta$ and $v_2\delta$ have no common parts, so (v_1, v_2) and (v_2, v_1) are also added into Υ.
- Here we add constraints in both directions ((v_1, v_2) and (v_2, v_1)). It is equivalent to only give the constrain in one direction.

Useless Branching (UB)

The next rule is called ***Useless Branching*** also called ***Auxiliary Branching***. This rule introduced in [23] is applied when an original variable is in conflict with another original variable in Γ and their substitutions in an XOR unifier share a common part.

Useless Branching (UB)

$$\frac{\sigma \parallel \Upsilon \parallel \Delta}{\sigma \circ \theta \parallel (\Upsilon[v'/v] \cup (v', s))\theta \parallel \Delta\theta \quad \bigvee \quad \sigma \parallel \Upsilon \cup \{(v, s)\} \parallel \Delta}$$

where

$$\theta = \{v \to v' \oplus s\}$$

with v' being a fresh support variable, and there exist two assignments $[x \mapsto v \oplus s \oplus S, y \mapsto v \oplus S']$ in σ. This rule is applied only if (i) x, y are in conflict in Γ, (ii) s is a simple non-variable term and $v \notin Vars(s)$ and (iii) $(v, s) \notin \Upsilon$.

The additional simple non-variable term s in the substitution for x in an XOR unifier is used to possibly eliminate the conflict with a new variable v', which stands for the

common shared part of x and y. This rule does not solve the conflict directly, it is preparing for the instantiation part like variable branching rule.

Example 4.3.3 (Taken from [1])**.**

$$\Gamma = \{x \oplus y \oplus z =^{?}_{\downarrow} a, \quad x \oplus y =^{?}_{\downarrow} x \oplus y, \quad z \oplus a =^{?}_{\downarrow} z \oplus a\}$$

$$\sigma = [x \to y \oplus z \oplus a]$$

Solution:

$$\sigma = [x \to y \oplus z \oplus a] \overset{S}{\Rightarrow}$$

$$\sigma_1 = [x \to v_1 \oplus a, \quad y \to v_1 \oplus z] \overset{S}{\Rightarrow}$$

$$\sigma_2 = [x \to v_1 \oplus a, \quad y \to v_1 \oplus v_2, z \to v_2]$$

After applying Variable Branching for the case that y has a conflict at v_1, by letting $v_1 \to v_1' \oplus v$ and $v_2 \to v_2' \oplus v$, we get

$$\sigma_2 = [x \to v_1 \oplus a, \quad y \to v_1 \oplus v_2, z \to v_2] \overset{VB}{\Rightarrow}$$

Branch 1:

$$\sigma_{2.1} = [x \to v_1' \oplus v \oplus a, \quad y \to v_1' \oplus v_2', \quad z \to v_2' \oplus v]$$

$$\Upsilon_1 = \{(v_1', v), (v_1', v_2'), (v, v_1'), (v, v_2'), (v_2', v), (v_2', v_1')\}$$

Branch 2:

$$\sigma_{2.2} = [x \to v_1 \oplus a, \quad y \to v_1 \oplus v_2, \quad z \to v_2]$$

$$\Upsilon_2 = \{(v_2, v_1), (v_1, v_2)\}$$

We can apply Useless Branching to Branch 1 because x has a conflict at v_1' and $(v_1', a) \notin \Upsilon$, after applying Useless Branching by letting $v_1' \to v_1'' \oplus a$ we get:

$$\sigma_{2.1} = [x \to v_1' \oplus v \oplus a, \quad y \to v_1' \oplus v_2', \quad z \to v_2' \oplus v] \overset{UB}{\Rightarrow}$$

Branch 1.1:

$$\sigma_{2.1.1} = [x \to v_1'' \oplus v, \quad y \to v_1'' \oplus a \oplus v_2', \quad z \to v_2' \oplus v]$$

$$\Upsilon_{1.2} = \{(v_1'', v), (v_1'', v_2'), (v, v_1''), (v, a), (v, v_2'), (v_2', v), (v_2', v_2''), (v_2', a), (v_1'', a)\}$$

Branch 1.2:

$$\sigma_{2.1.2} = [x \to v_1' \oplus v \oplus a, \quad y \to v_1' \oplus v_2', \quad z \to v_2' \oplus v]$$

$$\Upsilon_{1.2} = \{(v_1', v), (v_1', v_2' (v, v_1'), (v, v_2'), (v_2', v), (v_2', v_1'), (v_1', a)\}$$

We can not apply any Branching rules to Branch 1.1 and 1.2.

We look at Branch 2. We still can apply Useless Branching to Branch 2 since $(v_1, a) \notin \Upsilon_2$. *Let* $v_1 \to v_1''' \oplus a$ *and get:*

$$\sigma_{2.2} = [x \to v_1 \oplus a, \quad y \to v_1 \oplus v_2, \quad z \to v_2] \overset{UB}{\Rightarrow}$$

Branch 2.1:

$$\sigma_{2.2.1} = [x \to v_1''', \quad y \to v_1''' \oplus a \oplus v_2, \quad z \to v_2]$$

$$\Upsilon_{2.1} = \{(v_2, v_1'''), (v_2, a), (v_1''', v_2), (v_1''', a)\}$$

Branch 2.2:

$$\sigma_{2.2.2} = [x \to v_1 \oplus a, \quad y \to v_1 \oplus v_2, \quad z \to v_2]$$

$$\Upsilon_{2.2} = \{(v_2, v_1), (v_1, v_2), (v_1, a)\}$$

For Branch 1.1 , 1.2 , 2.1 , and 2.2 we have no rules in the Branching part applicable.

Instantiation Rules

The instantiation rules introduced in [23] are used for solving the conflicts by instantiating some support variables based on the equations $x + x \to 0$ and $x + 0 \to x$.

Once we get an instance, the result may violate Υ or Δ. If this is true, we will throw out this instantiation branch.

Decomposition Instantiation (DI)

The ***Decomposition Instantiation*** rule introduced in [23] is used to solve the case when there is a conflict between some original variable x and a simple non-variable term t.

Decomposition Instantiation (DI)

$$\frac{\sigma \parallel \Upsilon \parallel \Delta}{\sigma \circ \theta_1 \parallel \Upsilon\theta_1 \parallel \Delta\theta_1 \quad \bigvee \cdots \bigvee \quad \parallel \sigma \circ \theta_n \parallel \Upsilon\theta_n \parallel \Delta\theta_n \quad \bigvee \quad \sigma \parallel \Upsilon \parallel \Delta''}$$

where there exists an assignment $[x \mapsto s \oplus t \oplus S]$ in σ, x has a conflict with a simple non-variable subterm s in Γ and s and t have the same topmost uninterpreted symbol $\{\theta_1, \ldots, \theta_n\}$ is a complete set of XOR unifiers of $s =^? t$ and $\Delta'' = \Delta \cup \{s \oplus t \neq^? 0\}$.

Example 4.3.4 (Taken from [1])**.**

$$\Gamma = \{f(a) \oplus f(b) =^?_{\downarrow} x \oplus f(y)\}$$

$$\sigma = [x \to f(y) \oplus f(a) \oplus f(b)]$$

Solution:

Since x has a conflict at $f(y)$, we apply Decomposition Instantiation by unifying $f(a)$ and $f(y)$. Then get:

$$\sigma = [x \to f(y) \oplus f(a) \oplus f(b)] \overset{DI}{\Rightarrow}$$

Branch 1:

$$\sigma_1 = [x \to f(b), \quad y \to a]$$

$$\Delta_1 = \phi$$

Branch 2:

$$\sigma_2 = [x \to f(y) \oplus f(a) \oplus f(b)]$$

$$\Delta_2 = \{f(y) \oplus f(a) \neq^? 0\}$$

σ_1 is an asymmetric unifier of Γ. We keep it and look at Branch 2.

In Branch 2, we still can apply Decomposition Instantiation since x has a conflict at $f(y)$, and $f(b) \oplus f(y) \neq^? 0 \notin \Delta$. We unify $f(y)$ and $f(b)$ and get:

$$\sigma_2 = [x \to f(y) \oplus f(a) \oplus f(b)] \overset{DI}{\Rightarrow}$$

Branch 2.1:

$$\sigma_{2.1} = [x \to f(a), \quad y \to b]$$

$$\Delta_{2.1} = \{f(y) \oplus f(a) \neq^? 0\}$$

Branch 2.2:

$$\sigma_{2.2} = [x \to f(y) \oplus f(a) \oplus f(b)]$$

$$\Delta_{2.2} = \{f(y) \oplus f(a) \neq^? 0, \quad f(y) \oplus f(b) \neq^? 0\}$$

$\sigma_{2.1}$ is another asymmetric unifier.
$\sigma_{2.2}$ is not an asymmetric unifier. However, we can not apply any rules so far to Branch 2.2.

Elimination Instantiation (EI)

The second instantiation rule ***Elimination Instantiation*** introduced in [23] is used to solve the case when there is a conflict between some original variable x and some support variable v.

Elimination Instantiation (EI)

$$\boxed{\frac{[x \mapsto v \oplus S] \cup \sigma \parallel \Upsilon \parallel \Delta}{([x \mapsto S] \cup \sigma) \circ \theta \parallel \Upsilon\theta \parallel \Delta\theta}}$$

where $\theta = \{v \mapsto 0\}, x$ and y are in conflict in Γ for some y. The rule is applied only if $y\sigma = v \oplus S'$ with S' having at least one subterm.

Because v maps to 0, all pairs (v, s) in Υ will be removed from Υ.

Example 4.3.5. *Continuing Example 4.3.3.*

$$\Gamma = \{x \oplus y \oplus z =^?_\downarrow a, \quad x \oplus y =^?_\downarrow x \oplus y, \quad z \oplus a =^?_\downarrow z \oplus a\}$$

$$\sigma = [x \to y \oplus z \oplus a]$$

Solution:

For each branch, we have no rules except Elimination Instantiation applicable. In Branch 1.1, x has a conflict at v_1'' in $x \to v_1'' \oplus v$, so we let $v_1'' \to 0$ and get:
Branch 1.1.1:

$$\sigma_{1.1.1} = [x \to v, \quad y \to a \oplus v_2', \quad z \to v_2' \oplus v]$$

$$\Upsilon_{1.1.1} = \{(v, a), (v, v_2'), (v_2', v), (v_2', a)\}$$

Here $\sigma_{1.1.1}$ is an asymmetric unifier.
In Branch 1.2, x has a conflict at v_1' in $x \to v_1' \oplus v \oplus a$, so we let $v_1' \to 0$ and get:

$$\sigma_{1.2.1} = [x \to v \oplus a, \quad y \to v_2', \quad z \to v_2' \oplus v]$$

$$\Upsilon_{1.2.1} = \{(v, v_2'), (v_2', v)\}$$

Here $\sigma_{1.2.1}$ is an asymmetric unifier, which is equivalent to $\sigma_{1.1.1}$.
In Branch 2.1, x has a conflict at v_1''' in $x \to v_1'''$, so we let $v_1''' \to 0$ and get:

$$\sigma_{2.1.1} = [x \to 0, \quad y \to a \oplus v_2, \quad z \to v_2]$$

$$\Upsilon_{2.1.1} = \{(v_2, a)\}$$

Here $\sigma_{2.1.1}$ is not an asymmetric unifier.
In Branch 2.2, x has a conflict at v_1 in $x \to v_1 \oplus a$, so we let $v_1 \to 0$ and get:

$$\sigma_{2.2.1} = [x \to a, \quad y \to v_2, \quad z \to v_2]$$

$$\Upsilon_{2.2.1} = \phi$$

$\sigma_{2.2.1}$ is an asymmetric unifier, which is an instance of $\sigma_{1.1.1}$.

4.4 Algorithms

Zhiqiang's inference system has three parts: Splitting, Branching and Instantiation, [1]. The algorithm given here is based in Zhiqiang's work. The SEARCHING Algorithm, Algorithm 4, calls the three sub-algorithms 1, 2, and 3, for splitting, branching and instantiate, respectively, and gives outputs a complete set of asymmetric unifiers. The rules in three parts are applied separately by the three sub-algorithms.

Algorithm 1: SPLITTING Algorithm

input :

- the unification problem Γ;
- an XOR unifier σ;
- a constraint set Υ;
- a disequation set Δ.

output:

- a set of triple Σ, which contains all the results by applying rules.

```
1  begin
2      if given σ is an asymmetric unifier of Γ then
3          return {σ ‖ Υ ‖ Δ};
4      else
5          Check whether the Splitting rule is applicable;
6      end
7      if Yes then
8          Apply Splitting rule to σ ‖ Υ ‖ Δ and get σ' ‖ Υ' ‖ Δ'. And return
           SPLITTING(Γ, σ', Υ', Δ');
9      else
10         return BRANCHING(Γ, {σ ‖ Υ ‖ Δ}, ∅);
11     end
12 end
```

Algorithm 2: BRANCHING Algorithm

input :

- the unification problem Γ;
- a set of triples Θ which stores all the branches which are waiting for being applied Branching rules;
- a set of triples Σ which stores all the results after applying branching rules.

output:

- a set of triple Σ, which represent the result of applying rules in Branching Part.

begin
  if Σ *has only one member* $\sigma \parallel \Upsilon \parallel \Delta$ *and* σ *is an asymmetric unifier of* Γ **then**
    return Σ;
  if Θ *is empty* **then**
    return INSTANTIATION(Γ, Σ);
  else
    pick up a member $\sigma \parallel \Upsilon \parallel \Delta$ from Θ;
  end
  if σ *is an asymmetric unifier* **then**
    return $\{\sigma \parallel \Upsilon \parallel \Delta\}$;
  if *Non-Variable Branching rule is applicable* **then**
    apply Non-Variable Branching to $\sigma \parallel \Upsilon \parallel \Delta$ and get $\sigma' \parallel \Upsilon' \parallel \Delta'$ and $\sigma'' \parallel \Upsilon'' \parallel \Delta''$, **return** BRANCHING$(\Gamma, \Theta \cup \{\sigma' \parallel \Upsilon' \parallel \Delta', \sigma'' \parallel \Upsilon'' \parallel \Delta''\}, \Sigma)$;
  if *Variable Branching rule is applicable* **then**
    apply Variable Branching to $\sigma \parallel \Upsilon \parallel \Delta$ and get $\sigma' \parallel \Upsilon' \parallel \Delta'$ and $\sigma'' \parallel \Upsilon'' \parallel \Delta''$, **return** BRANCHING$(\Gamma, \Theta \cup \{\sigma' \parallel \Upsilon' \parallel \Delta', \sigma'' \parallel \Upsilon'' \parallel \Delta''\}, \Sigma)$;
  if *Useless-Variable Branching rule is applicable* **then**
    apply Useless-Variable Branching to $\sigma \parallel \Upsilon \parallel \Delta$ and get $\sigma' \parallel \Upsilon' \parallel \Delta'$ and $\sigma'' \parallel \Upsilon'' \parallel \Delta''$, **return** BRANCHING$(\Gamma, \Theta \cup \{\sigma' \parallel \Upsilon' \parallel \Delta', \sigma'' \parallel \Upsilon'' \parallel \Delta''\}, \Sigma)$;
  else
    Add $\sigma \parallel \Upsilon \parallel \Delta$ to Σ, and **return** BRANCHING$(\Gamma, \Theta\ \{\sigma \parallel \Upsilon \parallel \Delta\}, \Sigma)$;
  end
end

Algorithm 3: INSTANTIATION Algorithm

input :

- the unification problem Γ;
- a set of triples Σ.

output:

- a set of triple Σ', which represent the result of applying rules in Instantiation Part.

begin
  Given an empty set of triple Σ', for each member $\sigma \parallel \Upsilon \parallel \Delta$ of Σ, **if** *Decomposition Instantiation is applicable* **then**
    apply Decomposition Instantiation to $\sigma \parallel \Upsilon \parallel \Delta$ and get $\sigma_1 \parallel \Upsilon_1 \parallel \Delta_1, \ldots, \sigma_n \parallel \Upsilon_n \parallel \Delta_n$. For each $\sigma_i \parallel \Upsilon_i \parallel \Delta_i$, check whether σ_i is an asymmetric unifier of Γ;
  if *it is an asymmetric unifier* **then**
    add it to Σ' and go to next member;
  else
    union SPLITTING$(\Gamma, \sigma_i, \Upsilon_i, \Delta_i)$ with Σ';
  end
  if *Elimination Instantiation is applicable* **then**
    apply Elimination Instantiation to $\sigma \parallel \Upsilon \parallel \Delta$ until it is not applicable and get $\sigma' \parallel \Upsilon' \parallel \Delta'$. Check whether σ' is an asymmetric unifier of Γ;
  if *it is* **then**
    add the triple to Σ' and go to next member;
  else
    u
  end
  nion BRANCHING $(\Gamma, \{\sigma' \parallel \Upsilon' \parallel \Delta'\}, \emptyset)$ with Σ';
  Return Σ'.
end

Algorithm 4: SEARCHING Algorithm

input :

- An asymmetric unification problem Γ;
- A standard unifier σ of Γ;

begin

1. Let $\Sigma' =$ SPLITTING $(\Gamma, \sigma, \emptyset, \emptyset)$
2. Let $\Sigma_\sigma = \{\sigma | \sigma \parallel \Upsilon \parallel \Delta \in \Sigma\}$.
3. **Return** Σ_σ.

end

4.5 Correctness

This section is taken from [23]. The detailed proofs of termination, soundness and completeness are given in [1].

4.5.1 Termination

For **termination**, it is necessary to prove that the algorithm does not generate cycles and does not introduce infinitely new variables after application of any inference rule. Throughout the algorithm, only the bounded number of new variables are introduced by various rules. The Splitting and Branching rules introduce new variables and these are applied only finitely often. And instantiation rules are applied only finitely often to avoid generating cycles.

Once it is proved that the algorithm only introduces finitely new variables, the proof of termination becomes easier. As splitting rule is used to introduce new variables just to replace original variables by support variables. Also elimination rules is used to introduce new variables to remove the conflicts between the original variable and other. It only needs to be made sure that the two instantiation rules cannot be applied infinitely often. The Decomposition instantiation rule reduces the number of simple terms and elimination Instantiation rule reduces the size of the triple since variables get instantiated to 0 to eliminate the conflicts.

4.5.2 Soundness

To prove the **soundness** of the algorithm, it's need to prove that if any of the inference rule from the set of inference rules, generates an asymmetric *XOR* unifier, then this

asymmetric unifier need to be either equivalent to an *XOR* unifier or an instance of an *XOR* unifier.

4.5.3 Completeness

For the proof of **completeness** we need to show that no asymmetric *XOR* unifier is dropped by the algorithm during the whole process .Every inference rule drops only those instances of an *XOR* unifier which are not asymmetric.

Chapter 5

Undecidability of *ACh*-unification and bounded *ACh*-unification

The equational theory ACh is defined over a signature Σ that includes the associative-commutative operator $+$, and a unary symbol h i.e., homomorphism and *(h distributes over addition)*.

$$\begin{aligned} x + (y + z) &= (x + y) + z && \text{(Associativity)} \\ x + y &= y + x && \text{(Commutativity)} \\ h(x + y) &= h(x) + h(y) && (h \text{ distributes over addition}) \end{aligned}$$

Unification modulo the equational theory ACh is a well-known undecidable unification problem [9]. This undecidability happens because of the addition of a homomorphic function symbol h, allows a reduction from equational problems in the structure of positive naturals, an undecidable restriction of Hilbert's tenth problem, to ACh unification problems. If the composition of the function symbol h has some defined bound then we can solve the problem between this bound. This is called **bounded *ACh* unification**. In this bounded version, the bound will be applied to the numbers of applications of h, i.e., the number of times h can be recursively applied to a term and the solution for that term must satisfy this bound. There is no bound on the number of occurrences of h in a term and also there is no bound on the number of occurrences of the $+$ symbol, i.e., the $+$ symbol can be applied an unlimited number of times to a term. This bound on applications of h will be denoted by the letter κ.

To solve a bounded ACh unification problem, we need to know about h-height of a term, i.e., the number of h symbols recursively applied to each other. Then we need to find a solution for that bounded h-height. This h-height restriction is just for the solution, not for the ACh unification problem Γ, also not for the h-depth of a variable, i.e., the number of h symbols on top of that variable in a term. A set of inference rules for bounded

ACh unification problem needs to check h-depth of variables more often because if the h-depth of any variable does not satisfy the bound κ then the problem has no solution in that given bound.

In this chapter Narendran's proof on the undecidability of the *ACh* unification problem [9] is presented in detail and the bounded *ACh* unification method introduced by [11] is discussed.

5.1 *ACh* Unification Problem is Undecidable

We discuss all details on undecidability of the *ACh* unification problem as introduced by Narendran in [9]. Following Narendran's approach, first the undecidability of the unification problem for the theory *ACUh* is proved and then adapted to the theory *ACh*. The theory *ACUh* includes in its signature a constant 0 for the unity of $+$, over which $h(0) = 0$ holds.

5.1.1 Some auxiliary lemmas

To prove that *ACh* **unification problem is undecidable**, we need to understand some lemmas given in [9].

Lemma 5.1.1 ([9]). *Let $x \in \{x_1, \ldots, x_n\}$ be an indeterminate. Then the solution set of*

$$(x-1)Y = Z - 1$$

over $\mathbb{N}[x_1, \ldots, x_n]$ is $\{Y = x^{k-1} + x^{k-2} + \cdots + 1, Z = x^k | k \geq 0\}$.

Proof. For $k = 0$, one has $Y = 0$ and $Z = 1$. For $k > 0$, instantiating the left-hand side of the equation with the proposed solution one has the following chain of equalities.

$$\begin{aligned}
&(x-1)(x^{k-1} + x^{k-2} + \cdots + 1) = \\
&(x^k + x^{k-1} + \ldots + x) - (x^{k-1} + x^{k-2} + \cdots + 1) = \\
&x^k - 1
\end{aligned}$$

To satisfy the equation $(x-1)Y = Z - 1$ the value of the polynomial Z must evaluate to 1 at $x = 1$. Since Z cannot have any negative coefficient, Z should be a monomial of the form x^k and with coefficient 1. □

Lemma 5.1.2 ([9]). *Let $x \in \{x_1, \ldots, x_n\}$ be an indeterminate, and d is a positive integer. Then the solution set of*

$$(x^d - 1)Y = Z - 1$$

over $\mathbb{N}[x_1, \ldots, x_n]$ is $\{Y = x^{d(k-1)} + x^{d(k-2)} + \cdots + 1, Z = x^{dk} | k \geq 0\}$.

Proof. As in the previous lemma, when $k = 0$ the equation holds for any $d > 0$. For $k > 0$, the result is proved instantiating the left-hand side of the equation with the given solutions for Y, through the following chain of equalities.

$$\begin{aligned}
&(x^d - 1)(x^{d(k-1)} + x^{d(k-2)} + \cdots + 1) = \\
&x^{d+d(k-1)} + x^{d+d(k-2)} + \cdots + x^d - (x^{d(k-1)} + x^{d(k-2)} + \cdots + 1) = \\
&x^{dk} + x^{d(k-1)} + \cdots + x^d - (x^{d(k-1)} + x^{d(k-2)} + \cdots + 1) = \\
&x^{dk} - 1
\end{aligned}$$

This lemma is a generalization of the above lemma. By the same argument as in the above lemma, i.e., since Z cannot have any negative coefficient, to satisfy the equation $(x^d - 1)Y = Z - 1$ the value of the polynomial Z must evaluate to 1 at $x = 1$. Z is a monomial i.e., x^{dk} and the coefficient of Z is 1. □

Lemma 5.1.3 ([9])**.** *Let $x \in \{x_1, \ldots, x_n\}$ be an indeterminate, and $v \in \mathbb{N}[x_1, \ldots, x_n]$. The equations*

1. $(x - 1)U_1 = W_1 - 1$
2. $(x^2 - 1)U_2 = W_2 - 1$
3. $(x - 1)U_3 = U_2 - U_1$
4. $(x - 2)Y_1 = U_1 - v$
5. $(x^2 - 2)Y_2 = U_2 - v$

have a solution over $\mathbb{N}[x_1, \ldots, x_n]$ if and only if $v = 2^k - 1$ for some $k \geq 0$

Proof. For if direction:

By lemmas 5.1.1 and 5.1.2 the solution set for the first equation is $\{U_1 = x^{k-1} + x^{k-2} + \cdots + 1, W_1 = x^k \mid k \in \mathbb{N}\}$ and for the second equation is $\{U_2 = x^{2(j-1)} + x^{2(j-2)} + \cdots + 1, W_2 = x^{2j} \mid j \in \mathbb{N}\}$.

The third equation at $x = 1$ becomes

$$\begin{aligned}
0 &= U_2 - U_1 \\
U_1 &= U_2
\end{aligned}$$

Thus, $k = j$, since at $x = 1$, $U_1 = x^{k-1} + x^{k-2} + \cdots + 1 = k$, and $U_2 = x^{2(j-1)} + x^{2(j-2)} + \cdots + 1 = j$.

For the fourth equation, at $x = 2$ we have $0 = U_1 - v$, thus, $v = U_1$. So,

$$v = x^{k-1} + x^{k-2} + \cdots + 1, \text{ at } x = 2$$

This is the geometric series $\sum_{i=0}^{k-1} x^i$, which at $x = 2$ gives:

$$v = 2^k - 1.$$

Now, for the last equation at $x = \sqrt{2}$,

$$\begin{aligned} 0 &= U_2 - v \\ v &= U_2 \\ v &= x^{2(j-1)} + x^{2(j-2)} + \cdots + 1 \end{aligned}$$

This is the geometric series $\sum_{i=0}^{k-1} (x^2)^i$, which at $x = \sqrt{2}$ gives:

$$v = (\sqrt{2}^2)^k - 1 = 2^k - 1$$

For only if direction:

We must now prove that if $v = 2^k - 1$, then all of the above equations have a solution over $\mathbb{N}[x_1, \ldots, x_n]$. We will do that by making smart choices of $U_1, U_2, Y_1, Y_2, U_3, W_1$ and W_2.

- For the first equation, set $U_1 = x^{k-1} + x^{k-2} + \ldots + 1$ and $W_1 = x^k$ and you have satisfied the first equation.
- For the second equation, set $U_2 = x^2(k-1) + x^2(k-2) + \ldots + 1$ and $W_2 = x^2 k$ and you have satisfied the second equation.
- For the third equation, set $U_3 = 0$ and you have satisfied the third equation.
- To satisfy the fourth equation, you must find a polynomial Y_1 such that $(x-2)Y_1 = U_1 - v$.
  For $k = 0$, $U_1 = 0$, $v = 0$ implies $Y_1 = 0$.
  For $k = 1$, $U_1 = 1$, $v = 1$ implies $Y_1 = 0$.
  For $k = 2$, $U_1 = x + 1$, $v = 3$ implies $Y_1 = 1$.
  For $k = 3$, $U_1 = x^2 + x + 1$, $v = 7$ implies $Y_1 = x + 3$
  ...
  Generalization of the above calculation gives a formula that holds for $k > 1$ i.e.,

$$Y_1 = \sum_{1 \le n \le k-1} [(2^n - 1)(x^{k-1-n})]$$

- To satisfy the fifth equation, you must find a polynomial Y_2 such that $(x^2-2)Y_2 = U_2 - v$.
  For $k=0$, $U_2=0$, $v=0$ implies $Y_2=0$.
  For $k=1$, $U_2=1$, $v=1$ implies $Y_2=0$.
  For $k=2$, $U_2=x^2+1$, $v=3$ implies $Y_2=1$.
  For $k=3$, $U_2=x^4+x^2+1$, $v=7$ implies $Y_2=x^2+3$
  ...
  Generalization of the above calculation gives a formula that holds for $k>1$ i.e.,

$$Y_2 = \sum_{1\leq n\leq k-1} [(2^n-1)(x^{2(k-1-n)})]$$

□

Corollary 5.1.4 ([9]). *Let $x \in \{x_1,\ldots,x_n\}$ be an indeterminate, and $v_1 \in \mathbb{N}[x_1,\ldots,x_n]$. The equations*

1. $(x-1)U_1 = W_1 - 1$
2. $(x^2-1)U_2 = W_2 - 1$
3. $(x-1)U_3 = U_2 - U_1$
4. $(x-2)Y_1 = U_1 - V$
5. $(x^2-2)Y_2 = U_2 - V$
6. $v_1 + V_2 = V$

have a solution over $\mathbb{N}[x_1,\ldots,x_n]$ if and only if v_1 is a natural number.

Proof. For if direction:

By the above lemma we have $V = 2^k - 1$ that by sixth equation, it implies

$$v_1 + V_2 = 2^k - 1$$

2^k-1 is a natural number, so the sum of v_1+V_2 must be a natural number, this is only possible when v_1 and V_2 are also a natural numbers.

For only if direction:

If v_1 is a natural number, then you can pick k and V_2 in such a way that satisfy the equation $v_1+V_2 = 2^k-1$ for some $k \geq 0$. Then, applying the above lemma, we conclude that the equations have a solution over $\mathbb{N}[x_1,\ldots,x_n]$. □

Lemma 5.1.5 ([9]). *Let $x \in \{x_1, \ldots, x_n\}$ be an indeterminate, and v_1 and v_2 be non-negative integers. The equations*

1. $(x-1)Y_1 = Z - 1$
2. $(x-1)Y_2 = Y_1 - v_1$
3. $(x-1)Y_3 = Y_2 - v_2$

have a solution over $\mathbb{N}[x_1, \ldots, x_n]$ if and only if $v_2 = \dfrac{v_1(v_1-1)}{2}$.

Proof. For if direction:

First, the case $k \geq 3$ is considered. By Lemma 5.1.1, $Z = x^k$ and Y_1 has to be $x^{k-1} + x^{k-2} + \cdots + 1$, for some $k \geq 0$. Now, evaluating the second equation at $x = 1$, we have $v_1 = Y_1$, and $Y_1 = k$, thus, $v_1 = k$. Then, the second equation becomes:

$$(x-1)Y_2 = (x^{k-1} + x^{k-2} + \cdots + 1) - k$$

that implies

$$Y_2 = \frac{x^{k-1} + x^{k-2} + \cdots + 1 - k}{(x-1)}$$

By polynomial division, when $k > 2$, we will get,

$$Y_2 = x^{k-2} + 2x^{k-3} + 3x^{k-4} + 4x^{k-5} + \frac{5x^{k-5} + \cdots + (k-1)x^{k-(k-1)} + 1 - k}{(x-1)}$$

The last term of the remainder gives:

$$(k-1)x^{k-(k-1)} + 1 - k = (k-1)x + 1 - k = (k-1)(x-1)$$

Thus,

$$Y_2 = x^{k-2} + 2x^{k-3} + \ldots + (k-2)x^2 + (k-1)$$

At $x = 1$, we have

$$Y_2 = 1 + 2 + \cdots + (k-2) + (k-1) = \frac{k(k-1)}{2}$$

Now for the third equation at $x = 1$ we have,

$$v_2 = Y_2$$

Thus,

$$v_2 = \frac{k(k-1)}{2} = \frac{v_1(v_1-1)}{2}$$

Therefore, $v_1 = k$.

To finish the proof, for the cases $k \leq 2$, direct calculations are given.

$k = 0$ By Lemma 5.1.1, $Y_1 = 0$ and $Z = 1$. Then, by the second equation at $x = 1$, one has that $v_1 = 0$. Also, $Y_2 = 0$ because the right-hand side of the second equation does not contain any monomial x^k. Similarly, by the third equation one has that $Y_3 = 0$ and $v_2 = 0$.

$k = 1$ By Lemma 5.1.1, $Y_1 = 1$ and $Z = x$. Thus, by the second equation at $x = 1$, $v_1 = 1$ and $Y_2 = 0$ by similar reason that has been given before. By the third equation when $x = 1$, $v_2 = Y_2 = 0$.

$k = 2$ By Lemma 5.1.1, $Y_1 = x + 1$ and $Z = x^2$. Thus, the second equation becomes,

$$(x-1)Y_2 = x + 1 - v_1$$

If Y_2 is monomial i.e., x, the left-hand side is quadratic, then Y_2 must be equal to 1. Then, the equation becomes $(x-1) = (x+1) - v_1$ which implies that $v_1 = 2$.

In the third equation, the right-hand side is not a polynomial because $Y_2 = 1$ and v_2 is also not a polynomial. So, to satisfy the equality Y_3 must be equal to 0. This implies that $v_2 = Y_2 = 1$. Thus equation

$$v_2 = \frac{v_1(v_1-1)}{2}$$

holds.

For only if direction:

We assume

$$v_2 = \frac{v_1(v_1-1)}{2}$$

and prove that the equations have a solution over $N[x_1, \ldots, x_n]$

If $v_1 = 0$, then $v_2 = 0$. By putting $Z = 1, Y_1 = 0, Y_2 = 0$ and $Y_3 = 0$ the equations are solved.

If $v_1 = 1$ then $v_2 = 0$. By putting, $Z = x, Y_1 = 1, Y_2 = 0$ and $Y_3 = 0$ the equations are solved.

If $v_1 > 1$, pick $Z = x^k, Y_1 = x^{k-1} + \ldots + x + 1$, where $k = v_1$. Notice that this solves the first equation.

let's prove that $Y_1 - v_1$ can be written as $(x-1)Y_2$, for some Y_2.

Notice that, by dividing Y_1 by $(x-1)$ we obtain:

$Y_1 = (x-1)(x^{k-2} + 2x^{k-3} + \ldots + k - 1) + k$

Since $k = v_1$, then

$Y_1 - v_1 = [(x-1)(x^{k-2} + 2x^{k-3} + \ldots + k - 1) + k] - v1 = (x-1)(x^{k-2} + 2x^{k-3} + \ldots + k - 1)$

By picking

$Y_2 = (x^{k-2} + 2x^{k-3} \ldots + k - 1)$

we get that $(x-1)Y_2 = Y_1 - v_1$ and the second equation is solved.

We must now solve the third equation by proving that $Y_2 - v_2$ is divisible by $x - 1$

This is indeed the case, observe from the following examples:

For $k = 2 : v_1 = 2, v_2 = 1$ and $Y_2 = 1$. By picking $Y_3 = 0$, we solve the third equation. For $k = 3 : v_1 = 3, v_2 = 3$ and $Y_2 = x + 2 = (x-1)1 + 3$. Since $v_2 = 3$, this means that $Y_2 - v_2 = (x-1)1$. By picking $Y_3 = 1$, we solve the third equation.

For $k = 4 : v_1 = 4, v_2 = 6$ and $Y_2 = x^2 + 2x + 3 = (x-1)(x+3) + 6$. Since $v_2 = 6$, this means that $Y_2 - v_2 = (x-1)(x+3)$. By picking $Y_3 = (x+3)$ we solve the third equation.

For $k = 5 : v_1 = 5, v_2 = 10$ and $Y_2 = x^3 + 2x^2 + 3x + 4 = (x-1)(x^2 + 3x + 6) + 10$. Since $v_2 = 10$, this means that $Y_2 - v_2 = (x-1)(x^2 + 3x + 6)$. By picking $Y_3 = (x^2 + 3x + 6)$ we solve the third equation.

Take the polynomial $(x^2 + 3x + 6)$ of the case $k = 5$. Notice that its coefficients are : $1, 3$ and 6. These are the values of v_2 for the cases $k = 2, 3$ and 4 respectively.

In general, for $k = l + 3$: $Y_2 - v_2 = (x-1)(a_l x^l + a_{l-1} x^{l-1} + \ldots + a_0)$ where $a_n = \frac{(k-n-l)(k-n-2)}{2}$ for $0 \leq n \leq m$.

□

5.1.2 Reduction from Hilbert's tenth problem

In this section, as in [9], it is proved that Hilbert's tenth problem reduces into the problem of solvability of linear equations over the ring $\mathbb{N}[x_1, \ldots, x_n]$.

Hilbert's tenth problem will be formulated over equations that belong to any one of the following forms.

- $x_i = a$ where a is natural,
- $x_i + y_j = z_k$,
- $x_i y_j = z_k$

By lemma 5.1.5 squaring is available and the following formulation of Hilbert's thenth problem is possible.

Lemma 5.1.6 ([9]). *The following problem is undecidable.*
A set of Diophantine equations S where each equation is either a linear equation or an equation of the form $x = y^2$ is solvable over $\mathbb{N}$?

Theorem 5.1.7 ([9]). *Solvability of linear equations over the ring $\mathbb{N}[x_1, \ldots, x_n]$ is undecidable for all $n \geq 1$.*

Proof. Consider a set S of Diophantine equations as given in Lemma 5.1.6. Let $\{z_1, \ldots, z_m\}$ be the set of variables in S.

1. Using Corollary 5.1.4 and providing a system of six equations with new variables as below, it can be guaranteed that the solution of each z_i, for $i = 1 \ldots, m$, is a natural.

   (a) $(x-1)U_1 = W_1 - 1$

   (b) $(x^2-1)U_2 = W_2 - 1$

   (c) $(x-1)U_3 = U_2 - U_1$

   (d) $(x-2)Y_1 = U_1 - V$

   (e) $(x^2-2)Y_2 = U_2 - V$

   (f) $z_1 + \ldots + z_m + V_2 = V$

   Indeed, Corollary 5.1.4 implies that solutions for $z_1 + \ldots + z_m$, but this also implies that each z_i should be a natural.

2. Take the linear equations in S as they are.

3. For each equation of the form $z_i = z_j^2$ in S, we use Lemma 5.1.5 in order to rewrite the problem as a linear system, as explained below.

Notice that we are adding a system of equations for each equation of the form $z_i = z_j^2$ that we are simulating here. For every pair (i, j) we want to use different polynomials $Y_{k,1}$, Z_k and so on. The way we do this, is by associating a different k to every pair (i, j), by setting $k = mi + j$.

A system of equations as given in Lemma 5.1.5 is given for each quadratic equation.

$$(x-1)Y_{k,1} = Z_k - 1$$

$$(x-1)Y_{k,2} = Y_{k,1} - z_j$$

$$(x-1)Y_{k,3} = Y_{k,2} - V_k$$

$$z_i = 2V_k + z_j$$

Then, by Lemma 5.1.5 solutions satisfy $2V_k = z_j(z_j - 1)$, and thus, by the last equation, $z_j^2 = z_i$.

This equation forces V_k to be also a natural number, since by the systems of equations related with Corollary 5.1.4, z_i and z_j are already forced to be naturals.

Let T be the obtained set of linear equations over $\mathbb{N}[x_1, \ldots, x_n]$. Then, T has a solution over $\mathbb{N}[x_1, \ldots, x_n]$ if and only if S has a solution over $\mathbb{N}$. By Lemma 5.1.6, finding if S has a solution over $\mathbb{N}$ is undecidable. Hence, we conclude that finding if T has a solution over $\mathbb{N}[x_1, \ldots, x_n]$ is undecidable. □

5.1.3 From solvability of linear equations over $\mathbb{N}[x]$ to *ACUh*

The equational theory *ACUh* is defined over a signature Σ that includes the associative-commutative operator $+$, the unity and a unary symbol h with the homomorphic properties below.

$$\begin{aligned} h(x + y) &= h(x) + h(y) && (h \text{ distributes over addition}) \\ h(0) &= 0 && (h \text{ distributes over unity}) \end{aligned}$$

Theorem 5.1.8 ([10])**.** *Solvability of linear equations over* $\mathbb{N}[x]$ *is reducible to the unifiability problem for ACUh.*

Proof. This is proved by Nutt in his paper [10]. Figures 1 and 2 of Nutt's papers shows how you relate an unification algorithm for a monoidal theory with solving linear equations over semirings. Also, an example given by Nutt in his paper at page 19 tells how *ACUh* is related to the semiring $\mathbb{N}[x]$. □

In Nutt's paper [10], the theory *ACUh* is what Narendran, in his paper [9], calls *AC1h*.

Corollary 5.1.9 ([9])**.** *ACUh unification is undecidable.*

Proof. Suppose *ACUh* unification is decidable. Then, by Theorem 5.1.8, we get that solvability of linear equations over $\mathbb{N}[x]$ is also decidable. However, this contradicts Theorem 5.1.7. Therefore, we conclude that *ACUh* unification is undecidable. □

5.1.4 *ACh* Unification is Undecidable

The theory *ACh* is the same as *ACUh* without unity, and only with the equation below for the operator h.

$$h(x + y) = h(x) + h(y) \qquad (h \text{ distributes over addition})$$

The proof of undecidability of *ACh* unification can be obtain by making some changes in the proof of undecidability of *ACUh* unification. *ACh* unification is also related to solving linear equations over $\mathbb{N}[x]$, but with the restriction that no variable can take the value 0. So, consider the structure $\mathbb{N}^+[x_1, \ldots, x_n] = \mathbb{N}[x_1, \ldots, x_n] - \{0\}$.

From Lemma 5.1.1 if we take the solutions of the form $\{Y = x^{k-1} + \ldots + 1, Z = x^k\}$, for the equation,

$$(x-1)Y = Z - 1$$

we cannot take $k = 0$ as is the structure $\mathbb{N}^+[x_1, \ldots, x_n]$, solution $Y = 0$ and $Z = 1$ is not possible.

In *ACh* the above linear equation is related with the following unification problem

$$h(y) + a =_{ACh} y + z$$

The relation interprets xY as $h(y)$, and 1 as a constant a: $xY + 1 = Y + Z$. Possible solutions of the unification problem will explain this more precisely. Since we have no unity in *ACh*, the unifier

$$\{z \mapsto a, y \mapsto 0\}$$

for the above unification problem is not possible. But other unifiers are possible:

$$\{z \mapsto h^k(a), y \mapsto h^{k-1}(a) + h^{k-2}(a) \ldots + a\}, \text{ for } k \geq 1.$$

Similarly, notice that the linear equation $(x^d - 1)Y = Z - 1$ in the structure $\mathbb{N}^+[x_1, \ldots, x_n]$ is related with the following unification problem in

$$h^d(y) + a = y + z$$

Since the unity does not belong to *ACh*, the unifier $\{z \mapsto a, y \mapsto 0\}$ is not possible, but solutions of the form below do.

$$\{z \mapsto h^{dk}(a), y \mapsto h^{d(k-1)}(a) + h^{d(k-2)}(a) + \ldots + a\}, \text{ for } k \geq 1.$$

We need to prove that the Lemmas of section 5.1.1 also hold for the structure $\mathbb{N}^+[x_1, \ldots, x_n]$.

Lemma 5.1.10 ([9]). *Let $x \in \{x_1, \ldots, x_n\}$ be an indeterminate. Then the solution set of*

$$(x-1)Y = Z - 1$$

over $\mathbb{N}^+[x_1, \ldots, x_n]$ is $\{Y = x^{k-1} + x^{k-2} + \cdots + 1, Z = x^k \mid k \geq 1\}$.

Proof. The reason given above for the linear equation $(x-1)Y = Z-1$ excludes from the solutions in $\mathbb{N}[x_1,\ldots,x_n]$ given in Lemma 5.1.1, the solution for $k=0$. Then, the set of solutions in $\mathbb{N}^+[x_1,\ldots,x_n]$ is $\{Y = x^{k-1}+x^{k-2}+\cdots+1, Z = x^k \mid k \geq 1\}$. □

Lemma 5.1.11 ([9]). *Let $x \in \{x_1,\ldots,x_n\}$ be an indeterminate, and d be a non-zero natural number. Then the solution set of*

$$(x^d-1)Y = Z-1$$

over $\mathbb{N}^+[x_1,\ldots,x_n]$ is $\{Y = x^{d(k-1)}+x^{d(k-2)}+\cdots+1, Z = x^{dk} \mid k \geq 1\}$.

Proof. As given in the proof of Lemma 5.1.2, all solutions of the form below are unifiers, but the solution for $k=0$, that is $\{Y \mapsto 0, Z \mapsto 1\}$ should be omitted since 0 does not belong to the structure $\mathbb{N}^+[x_1,\ldots,x_n]$.
The solution set is thus:

$$\{Y = x^{d(k-1)}+x^{d(k-2)}+\cdots+1, Z = x^{dk} \mid k \geq 1\}.$$

□

Lemma 5.1.12 ([9]). *Let $x \in \{x_1,\ldots,x_n\}$ be an indeterminate, and $v \in \mathbb{N}^+[x_1,\ldots,x_n]$. The equations*

1. $(x-1)U_1 = W_1 - 1$
2. $(x^2-1)U_2 = W_2 - 1$
3. $(x-1)U_3 = U_2 - U_1$
4. $(x-2)Y_1 = U_1 - v$
5. $(x^2-2)Y_2 = U_2 - v$

are solvable over $\mathbb{N}^+[x_1,\ldots,x_n]$ if and only if $v = 2^k-1$ for some $k \geq 2$.

Proof. By lemmas 5.1.10 and 5.1.11 the solution set for the first equation is $\{U_1 = x^{k-1}+x^{k-2}+\cdots+1, W_1 = x^k \mid k \geq 1\}$ and the solution set for second equation is $\{U_2 = x^{2(j-1)}+x^{2(j-2)}+\cdots+1, W_2 = x^{2j} \mid j \geq 1\}$.

By the third equation, as in the proof of Lemma 5.1.3, we took $x=1$ the expression $(x-1)U_3$ became 0; thus, $k=j$. Notice that having $x-1$ equal to zero is not an issue; what is restricted in $\mathbb{N}^+[x_1,\ldots,x_n]$ is to take $x=0$ or $U_3=0$.

By the fourth and fifth equations, as in Lemma 5.1.3, $v = 2^k-1$, for $k \geq 1$. But, k cannot be equal to 1. If we take $k=1$, the fourth equation becomes $(x-2)Y_1 = 1-v$,

which is not possible since the left-hand side is a polynomial with x (because you cannot put Y_1 equal to 0 anymore), and the right-hand side is a polynomial without x. Therefore, $k \geq 2$. □

From the Corollary 5.1.4 we can have the adaptation of corollary given below

Corollary 5.1.13. *Let $x \in \{x_1, \ldots, x_n\}$ be an indeterminate, and $v_1 \in \mathbb{N}^+[x_1, \ldots, x_n]$. The equations*

1. $(x-1)U_1 = W_1 - 1$
2. $(x^2-1)U_2 = W_2 - 1$
3. $(x-1)U_3 = U_2 - U_1$
4. $(x-2)Y_1 = U_1 - V$
5. $(x^2-2)Y_2 = U_2 - V$
6. $v_1 + V_2 = V$

have a solution over $\mathbb{N}^+[x_1, \ldots, x_n]$ if and only if v_1 is a positive natural number.

Lemma 5.1.14 ([9])**.** *Let $x \in \{x_1, \ldots, x_n\}$ be an indeterminate, and v_1 and v_2 be non-negative integers. The equations*

1. $(x-1)Y_1 = Z - 1$
2. $(x-1)Y_2 = Y_1 - v_1$
3. $(x-1)Y_3 = Y_2 - v_2$

are solvable over $\mathbb{N}^+[x_1, \ldots, x_n]$ if and only if $v_1 \geq 3$ and $v_2 = \dfrac{v_1(v_1-1)}{2}$. Thus $v_2 \geq 3$ and $2v_2 + v_1 = v_1^2$.

Proof. For $k \geq 3$, the proof is same as for Lemma 5.1.5. Notice that since $v_1 = k$, this implies $v_1 \geq 3$ and also that $v_2 \geq 3$.

The cases $k = 0, 1$ and 2 are not possible. The case $k = 0$ is not possible by the first equation and by Lemma 5.1.10.

For the case $k = 1$, as in the proof of Lemma 5.1.5, $Y_1 = 1$ and $Z = x$, which by the second equation at $x = 1$, implies $(x-1)Y_2 = 1 - v_1$. Which cannot be satisfied for every value of x, since $(1 - v_1)$ does not contain any monomial x^k while $(x-1)Y_2$ is a polynomial that certainly contain x. To satisfy the equation, Y_2 must be equal to zero that gives rise to a contradiction. As we cannot take $Y_2 = 0$.

Finally, for the case $k = 2$, as in the proof of Lemma 5.1.5, $Y_1 = x + 1$. Thus, the second equation becomes $(x-1)Y_2 = x + 1 - v_1$. At $x = 1$ this equation implies that $v_1 = 2$ and then that $v_2 = 1$, then it becomes $(x-1)Y_2 = x-1$, which implies that $Y_2 = 1$.

From this, the third equation becomes $(x-1)Y_3 = Y_2 - v_2$. Which cannot possibly be satisfied for every value of x since $Y_2 - v_2$ is a number and $(x-1)Y_3$ is a polynomial that certainly contain x. To satisfy the equation, Y_3 must be equal to zero that gives rise to a contradiction. As we cannot take $Y_3 = 0$. □

The all above lemmas will help to prove the undecidability of *ACh* unification.

Reduction from Hilbert's tenth problem

By following the Section 5.1.2 and by making some small changes we can prove that solvability of linear equations over the structure $\mathbb{N}^+[x_1, \ldots, x_n]$ is undecidable.

From Lemma 5.1.6 we can have the following lemma.

Lemma 5.1.15. *The following problem is undecidable.*

A set of Diophantine equations S is solvable over $\mathbb{N} - \{0, 1, 2\}$ where each equation is either a linear equation or an equation of the form $x = y^2$ is solvable over $\mathbb{N}^+$?

From Theorem 5.1.7 we have the following theorem.

Theorem 5.1.16. *Solvability of linear equations over the structure $\mathbb{N}^+[x_1, \ldots, x_n]$ is undecidable for all solutions greater or equal than 3.*

Proof. Basically, the proof of this theorem is almost adaptation of the proof of the Theorem 5.1.7, but the main difference is that we are now working with the structure $\mathbb{N}^+[x_1, \ldots, x_n]$ instead of $\mathbb{N}[x_1, \ldots, x_n]$. In the proof of 5.1.7, Corollary 5.1.4 was used in proof and Lemma 5.1.5 was used to simulate the equation of the form $z_i = z_j^2$, In this proof instead of Lemma 5.1.5, we can use Lemma 5.1.14 and the adapted version of Corollary 5.1.4. So, by using Lemma 5.1.14, we will add the constraint that the solutions z_i are greater or equal than 3. Because of this, as compared to the Lemma 5.1.6, we have the restriction that the solutions of Lemma 5.1.15 have to be greater or equal than 3. □

Reduction to the theory of *ACh*

By following the Section 5.1.3 and by making some small changes we can see the *ACh* unification problem is undecidable.

From Theorem 5.1.8 we can have a following theorem.

Theorem 5.1.17. *Solvability of linear equations over $\mathbb{N}^+[x]$ is reducible to the unifiability problem for ACh.*

Proof. The proof can be done by making small adaptations from the proof of Theorem 5.1.8. When we are working with $\mathbb{N}^+[x]$ instead of $\mathbb{N}[x]$, we are not including the neutral element i.e., 0 in the set of naturals $\mathbb{N}$. But in the unification theory this corresponds to a theory without the property of unity. Therefore, the fact that we are working with the structure $\mathbb{N}^+[x]$ can be translated to the theory *ACh* as the semiring $\mathbb{N}[x]$ can be translated to the theory *ACUh*.

□

Theorem 5.1.18 ([9])**.** *ACh unification is undecidable.*

Proof. Let suppose *ACh* unification is decidable, then by theorem 5.1.17, this would mean that solvability of linear equations over $\mathbb{N}^+$ is also decidable. This however contradicts Theorem 5.1.16. So, *ACh* unification is undecidable. □

5.2 Bounded *ACh* Unification

In this subsection we will discuss the bounded *ACh* unification approach introduced in [11]. We present the inference rules given in [11], examples of their application and propose a procedure based on these rules. Proofs of termination, correctness and completion, presented in [11], is not included in the discussion.

5.2.1 Important Definitions

Definition 17 [Graph $\mathbb{G}(\Gamma)$ [11]]

Let Γ be a unification problem. A graph $\mathbb{G}(\Gamma)$ is a graph where each node represents a variable in Γ and each edge represents a function symbol in Γ. In an equation $y =^? f(x_1, \ldots, x_n)$, where f is a function symbol with $n \geq 1$, the graph $\mathbb{G}(\Gamma)$ contains n edges $y \xrightarrow{f} x_1, y \xrightarrow{f} x_2 \ldots, y \xrightarrow{f} x_n$. For a constant symbol c, similarly in an equation $y =^? c$, the graph $\mathbb{G}(\Gamma)$ contains a vertex y. Finally, the graph $\mathbb{G}(\Gamma)$ contains two vertices y and x if an equation $y =^? x$ is in Γ.

Definition 18 [h-Depth [11]]

Let Γ be a unification problem and x be a variable that occurs in Γ. Let h be a unary symbol and f be a symbol with arity greater than or equal to 1 in Γ. Then h-depth of a variable x is the maximum number of h-symbol along a path to x in

$\mathbb{G}(\Gamma)$, and it is denoted by $h_d(x, \Gamma)$. i,e,.

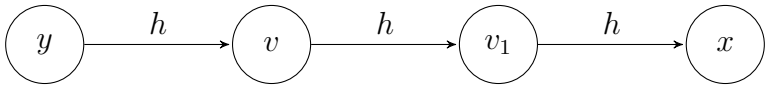
$$h_d(x, \Gamma) := max\{h_{dh}(x, \Gamma), h_{df}(x, \Gamma), 0\},$$

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 $h_{df}(x, \Gamma) := max\{h_d(y, \Gamma) \mid \text{there exist } f \neq h \text{ such that } y \xrightarrow{f} x \text{ is in } \mathbb{G}(\Gamma)\}$

Example 5.2.1.

$$\Gamma = \{y =^? h(v), v =^? h(v_1), v_1 =^? h(x)\}.$$

Solution: *So, the graph* $\mathbb{G}(\Gamma)$ *will be*



Here h**-depth** *of* x *is* 3 *as there are three edges of* h *from* y *to* x, h**-depth** *of* v_1 *is* 2 *as there are two edges of* h *from* y *to* v_1, h**-depth** *of* v *is* 1 *as there is one edge of* h *from* y *to* v *in the graph* $\mathbb{G}(\Gamma)$.

Definition 19 [h-Depth Set [11]]

Let Γ be a set of equations and V be the set of variables occurring in Γ. Then the h-Depth Set of Γ, denoted as Δ, is a set whose elements are pairs of a variable from V and a non-negative integer of the form (x, c), such that $x \in V$ and $c = h_d(x, \Gamma)$. The maximum value of the h-depth in Δ, is the maximum of all c values, and is denoted by $MaxVal(\Delta)$, i.e.,

$$MaxVal(\Delta) = max\{c \mid (x, c) \in \Delta \text{ for some } x\}.$$

Example 5.2.2.

$$\Gamma = \{y =^? h(v), v =^? h(v_1), v_1 =^? h(x)\} \quad and \quad V = \{x, y, v, v_1\}$$

Solution: *Then we can see from the graph* $\mathbb{G}(\Gamma)$ *in previous example that the* h*-depth set of* Γ *is*

$$\Delta = \{(x, 3), (y, 0), (v, 1), (v_1, 2)\}$$

Then we can see from the h*-depth set* Δ*, the maximum value of* h*-depth in* Δ *is 3, i,e,.*

$$MaxVal(\Delta) = \{3\}$$

Definition 20 [h-Height [11]]

Let Γ be a unification problem and t be a term that occurs in Γ. Then the h-height of t is defined as

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

Definition 21 [Bounded ACh Unifier [11]]

A κ-bounded unifier or κ-bounded solution of the unification problem $\Gamma = \{s_1 =^?_{ACh} t_1, \ldots, s_n =^?_{ACh} t_n\}$ modulo ACh 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 $1 \leq i \leq n$. If $\kappa \in N$ is a bound on the h-depth of the variables then to satisfy this bound, $MaxVal(\Delta) \leq \kappa$ must hold.

The Example 5.2.13 given after presentation of the inference rule Bound Check 5.2.3 will help to understand the notion of bounded ACh unifier.

5.2.2 Inference System

The inference system introduced by Eeralla and Lynch in [11] will be discussed and examples presented to illustrate how these rules work to compute bounded solutions for ACh unification problems. This system consist of set of inference rules over triples as illustrated below.

$$\frac{\Gamma \parallel \Delta \parallel \sigma}{\Gamma' \parallel \Delta' \parallel \sigma'}$$

This means that if something matches the top of any rule then it can be replaced with the bottom of that rule. The triples consist of:

- Γ, a set of equations of a unification problem modulo ACh theory,
- Δ, an h-Depth set of Γ, and
- σ, a set of substitutions.

By using Eeralla and Lynch inference rules in [11], an equational unification problem can be transformed into other. Generally, the inference procedure is based on priority of rules but when there is no priority restriction, any rule from the set of rules can be

applied. Initially, Γ is given as the (non-empty) set of equations to be solved; Δ, as a set of pairs of the form $(x, 0)$, one of such pairs for each x that belongs to the variables in Γ; and σ, as the identity substitution. Then, to solve Γ, the inference rules are applied until one gets either the most general unifier σ or $\perp$ for no solution.

To solve the unification problem modulo *ACh*, Eeralla and Lynch's approach in [11] maintains all equations in Γ in flattened form. Flattened equations are of one of the following forms:

- $x =^? y$ called **Var-equations**.
- $x =^? h(y)$ called ***h*-equations**.
- $x =^? y_1 + \cdots + y_n$ called **+-equations**.
- $x =^? f(x_1, \ldots, x_n)$ called **free-equations**.

Where x, y, y_i, and x_i are variables, and f is any free function symbol with $n \geq 0$. If the given equations in the unification problem are not in flattened form, then by using inference rules for flattening provided in [11], they are easily converted into equivalent equations in the required flattened form.

5.2.3 Inference Rules

First, all flattening rules in [11] will be discussed here. These rules are used to put any equation given in the input set of equations Γ into flattened form. According to the type of equation we can use one of the flattening rules below.

Flattening Rules

1. **Flatten Both Sides (FBS)**

$$\frac{\{t_1 =^? t_2\} \cup \Gamma \parallel \Delta \parallel \sigma}{\{v =^? t_1, v =^? t_2\} \cup \Gamma \parallel \{(v, 0)\} \cup \Delta \parallel \sigma}$$, if t_1 and $t_2 \notin V$. And v is a fresh variable.

2. **Flatten Left + (FL+)**

$$\frac{\{t =^? t_1 + t_2\} \cup \Gamma \parallel \Delta \parallel \sigma}{\{t =^? v + t_2, v =^? t_1\} \cup \Gamma \parallel \{(v, 0)\} \cup \Delta \parallel \sigma}$$, if $t_1 \notin V$. And v is a fresh variable.

3. **Flatten Right + (FR+)**

$$\boxed{\frac{\{t =^? t_1 + t_2\} \cup \Gamma \parallel \Delta \parallel \sigma}{\{t =^? t_1 + v, v =^? t_2\} \cup \Gamma \parallel \{(v,0)\} \cup \Delta \parallel \sigma}}, \text{ if } t_2 \notin V. \text{ And } v \text{ is a fresh variable.}$$

4. **Flatten Under h (FUh)**

$$\boxed{\frac{\{t_1 =^? h(t)\} \cup \Gamma \parallel \Delta \parallel \sigma}{\{t_1 =^? h(v), v =^? t\} \cup \Gamma \parallel \{(v,0)\} \cup \Delta \parallel \sigma}}, \text{ if } t \notin V. \text{ And } v \text{ is a fresh variable.}$$

Example 5.2.3. *Solve the unification problem:*

$$\{h(x) + y = (z + w)\}$$

Solution: *For simplicity, only the set of equations Γ is consider here, and not the full triple.*

$$\begin{aligned}
&\{\underbrace{h(x) + y = z + w}\}||\{(x,0),(y,0),(z,0),(w,0)\} && \stackrel{FBS}{\Longrightarrow}\\
&\{\underbrace{v = h(x) + y}, v = z + w\}||\{(v,0),(x,0),(y,0),(z,0),(w,0)\} && \stackrel{FL+}{\Longrightarrow}\\
&\{v = v_1 + y, v_1 = h(x), v = z + w\}||\{(v,0),(v_1,0),(x,0),(y,0),(z,0),(w,0)\}
\end{aligned}$$

Each equation in the set $\{v = v_1 + y, v_1 = h(x), v = z + w\}$ is in the flattened form of the given problem $\{h(x) + y = (z + w)\}$.

Example 5.2.4. *Solve the unification problem:*

$$\{h(x) + h(y) = (x + y) + (w + z)\}$$

Solution: *For simplicity, only the set of equations is considered here, and not the full triple. Let $\Delta_0 = \{(x,0),(y,0),(w,0),(z,0)\}$*

$$\begin{aligned}
&\{\underbrace{h(x) + h(y) = (x + y) + (w + z)}\}||\Delta_0 && \stackrel{FBS}{\Longrightarrow}\\
&\{\underbrace{v = h(x) + h(y)}, v = (x + y) + (w + z)\}||\{(v,0)\} \cup \Delta_0 && \stackrel{FL+}{\Longrightarrow}\\
&\{\underbrace{v = v_1 + h(y)}, v_1 = h(x), v = (x + y) + (w + z)\}||\{(v,0),(v_1,0)\} \cup \Delta_0 && \stackrel{FR+}{\Longrightarrow}\\
&\{v = v_1 + v_2, v_1 = h(x), v_2 = h(y), \underbrace{v = (x + y) + (w + z)}\}|| && \\
&\{(v,0),(v_1,0),(v_2,0)\} \cup \Delta_0 && \stackrel{FL+}{\Longrightarrow}
\end{aligned}$$

$$\{v = v_1 + v_2, v_1 = h(x), v_2 = h(y), \underbrace{v = v_3 + (w + z)}, v_3 = (x + y)\}||$$
$$\{(v, 0), (v_1, 0), (v_2, 0), (v_3, 0)\} \cup \Delta_0 \qquad \overset{FR+}{\Longrightarrow}$$
$$\{v = v_1 + v_2, v_1 = h(x), v_2 = h(y), v = v_3 + v_4, v_3 = (x + y), v_4 = (w + z)\}||$$
$$\{(v, 0), (v_1, 0), (v_2, 0), (v_3, 0), (v_4, 0)\} \cup \Delta_0$$

Each equation in the set $\{v = v_1+v_2, v_1 = h(x), v_2 = h(y), v = v_3+v_4, v_3 = (x+y), v_4 = (w + z)\}$ *is in the flattened form of the given problem* $\{h(x) + h(y) = (x + y) + (w + z)\}$.

Update h-Depth Set

Eerlla and Lynch also proposed an inference rule to update the h-depth set Δ ([11]).

Update h-depth set (Uh)

$$\boxed{\frac{\{x =^? h(y)\} \cup \Gamma \parallel \{(x, c_1), (y, c_2)\} \cup \Delta \parallel \sigma}{\{x =^? h(y)\} \cup \Gamma \parallel \{(x, c_1), (y, c_1 + 1)\} \cup \Delta \parallel \sigma}}, \text{ if } c_2 < (c_1 + 1).$$

Example 5.2.5 (Taken from [11])**.** *Solve the unification problem:*

$$\{y =^? h(h(h(x)))\}$$

The flattened form of this unification problem is the input problem in Example 5.2.2. Here, the solution is given to show how to put the unification problem into the flattened form.

Solution: *We only consider the pair* $\Gamma||\Delta$*, but not* σ *since it does not change after applying flattening rules.*

First, rules for flattening are applied.

$$\underbrace{\{y =^? h(h(h(x)))\}}||\{(x, 0), (y, 0)\} \qquad \overset{FUh}{\Longrightarrow}$$
$$\{y =^? h(v), \underbrace{v = h(h(x))}\}||\{(v, 0), (x, 0), (y, 0)\} \qquad \overset{FUh}{\Longrightarrow}$$
$$\{y =^? h(v), v = h(v_1), v_1 = h(x)\}||\{(v, 0), (v_1, 0), (x, 0), (y, 0)\}$$

Then, the rule to update the h-depth set Δ *will be applied.*

$$\{y =^? h(v), v =^? h(v_1), v_1 =^? h(x)\}||\{\underbrace{(v, 0)}, (v_1, 0), (x, 0), (y, 0)\} \quad \overset{Uh}{\Longrightarrow}$$
$$\{y =^? h(v), v =^? h(v_1), v_1 =^? h(x)\}||\{(v, 1), \underbrace{(v_1, 0)}, (x, 0), (y, 0)\} \quad \overset{Uh}{\Longrightarrow}$$
$$\{y =^? h(v), v =^? h(v_1), v_1 =^? h(x)\}||\{(v, 1), (v_1, 2), \underbrace{(x, 0)}, (y, 0)\} \quad \overset{Uh}{\Longrightarrow}$$
$$\{y =^? h(v), v =^? h(v_1), v_1 =^? h(x)\}||\{(v, 1), (v_1, 2), (x, 3), (y, 0)\}$$

Update + Rule (U+)

Updating rules proposed in [11], are used to update the set Δ according to +-equations.

1. $\dfrac{\{x =^? y_1 + y_2\} \cup \Gamma \parallel \{(x,c),(y_1,c_1),(y_2,c_2)\} \cup \Delta \parallel \sigma}{\{x =^? y_1 + y_2\} \cup \Gamma \parallel \{(x,c),(y_1,c),(y_2,c_2)\} \cup \Delta \parallel \sigma}$, if $c_1 < c$.

2. $\dfrac{\{x =^? y_1 + y_2\} \cup \Gamma \parallel \{(x,c),(y_1,c_1),(y_2,c_2)\} \cup \Delta \parallel \sigma}{\{x =^? y_1 + y_2\} \cup \Gamma \parallel \{(x,c),(y_1,c_1),(y_2,c)\} \cup \Delta \parallel \sigma}$, if $c_2 < c$.

Example 5.2.6. *Solve the unification problem:*

$$\{x = x_1 + x_2 + x_3, y = h(h(x))\}$$

Solution: *For simplicity, the substitution σ is omitted and we only consider the pair $\Gamma||\Delta$.*

$$\begin{aligned}
&\{x =^? x_1 + x_2 + x_3, \underbrace{y =^? h(h(x))}\}||\{(x_1,0),(x_2,0),(x_3,0),(x,0),(y,0)\} && \overset{FUh}{\Longrightarrow}\\
&\{\underbrace{x =^? x_1 + x_2 + x_3}, y =^? h(v), v =^? h(x)\}||\{(x_1,0),(x_2,0),(x_3,0),(x,0),\\
&(y,0),(v,0)\} && \overset{FR+}{\Longrightarrow}\\
&\{x =^? x_1 + v_1, v_1 = x_2 + x_3, y =^? h(v), v =^? h(x)\}||\{(x_1,0),(x_2,0),(x_3,0),\\
&\underbrace{(x,0)},(y,0),\underbrace{(v,0)},(v_1,0)\} && \overset{Uh}{\Longrightarrow}{}^2\\
&\{x =^? x_1 + v_1, v_1 = x_2 + x_3, y =^? h(v), v =^? h(x)\}||\{\underbrace{(x_1,0)},(x_2,0),(x_3,0),\\
&(x,2),(y,0),(v,1),\underbrace{(v_1,0)}\} && \overset{U+}{\Longrightarrow}{}^2\\
&\{x =^? x_1 + v_1, v_1 = x_2 + x_3, y =^? h(v), v =^? h(x)\}||\{(x_1,2),\underbrace{(x_2,0)},\underbrace{(x_3,0)},\\
&(x,2),(y,0),(v,1),(v_1,2)\} && \overset{U+}{\Longrightarrow}{}^2\\
&\{x =^? x_1 + v_1, v_1 = x_2 + x_3, y =^? h(v), v =^? h(x)\}||\{(x_1,2),(x_2,2),(x_3,2),\\
&(x,2),(y,0),(v,1),(v_1,2)\}
\end{aligned}$$

As there are two edges of h from y to x in the graph $\mathbb{G}(\Gamma)$, h-depth of x is 2 thats why h-depths of x_1, x_2, x_3 and v_1 are also updated according to x by using the Update + (U+) Rule.

Splitting Rule (S)

The splitting rule introduced by Eeralla and Lynch in [11], deals with the function symbol h. To solve h-equations of the type $h(y) =^? x_1 + x_2$, we need to write y as the sum of two new variables $y = v_1 + v_2$, where v_1 and v_2 are fresh variables.

Splitting

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

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

Example 5.2.7. *Solve the unification problem:*

$$\{h(y) =^? x_1 + x_2\}$$

Solution: *Consider pair $\Gamma||\Delta$, since rules for obtaining σ are not introduced yet.*

$$\begin{aligned}
&\{\underbrace{h(y) =^? x_1 + x_2}\}||\{(y,0),(x_1,0),(x_2,0)\} && \overset{FBS}{\Longrightarrow}\\
&\{v =^? h(y), v =^? x_1 + x_2\}||\{\underbrace{(y,0)},(x_1,0),(x_2,0),(v,0)\} && \overset{Uh}{\Longrightarrow}\\
&\{\underbrace{v =^? h(y), v =^? x_1 + x_2}\}||\{(y,1),(x_1,0),(x_2,0),(v,0)\} && \overset{S}{\Longrightarrow}\\
&\{v =^? h(y), y =^? v_1 + v_2, x_1 =^? h(v_1), x_2 =^? h(v_2)\}||\\
&\{(y,1),(x_1,0),(x_2,0),(v,0),\underbrace{(v_1,0)},\underbrace{(v_2,0)}\} && \overset{Uh}{\Longrightarrow}{}^{2}\\
&\{v =^? h(y), y =^? v_1 + v_2, x_1 =^? h(v_1), x_2 =^? h(v_2)\}||\\
&\{(y,1),(x_1,0),(x_2,0),(v,0),(v_1,1),(v_2,1)\}
\end{aligned}$$

We pause this example here until the rules for σ will be introduced.

Trivial

This rule, introduced in [11], is used to eliminate all trivial equations from the given problem Γ.

$$\frac{\{t =^? t\} \cup \Gamma \parallel \Delta \parallel \sigma}{\Gamma \parallel \Delta \parallel \sigma}$$

Variable Elimination (VE)

These rules, introduced in [11], are used to find the most general unifier. Variable Elimination rule (VEoc) is applied when no other rule can be applied.

1. **Variable Elimination with different variables (VEv)**

$$\boxed{\frac{\{x =^? y\} \cup \Gamma \parallel \Delta \parallel \sigma}{\Gamma\{x \mapsto y\} \parallel \Delta \parallel \sigma\{x \mapsto y\} \cup \{x \mapsto y\}}}, \text{ if } x \text{ and } y \text{ are distinct variables.}$$

2. **Variable Elimination with occur check (VEoc)**

$$\boxed{\frac{\{x =^? t\} \cup \Gamma \parallel \Delta \parallel \sigma}{\Gamma\{x \mapsto t\} \parallel \Delta \parallel \sigma\{x \mapsto t\} \cup \{x \mapsto t\}}}, \text{ if } t \notin V \text{ and } x \text{ does not occur in } t.$$

Example 5.2.8. *Continuing Example 5.2.7:*

$$\{h(y) =^? x_1 + x_2\}$$

the triple below, where $\Delta = \{(y,1),(x_1,0),(x_2,0),(v,0),(v_1,1),(v_2,1)\}$*, was obtained.*

$$\{v =^? h(y), y =^? v_1 + v_2, x_1 =^? h(v_1), x_2 =^? h(v_2)\}||\Delta||\emptyset$$

Solution:

$$\begin{aligned}
&\{\underbrace{v =^? h(y)}, y =^? v_1 + v_2, x_1 =^? h(v_1), x_2 =^? h(v_2)\}||\Delta||\emptyset && \overset{VEoc}{\Longrightarrow}\\
&\{\underbrace{y =^? v_1 + v_2}, x_1 =^? h(v_1), x_2 =^? h(v_2)\}||\Delta||\{v \mapsto h(y)\} && \overset{VEoc}{\Longrightarrow}\\
&\{\underbrace{x_1 =^? h(v_1)}, x_2 =^? h(v_2)\}||\Delta||\{v \mapsto h(v_1 + v_2), y \mapsto v_1 + v_2\} && \overset{VEoc}{\Longrightarrow}\\
&\{\underbrace{x_2 =^? h(v_2)}\}||\Delta||\{v \mapsto h(v_1 + v_2), y \mapsto v_1 + v_2, x_1 \mapsto h(v_1)\} && \overset{VEoc}{\Longrightarrow}\\
&\emptyset||\Delta||\{v \mapsto h(v_1 + v_2), y \mapsto v_1 + v_2, x_1 \mapsto h(v_1), x_2 \mapsto h(v_2)\}
\end{aligned}$$

So, $\{y \mapsto v_1 + v_2, x_1 \mapsto h(v_1), x_2 \mapsto h(v_2)\}$ *is a unifier of the problem* $\{h(y) =^? x_1 + x_2\}$*.*

Decomposition (D)

This rule introduced in [11] is used to decomposes an equation into several sub-equations if the top function symbol matches on both sides.

$$\boxed{\frac{\{x =^? f(s_1,\ldots,s_n), x =^? f(t_1,\ldots,t_n)\} \cup \Gamma \parallel \Delta \parallel \sigma}{\{x =^? f(t_1,\ldots,t_n), s_1 =^? t_1, \ldots, s_n =^? t_n\} \cup \Gamma \parallel \Delta \parallel \sigma}}, \text{ if } f \neq +$$

Example 5.2.9. *Solve the unification problem:*

$$\{f(x_1 + x_2, y_1) =^? f(h(y), y_2)\}$$

Consider Δ *as in Example 5.2.8, and* $\sigma = \{v \mapsto h(y), y \mapsto v_1 + v_2, x_1 \mapsto h(v_1), x_2 \mapsto h(v_2)\}$*, the solution for* $\{h(y) =^? x_1 + x_2\}$*, obtained in that example.*

Solution:

$$\{\underbrace{f(x_1 + x_2, y_1) =^? f(h(y), y_2)}\}||\{(y,0),(x_1,0),(y_1,0),(x_2,0),(y_2,0)\}||\emptyset \overset{FBS}{\Longrightarrow}$$
$$\{\underbrace{x =^? f(x_1 + x_2, y_1), x =^? f(h(y), y_2)}\}||\{(x,0),(y,0),(x_1,0),(y_1,0),(x_2,0),$$
$$(y_2,0)\}||\emptyset \overset{D}{\Longrightarrow}$$
$$\{x =^? f(h(y), y_2), x_1 + x_2 =^? h(y), y_1 =^? y_2\}||\ \{(x,0),(y,0),(x_1,0),(y_1,0),$$
$$(x_2,0),(y_2,0)\}||\emptyset \overset{*}{\Longrightarrow}$$
$$\emptyset||\{(x,0),(y_1,0),(y_2,0)\} \cup \Delta||\sigma \cup \{y_1 \mapsto y_2\}$$

Associative-Commutative unification Rule (ACu)

This rule, as introduced in [11], will call any AC syntactic unification algorithm to unify the AC part of the given problem. We will apply this rule exhaustively generating all different possibilities from the AC operator. All possibilities would be solved to build the complete set of bounded unifiers.

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

where Ψ is the set of all equations with the $+$ symbol on the right hand side; and, Γ is the set of equations not containing a $+$ symbol. In the rule (ACu), $GetsEqs$ is a function that takes a substitution and returns the equational form of that substitution, i.e.,

$$GetsEqs(\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}) = \{x_1 = t_1, \ldots, x_n = t_n\}$$

Finally, *Unify* is a function that returns the complete set of unifiers, $\{\theta_1, \ldots, \theta_n\}$, computed by the selected AC-unification algorithm: $Unify(\Psi) = \{\theta_1, \ldots, \theta_n\}$.

Example 5.2.10. *Solve the unification problem:*

$$\{h(x + u) =^? y + h(x) + z\}$$

For this kind of examples i.e., where the (ACu) rule will be applicable, we need to see for which branches we will have a solution. In particular, in this example we can have several branches after application of the (ACu) rule as below.

- *for the left-hand side of the equation the cases $h(x + u)$ and $h(u + x)$;*
- *for the right-hand side of the equation we will have several possibilities, for instance: $(y + h(x)) + z$, $z + (h(x) + y)$, $h(x) + (y + z)$, $h(x) + (z + y)$, $(y + z) + h(x)$, etc.*

But we will not have a solution for all of above branches. We need to check which branch will provide us a solution for the given problem. Here, we will consider just three branches fixing the left-hand side and changing the right-hand side of the equation and check if they will provide solutions or not.

Consider the first branch $h(x+u) =^? h(x) + (y+z)$

Solution:

$$\{\underbrace{h(x+u) =^? h(x) + (y+z)}\}||\{(x,0),(u,0),(y,0),(z,0)\}||\emptyset \overset{FBS}{\Longrightarrow}$$
$$\{\underbrace{v =^? h(x+u)}, v =^? h(x) + (y+z)\}||\{(v,0),(x,0),(u,0),$$
$$(y,0),(z,0)\}||\emptyset \overset{FUh}{\Longrightarrow}$$
$$\{v =^? h(w), w =^? x+u, \underbrace{v =^? h(x) + (y+z)}\}||$$
$$\{(w,0),(v,0),(x,0),(u,0),(y,0),(z,0)\}||\emptyset \overset{FL+}{\Longrightarrow}$$
$$\{v =^? h(w), w =^? x+u, \underbrace{v =^? v_1 + (y+z)}, v_1 =^? h(x)\}||$$
$$\{(v_1,0),(w,0),(v,0),(x,0),(u,0),(y,0),(z,0)\}||\emptyset \overset{FR+}{\Longrightarrow}$$
$$\{\underbrace{v =^? h(w)}, w =^? x+u, \underbrace{v =^? v_1 + v_2}, v_1 =^? h(x), v_2 =^? y+z\}||$$
$$\{(v_2,0),(v_1,0),(w,0),(v,0),(x,0),(u,0),(y,0),(z,0)\}||\emptyset \overset{S}{\Longrightarrow}$$
$$\{v =^? h(w), w =^? v_3 + v_4, v_1 =^? h(v_3), v_2 =^? h(v_4), w =^? x+u,$$
$$v_1 =^? h(x), v_2 =^? y+z\}||\{\underbrace{(v_4,0)}, \underbrace{(v_3,0)}, (v_2,0),(v_1,0), \underbrace{(w,0)}, (v,0),$$
$$\underbrace{(x,0)}, (u,0),(y,0),(z,0)\}||\emptyset \overset{Uh}{\Longrightarrow}$$
$$\{v =^? h(w), w =^? v_3 + v_4, v_1 =^? h(v_3), \underbrace{v_2 =^? h(v_4)}, w =^? x+u, v_1 =^? h(x),$$
$$\underbrace{v_2 =^? y+z}\}||\{(v_4,1),(v_3,1),(v_2,0),(v_1,0),(w,1),(v,0),(x,1),$$
$$(u,0),(y,0),(z,0)\}||\emptyset \overset{S}{\Longrightarrow}$$
$$\{v =^? h(w), w =^? v_3 + v_4, v_1 =^? h(v_3), v_2 =^? h(v_4), v_4 =^? v_5 + v_6,$$
$$y =^? h(v_5), z =^? h(v_6), w =^? x+u, v_1 =^? h(x)\}||\{\underbrace{(v_6,0)}, \underbrace{(v_5,0)},$$
$$(v_4,1),(v_3,1),(v_2,0),(v_1,0),(w,1),(v,0),(x,1),(u,0),(y,0),(z,0)\}||\emptyset \overset{Uh}{\Longrightarrow}$$
$$\{v =^? h(w), w =^? v_3 + v_4, \underbrace{v_1 =^? h(v_3)}, v_2 =^? h(v_4), v_4 =^? v_5 + v_6,$$
$$y =^? h(v_5), z =^? h(v_6), w =^? x+u, \underbrace{v_1 =^? h(x)}\}||\{(v_6,1),(v_5,1),(v_4,1),$$
$$(v_3,1),(v_2,0),(v_1,0),(w,1),(v,0),(x,1),(u,0),(y,0),(z,0)\}||\emptyset \overset{D}{\Longrightarrow}$$
$$\{v =^? h(w), \underbrace{w =^? v_3 + v_4}, v_2 =^? h(v_4), v_4 =^? v_5 + v_6, y =^? h(v_5), z =^? h(v_6),$$
$$\underbrace{w =^? x+u}, v_1 = h(x), x =^? v_3\}||\{(v_6,1),(v_5,1),(v_4,1),(v_3,1),(v_2,0),$$
$$(v_1,0),(w,1),(v,0),(x,1),(u,0),(y,0),(z,0)\}||\emptyset \overset{AC}{\Longrightarrow}$$

$\{\underbrace{v =^? h(w)}, w =^? x + u, x =^? v_3, u =^? v_4, v_2 =^? h(v_4), v_4 =^? v_5 + v_6,$
$y =^? h(v_5), z =^? h(v_6), v_1 =^? h(x)\}||\{(v_6, 1), (v_5, 1), (v_4, 1), (v_3, 1), (v_2, 0),$
$(v_1, 0), (w, 1), (v, 0), (x, 1), (u, 0), (y, 0), (z, 0)\}||\emptyset$ $\overset{VEoc}{\Longrightarrow}$
$\{\underbrace{w =^? x + u}, x =^? v_3, u =^? v_4, v_2 =^? h(v_4), v_4 =^? v_5 + v_6,$
$y =^? h(v_5), z =^? h(v_6), v_1 =^? h(x)\}||$
$\{(v_6, 1), (v_5, 1), (v_4, 1), (v_3, 1), (v_2, 0),$
$(v_1, 0), (w, 1), (v, 0), (x, 1), (u, 0), (y, 0), (z, 0)\}||$
$\{v \mapsto h(w)\}$ $\overset{VEoc}{\Longrightarrow}$
$\{\underbrace{x =^? v_3, u =^? v_4}, v_2 =^? h(v_4), v_4 =^? v_5 + v_6, y =^? h(v_5), z =^? h(v_6), v_1 =^? h(x)\}||$
$\{(v_6, 1), (v_5, 1), (v_4, 1), (v_3, 1), (v_2, 0), (v_1, 0), (w, 1), (v, 0), (x, 1), (u, 0), (y, 0), (z, 0)\}||$
$\{v \mapsto h(x + u), w \mapsto x + u\}$ $\overset{VEoc^2}{\Longrightarrow}$
$\{v_2 =^? h(v_4), \underbrace{v_4 =^? v_5 + v_6}, y =^? h(v_5), z =^? h(v_6), v_1 =^? h(v_3)\}||$
$\{(v_6, 1), (v_5, 1), (v_4, 1), (v_3, 1), (v_2, 0), (v_1, 0), (w, 1), (v, 0), (x, 1), (u, 0), (y, 0), (z, 0)\}||$
$\{v \mapsto h(v_3 + v_4), w \mapsto v_3 + v_4, x \mapsto v_3, u \mapsto v_4\}$ $\overset{VEoc}{\Longrightarrow}$
$\{v_2 =^? h(v_5 + v_6), \underbrace{y =^? h(v_5)}, \underbrace{z =^? h(v_6)}, v_1 =^? h(v_3)\}||$
$\{(v_6, 1), (v_5, 1), (v_4, 1), (v_3, 1), (v_2, 0), (v_1, 0), (w, 1), (v, 0), (x, 1), (u, 0), (y, 0), (z, 0)\}||$
$\{v \mapsto h(v_3 + v_5 + v_6), w \mapsto v_3 + v_5 + v_6, x \mapsto v_3, u \mapsto v_5 + v_6, v_4 \mapsto v_5 + v_6\}$ $\overset{VEoc^2}{\Longrightarrow}$
$\{v_2 =^? h(v_5 + v_6), v_1 =^? h(v_3)\}||$
$\{(v_6, 1), (v_5, 1), (v_4, 1), (v_3, 1), (v_2, 0), (v_1, 0), (w, 1), (v, 0), (x, 1), (u, 0), (y, 0), (z, 0)\}||$
$\{v \mapsto h(v_3 + v_5 + v_6), w \mapsto v_3 + v_5 + v_6, x \mapsto v_3, u \mapsto v_5 + v_6, v_4 \mapsto v_5 + v_6,$
$y \mapsto h(v_5), z \mapsto h(v_6)\}$

Therefore the above first branch provides us a solution for the problem $h(x + u) =^? h(x) + (y + z)$*:*

$$\sigma = \{x \mapsto v_3, u \mapsto v_5 + v_6, y \mapsto h(v_5), z \mapsto h(v_6)\}$$

Now consider the second branch $h(x + u) =^? y + h(x) + z$

$\{\underbrace{h(x + u) =^? y + h(x) + z}\}||\{(x, 0), (u, 0), (y, 0), (z, 0)\}||\emptyset$ $\overset{FBS}{\Longrightarrow}$
$\{\underbrace{v =^? h(x + u)}, v =^? y + h(x) + z\}||\{(v, 0), (x, 0), (u, 0),$
$(y, 0), (z, 0)\}||\emptyset$ $\overset{FUh}{\Longrightarrow}$
$\{v =^? h(w), w =^? x + u, \underbrace{v =^? y + h(x) + z}\}||$
$\{(w, 0), (v, 0), (x, 0), (u, 0), (y, 0), (z, 0)\}||\emptyset$ $\overset{FL+}{\Longrightarrow}$

$\{v =^? h(w), w =^? x + u, v =^? v_1 + z, \underbrace{v_1 =^? y + h(x)}\}||$

$\{(v_1, 0), (w, 0), (v, 0), (x, 0), (u, 0), (y, 0), (z, 0)\}||\emptyset$ $\overset{FR+}{\Longrightarrow}$

$\{v =^? h(w), w =^? x + u, v =^? v_1 + z, v_1 =^? y + v_2, v_2 =^? h(x)\}||$

$\{(v_2, 0), (v_1, 0), \underbrace{(w, 0)}, (v, 0), \underbrace{(x, 0)}, (u, 0), (y, 0), (z, 0)\}||$ $\overset{Uh}{\Longrightarrow}$

$\{v =^? h(w), \underbrace{w =^? x + u}, v =^? v_1 + z, v_1 =^? y + v_2, v_2 =^? h(x)\}||$

$\{(v_2, 0), (v_1, 0), (w, 1), (v, 0), (x, 1), \underbrace{(u, 0)}, (y, 0), (z, 0)\}||$ $\overset{U+}{\Longrightarrow}$

$\{\underbrace{v =^? h(w)}, w =^? x + u, \underbrace{v =^? v_1 + z}, v_1 =^? y + v_2, v_2 =^? h(x)\}||$

$\{(v_2, 0), (v_1, 0), (w, 1), (v, 0), (x, 1), (u, 1), (y, 0), (z, 0)\}||\emptyset$ $\overset{S}{\Longrightarrow}$

$\{v =^? h(w), w =^? v_3 + v_4, \underbrace{v_1 =^? h(v_3)}, z =^? h(v_4), w =^? x + u, \underbrace{v_1 =^? y + v_2}, v_2 =^? h(x)\}||$

$\{(v_4, 0), (v_3, 0), (v_2, 0), (v_1, 0), (w, 1), (v, 0), (x, 1), (u, 1), (y, 0), (z, 0)\}||\emptyset$ $\overset{S}{\Longrightarrow}$

$\{v =^? h(w), w =^? v_3 + v_4, v_1 =^? h(v_3), v_3 =^? v_5 + v_6, y =^? h(v_5), \underbrace{v_2 =^? h(v_6)}, z =^? h(v_4),$

$w =^? x + u \underbrace{v_2 =^? h(x)}\}||\{(v_6, 0), (v_5, 0), (v_4, 0), (v_3, 0), (v_2, 0), (w, 1), (v_1, 0), (v, 0),$

$(x, 1), (u, 1), (y, 0), (z, 0), \}||\emptyset$ $\overset{D}{\Longrightarrow}$

$\{v_2 =^? h(x), v_6 =^? x, v =^? h(w), \underbrace{w =^? v_3 + v_4}, \underbrace{w =^? x + u}, v_1 =^? h(v_3), v_3 =^? v_5 + v_6,$

$y =^? h(v_5), z =^? h(v_4)\}||\{(v_6, 0), (v_5, 0), (v_4, 0), (v_3, 0), (v_2, 0), (w, 1), (v_1, 0), (v, 0),$

$(x, 1), (u, 1), (y, 0), (z, 0), \}||\emptyset$ $\overset{ACu}{\Longrightarrow}$

$\{w =^? x + u, \underbrace{v_3 =^? x}, v_4 =^? u, v_2 =^? h(x), v_6 =^? x, v =^? h(w), v_1 =^? h(v_3), v_3 =^? v_5 + v_6,$

$y =^? h(v_5), z =^? h(v_4)\}||\{(v_6, 0), (v_5, 0), (v_4, 0), (v_3, 0), (v_2, 0), (w, 1), (v_1, 0), (v, 0),$

$(x, 1), (u, 1), (y, 0), (z, 0), \}||\emptyset$ $\overset{VEoc}{\Longrightarrow}$

$\{w =^? x + u, u =^? v_4, v_2 =^? h(x), \underbrace{v_6 =^? x}, v =^? h(w), v_1 =^? h(x), x =^? v_5 + v_6,$

$y =^? h(v_5), z =^? h(v_4)\}||\{(v_6, 0), (v_5, 0), (v_4, 0), (v_3, 0), (v_2, 0), (w, 1), (v_1, 0), (v, 0),$

$(x, 1), (u, 1), (y, 0), (z, 0), \}||\{v_3 \mapsto x\}$ $\overset{VEoc}{\Longrightarrow}$

$\{w =^? x + u, u =^? v_4, v_2 =^? h(x), v =^? h(w), v_1 =^? h(x), \underbrace{x =^? v_5 + x},$

$y =^? h(v_5), z =^? h(v_4)\}||$

$\{(v_6, 0), (v_5, 0), (v_4, 0), (v_3, 0), (v_2, 0), (w, 1), (v_1, 0), (v, 0),$

$(x, 1), (u, 1), (y, 0), (z, 0), \}||\{v_3 \mapsto x, v_6 \mapsto x\}$ $\overset{OC}{\Longrightarrow}$

$\bot$

The second branch above, fails because x is the variable in the left-hand side and also occurs (non trivially) in the right-hand side of and equation. This will be implemented through an Occur Check (OC) rule to be presented after the example.

Now, consider the third branch for solving the equation $h(x + u) =^? (y + z) + h(x)$.

$\{\underbrace{h(x+u) =^? (y+z)+h(x)}\}||\{(x,0),(u,0),(y,0),(z,0)\}||\emptyset$ $\overset{FBS}{\Longrightarrow}$

$\{\underbrace{v =^? h(x+u)}, v =^? (y+z)+h(x)\}||\{(v,0),(x,0),(u,0),(y,0),(z,0)\}||\emptyset$ $\overset{FUh}{\Longrightarrow}$

$\{v =^? h(w), w =^? x+u, \underbrace{v =^? (y+z)+h(x)}\}||$

$\{(w,0),(v,0),(x,0),(u,0),(y,0),(z,0)\}||\emptyset$ $\overset{FL+}{\Longrightarrow}$

$\{v =^? h(w), w =^? x+u, \underbrace{v =^? v_1+h(x)}, v_1 =^? y+z\}||$

$\{(v_1,0),(w,0),(v,0),(x,0),(u,0),(y,0),(z,0)\}||\emptyset$ $\overset{FR+}{\Longrightarrow}$

$\{v =^? h(w), w =^? x+u, v =^? v_1+v_2, v_1 =^? y+z, v_2 =^? h(x)\}||$

$\{(v_2,0),(v_1,0),\underbrace{(w,0)},(v,0),\underbrace{(x,0)},(u,0),(y,0),(z,0)\}||\emptyset$ $\overset{Uh}{\Longrightarrow}$

$\{v =^? h(w), \underbrace{w =^? x+u}, v =^? v_1+v_2, v_1 =^? y+z, v_2 =^? h(x)\}||$

$\{(v_2,0),(v_1,0),(w,1),(v,0),(x,1),\underbrace{(u,0)},(y,0),(z,0)\}||\emptyset$ $\overset{U+}{\Longrightarrow}$

$\{\underbrace{v =^? h(w)}, w =^? x+u, \underbrace{v =^? v_1+v_2}, v_1 =^? y+z, v_2 =^? h(x)\}||$

$\{(v_2,0),(v_1,0),(w,1),(v,0),(x,1),(u,1),(y,0),(z,0)\}||\emptyset$ $\overset{S}{\Longrightarrow}$

$\{v =^? h(w), w =^? v_3+v_4, \underbrace{v_1 =^? h(v_3)}, v_2 =^? h(v_4), w =^? x+u, \underbrace{v_1 =^? y+z}, v_2 =^? h(x)\}||$

$\{(v_4,0),(v_3,0),(v_2,0),(v_1,0),(w,1),(v,0),(x,1),(u,1),(y,0),(z,0)\}||\emptyset$ $\overset{S}{\Longrightarrow}$

$\{v =^? h(w), w =^? v_3+v_4, v_1 =^? h(v_3), v_3 =^? v_5+v_6, y =^? h(v_5), z =^? h(v_6), v_2 =^? h(v_4),$

$w =^? x+u, v_2 =^? h(x)\}||\{\underbrace{(v_6,0)},\underbrace{(v_5,0)},\underbrace{(v_4,0)},\underbrace{(v_3,0)},(v_2,0),$

$(v_1,0),(w,1),(v,0),(x,1),(u,1),(y,0),(z,0)\}||\emptyset$ $\overset{Uh}{\Longrightarrow}^*$

$\{v =^? h(w), w =^? v_3+v_4, v_1 =^? h(v_3), v_3 =^? v_5+v_6, y =^? h(v_5), z =^? h(v_6), \underbrace{v_2 =^? h(v_4)},$

$w =^? x+u, \underbrace{v_2 =^? h(x)}\}||\{(v_6,1),(v_5,1),(v_4,1),(v_3,1),(v_2,0),$

$(v_1,0),(w,1),(v,0),(x,1),(u,1),(y,0),(z,0)\}||\emptyset$ $\overset{D}{\Longrightarrow}$

$\{v =^? h(w), \underbrace{w =^? v_3+v_4}, v_1 =^? h(v_3), v_3 =^? v_5+v_6, y =^? h(v_5), z =^? h(v_6),$

$\underbrace{w =^? x+u}, v_2 =^? h(x), x =^? v_4\}||\{(v_6,1),(v_5,1),(v_4,1),(v_3,1),(v_2,0),$

$(v_1,0),(w,1),(v,0),(x,1),(u,1),(y,0),(z,0)\}||\emptyset$ $\overset{AC}{\Longrightarrow}$

$\{\underbrace{v =^? h(w)}, w =^? x+u, x =^? v_4, u =^? v_3, v_1 =^? h(v_3), v_3 =^? v_5+v_6, y =^? h(v_5),$

$z =^? h(v_6), v_2 =^? h(x)\}||\{(v_6,1),(v_5,1),(v_4,1),(v_3,1),(v_2,0),$

$(v_1,0),(w,1),(v,0),(x,1),(u,1),(y,0),(z,0)\}||\emptyset$ $\overset{VEoc}{\Longrightarrow}$

$\{\underbrace{w =^? x+u}, x =^? v_4, u =^? v_3, v_1 =^? h(v_3), v_3 =^? v_5+v_6, y =^? h(v_5),$

$z =^? h(v_6), v_2 =^? h(x)\}||$

$\{(v_6,1),(v_5,1),(v_4,1),(v_3,1),(v_2,0),(v_1,0),(w,1),(v,0),(x,1),(u,1),(y,0),(z,0)\}||$

$\{v \mapsto h(w)\}$ $\overset{VEoc}{\Longrightarrow}$

$\{\underbrace{x =^? v_4}, \underbrace{u =^? v_3}, v_1 =^? h(v_3), \underbrace{v_3 =^? v_5 + v_6}, y =^? h(v_5), z =^? h(v_6), v_2 =^? h(x)\}||$
$\{(v_6, 1), (v_5, 1), (v_4, 1), (v_3, 1), (v_2, 0), (v_1, 0), (w, 1), (v, 0), (x, 1), (u, 1), (y, 0), (z, 0)\}||$
$\{v \mapsto h(x + u), w \mapsto x + u\}$ $\overset{VEqc^3}{\Longrightarrow}$
$\{v_1 =^? h(v_5 + v_6), \underbrace{y =^? h(v_5)}, \underbrace{z =^? h(v_6)}, v_2 =^? h(v_4)\}||$
$\{(v_6, 1), (v_5, 1), (v_4, 1), (v_3, 1), (v_2, 0), (v_1, 0), (w, 1), (v, 0), (x, 1), (u, 1), (y, 0), (z, 0)\}||$
$\{v \mapsto h(v_4 + u)), w \mapsto v_4 + u, x \mapsto v_4, u \mapsto v_5 + v_6, v_3 \mapsto v_5 + v_6\}$ $\overset{VEqc^2}{\Longrightarrow}$
$\{v_1 =^? h(v_3), v_2 =^? h(x)\}||$
$\{(v_6, 1), (v_5, 1), (v_4, 1), (v_3, 1), (v_2, 0), (v_1, 0), (w, 1), (v, 0), (x, 1), (u, 1), (y, 0), (z, 0)\}||$
$\{v \mapsto h(v_4 + u)), w \mapsto v_4 + u, x \mapsto v_4, u \mapsto v_5 + v_6, v_3 \mapsto v_5 + v_6, y =^? h(v_5), z =^? h(v_6)\}$

The substitution provides a solution for the problem $h(x + u) =^? (y + z) + h(x)$*:* $\sigma = \{x \mapsto v_4, u \mapsto v_5 + v_6, y \mapsto h(v_5), z \mapsto h(v_6)\}$.

Occur Check Rule (OC)

This rule introduced in [11] is used to check if a variable on the left-hand side of an equation occurs on the right-hand side of the equation. If yes, then the problem has no solution.

$$\boxed{\frac{\{x =^? f(t_1, \ldots, t_n)\} \cup \Gamma \parallel \Delta \parallel \sigma}{\bot}}, \text{ if } x \in Var(f(t_1, \ldots, t_n)\sigma)$$

Example 5.2.11. *As first example see the second branch of Example 5.2.10.*

As a second simple example, solve the unification problem:

$$\{x + y =^? h(y)\}$$

Solution:

$\{\underbrace{x + y =^? h(y)}\}||\{(x, 0), (y, 0)\}||\emptyset$ $\overset{FBS}{\Longrightarrow}$
$\{\underbrace{v =^? x + y}, \underbrace{v =^? h(y)}\}||\{(x, 0), (y, 0), (v, 0)\}||\emptyset$ $\overset{OC}{\Longrightarrow}$
$\bot$

Hence, the problem has no solution.

Clash Rule (C)

The Clash Rule, introduced by Eeralla and Lynch in [11], is used to check if the top symbol on the left-hand side and on the right-hand side is the same. If not, then the

problem has no solution, unless one of them is h and the other is $+$.

$$\boxed{\frac{\{x =^? f(s_1, \ldots, s_m), x =^? g(t_1, \ldots, t_n)\} \cup \Gamma \parallel \Delta \parallel \sigma}{\bot}}, \text{ if } f \notin \{h, +\} \text{ or } g \notin \{h, +\}.$$

Example 5.2.12. *Solve the unification problem:*

$$\{f(x_1, y_1) =^? g(x_2, y_2)\}$$

Solution:

$$\begin{array}{ll} \{\underbrace{f(x_1, y_1) =^? g(x_2, y_2)}\}||\{(x_1, 0), (y_1, 0), (x_2, 0), (y_2, 0)\}||\emptyset & \stackrel{FBS}{\Longrightarrow} \\ \{\underbrace{v =^? f(x_1, y_1)}, \underbrace{v =^? g(x_2, y_2)}\}||\{(x_1, 0), (y_1, 0), (x_2, 0), (y_2, 0), (v, 0)\}||\emptyset & \stackrel{C}{\Longrightarrow} \\ \bot & \end{array}$$

Hence, the problem has no solution.

Bound Check (BC)

This inference rule, introduced in [11], is used to check if a solution exists within the given bound. The Bound Check rule must be applied right after the application of the updating rules of the h-depth set.

$$\boxed{\frac{\Gamma \parallel \Delta \parallel \sigma}{\bot}}, \text{ if } MaxVal(\Delta) > \kappa.$$

Example 5.2.13. *Solve the unification problem considering as bound for the solution* $\kappa = 2$*:*

$$\{h(x) =^? x + y + z\}$$

Solution:

$$\{\underbrace{h(x) =^? x + y + z}\}||\{(x,0),(y,0),(z,0)\}||\emptyset \xRightarrow{FBS}$$
$$\{v =^? h(x), v =^? x + y + z\}||\{\underbrace{(x,0)},(y,0),(z,0),(v,0)\}||\emptyset \xRightarrow{Uh}$$
$$\{\underbrace{v =^? h(x), v =^? x + y + z}\}||\{(x,1),(y,0),(z,0),(v,0)\}||\emptyset \xRightarrow{S}$$
$$\{v =^? h(x), x =^? v_1 + v_2 + v_3, x =^? h(v_1), y =^? h(v_2), z =^? h(v_3)\}||$$
$$\{(x,1),(y,0),(z,0),(v,0),\underbrace{(v_1,0)},\underbrace{(v_2,0)},\underbrace{(v_3,0)}\}||\emptyset \xRightarrow{Uh}$$
$$\{v =^? h(x),\underbrace{x =^? v_1 + v_2 + v_3, x =^? h(v_1)}, y =^? h(v_2), z =^? h(v_3)\}||$$
$$\{(x,1),(y,0),(z,0),(v,0),(v_1,2),(v_2,1),(v_3,1)\}||\emptyset \xRightarrow{S}$$
$$\{v =^? h(x), x =^? h(v_1), v_1 =^? v_{11} + v_{12} + v_{13}, v_1 =^? h(v_{11}), v_2 =^? h(v_{12}),$$
$$v_3 =^? h(v_{13}), y =^? h(v_2), z =^? h(v_3)\}||\{(x,1),(y,0),(z,0),(v,0),(v_1,2),$$
$$(v_2,1),(v_3,1),\underbrace{(v_{11},0)},\underbrace{(v_{12},0)},\underbrace{(v_{13},0)}\}||\emptyset \xRightarrow{Uh}$$

$$\{v =^? h(x), x =^? h(v_1), v_1 =^? v_{11} + v_{12} + v_{13}, v_1 =^? h(v_{11}), v_2 =^? h(v_{12}),$$
$$v_3 =^? h(v_{13}), y =^? h(v_2), z =^? h(v_3)\}||\{(x,1),(y,0),(z,0),(v,0),(v_1,2),$$
$$(v_2,1),(v_3,1),(v_{11},3),(v_{12},2),(v_{13},2)\}||\emptyset \xRightarrow{BC}$$
$$\bot$$

Rule (BC) applies since $MaxVal(\Delta) = 3 > \kappa$. *Thus, the problem* $\{h(x) =^? x + y + z\}$ *has no solution within the given bound* $\kappa = 2$.

Orient Rule (O)

The orient rule, introduced by Eeralla and Lynch in [11], is used to swap the left- and right-hand side of an equation, when the left-hand side is a variable.

$$\boxed{\frac{\{t =^? x\} \cup \Gamma \parallel \Delta \parallel \sigma}{\{x =^? t\} \cup \Gamma \parallel \Delta \parallel \sigma}}, \text{ if } t \text{ is not a variable.}$$

5.2.4 Bounded *ACh* unification Algorithm

The proposed algorithm given below, Algorithm 6, is based on the inference rules introduced by Eeralla and Lynch in his work [11]. According to the inference rules we divided the algorithm in two parts: initially, the input set of equations Γ is flattened with Algo-

rithm 5 that applies only the flattening rules; after that, the main unification Algorithm 6 applies the rest of the inference rules.

Algorithm 5: Procedure Flatten Γ

input : A set of equations Γ

output: A set of equations Γ in flatten form

```
1 begin
2     repeat
3         Apply: Flatten Both Sides;
4         Apply: Flatten Under h;
5         if One application of Flatten Right/Left + does not eliminate a
            possible application of Splitting Rule then
6             Apply: this Flatten Right/Left + Rules;
7         end
8     until none of these rules apply;
9 end
```

Algorithm 6: Procedure for *ACh* bounded Unification

input : A set of equations Γ, a bound κ, an empty set σ and an empty h-depth set Δ.

output: Either a complete set of κ-bounded *ACh* unifiers $\{\sigma_1, \ldots, \sigma_n\}$ or $\perp$ means the problem has no solution.

```
1  begin
2      Apply flattening Algorithm 5 on Γ.
3      repeat
4          repeat
5              Trivial Rule;
6              Splitting Rule;
7              Update h-depth Set Rule;
8              Update + Rule;
9              AC Unification Rule;
10             Orient Rule;
11             Decomposition Rule;
12         until none of these rules apply;
13         repeat
14             if Bound Check Rule applies then return ⊥; /* MaxVal(Δ)>κ */
15             if Occur Check Rule or Clash Rule apply then return ⊥ ;
16         until no longer possible application;
17         repeat
18             Variable Elimination Rule;
19         until no longer the rule applies;
20     until until no rule applies;
21 end
```

Example 5.2.14. *Consider the problem below.*

$$\{h(y) =^? x_1 + x_2 + x_3\}$$

Notice that this example shows that the **Splitting** *rule applies to equations with tuples of variables, but the flatten rules* **FR**+ *and* **FL**+ *apply to additions of tuples of variables until additions of pairs of variables are reached. Thus, it would be interesting to prioritize the application of the* **Splitting** *rule.*

First, consider a derivation in which the **Splitting** *rule is applied directly to the problem above.*

$\{\underbrace{h(y) =^? x_1 + x_2 + x_3}\}||\{(y,0),(x_1,0),(x_2,0),(x_3,0)\}||\emptyset$ $\overset{FBS}{\Longrightarrow}$

$\{\underbrace{v =^? h(y)}, \underbrace{v =^? x_1 + x_2 + x_3}\}||\{(v,0),(y,0),(x_1,0),(x_2,0),(x_3,0)\}||\emptyset$ $\overset{S}{\Longrightarrow}$

$\{v =^? h(y), y =^? v_1 + v_2 + v_3, x_1 =^? h(v_1), x_2 =^? h(v_2), x_3 =^? h(v_3)\}||$
$\{\underbrace{(v_1,0)}, \underbrace{(v_2,0)}, \underbrace{(v_3,0)}, (v,0), \underbrace{(y,0)}, (x_1,0),(x_2,0),(x_3,0)\}||\emptyset$ $\overset{Uh}{\Longrightarrow}$

$\{\underbrace{x =^? h(y)}, y =^? v_1 + v_2 + v_3, x_1 =^? h(v_1), x_2 =^? h(v_2), x_3 =^? h(v_3)\}||$
$\{(v_1,1),(v_2,1),(v_3,1),(v,0),(y,1),(x_1,0),(x_2,0),(x_3,0)\}||\emptyset$ $\overset{VEoc}{\Longrightarrow}$

$\{\underbrace{y =^? v_1 + v_2 + v_3}, x_1 =^? h(v_1), x_2 =^? h(v_2), x_3 =^? h(v_3)\}||\{(v_1,1),(v_2,1),$
$(v_3,1),(v,0),(y,1),(x_1,0),(x_2,0),(x_3,0)\}||\{x \mapsto h(y)\}$ $\overset{VEoc}{\Longrightarrow}$

$\{\underbrace{x_1 =^? h(v_1)}, x_2 =^? h(v_2), x_3 =^? h(v_3)\}||\{(v_1,1),(v_2,1),(v_3,1),(v,0),(y,1)$
$,(x_1,0),(x_2,0),(x_3,0)\}||\{x \mapsto h(v_1 + v_2 + v_3), y \mapsto v_1 + v_2 + v_3\}$ $\overset{VEoc}{\Longrightarrow}$

$\{\underbrace{x_2 =^? h(v_2)}, x_3 =^? h(v_3)\}||\{(v_1,1),(v_2,1),(v_3,1),(v,0),(y,1),(x_1,0),$
$(x_2,0),(x_3,0)\}||\{x \mapsto h(v_1 + v_2 + v_3), y \mapsto v_1 + v_2 + v_3, x_1 \mapsto h(v_1)\}$ $\overset{VEoc}{\Longrightarrow}$

$\{\underbrace{x_3 =^? h(v_3)}\}||\{(v_1,1),(v_2,1),(v_3,1),(v,0),(y,1),(x_1,0),(x_2,0),(x_3,0)\}||$
$\{x \mapsto h(v_1 + v_2 + v_3), y \mapsto v_1 + v_2 + v_3, x_1 \mapsto h(v_1), x_2 \mapsto h(v_2)\}$ $\overset{VEoc}{\Longrightarrow}$

$\emptyset||\{(v_1,1),(v_2,1),(v_3,1),(v,0),(y,1),(x_1,0),(x_2,0),(x_3,0)\}||$
$\{x \mapsto h(v_1 + v_2 + v_3), y \mapsto v_1 + v_2 + v_3, x_1 \mapsto h(v_1), x_2 \mapsto h(v_2), x_3 \mapsto h(v_3)\}$

The solution set for the problem $\{h(y) =^? x_1 + x_2 + x_3\}$ *is*

$$\sigma = \{y \mapsto v_1 + v_2 + v_3, x_1 \mapsto h(v_1), x_2 \mapsto h(v_2), x_3 \mapsto h(v_3)\}$$

Second, if rules **FR**+ *and/or* **FL**+ *are applied before the* **Splitting** *rule, a different derivation, as below, is possible. But in the end the solution is the same modulo renaming of variables.*

$\{\underbrace{h(y) =^? x_1 + (x_2 + x_3)}\}||\{(y,0),(x_1,0),(x_2,0),(x_3,0)\}||\emptyset$ $\overset{FBS}{\Longrightarrow}$

$\{x =^? h(y)\ \underbrace{x =^? x_1 + (x_2 + x_3)}\}||\{(x,0),(y,0),(x_1,0),(x_2,0),(x_3,0)\}||\emptyset$ $\overset{FR+}{\Longrightarrow}$

$\{\underbrace{x =^? h(y)}, \underbrace{x =^? x_1 + v}, v =^? x_2 + x_3\}||\{(v,0),(x,0),(y,0),(x_1,0),(x_2,0),$
$(x_3,0)\}||\emptyset$ $\overset{S}{\Longrightarrow}$

$\{x =^? h(y), y =^? v_1 + v_2, x_1 =^? h(v_1), v =^? h(v_2), v =^? x_2 + x_3\}||$
$\{\underbrace{(v_1,0)}, \underbrace{(v_2,0)}, (x,0), (v.0), \underbrace{(y,0)}, (x_1,0),(x_2,0),(x_3,0)\}||\emptyset$ $\overset{Uh}{\Longrightarrow}$

$\{x =^? h(y), y =^? v_1 + v_2, x_1 =^? h(v_1), \underbrace{v =^? h(v_2)}, \underbrace{v =^? x_2 + x_3}\}||$
$\{(v_1, 1), (v_2, 1), (x, 0), (v.0), (y, 1), (x_1, 0), (x_2, 0), (x_3, 0)\}||\emptyset$ $\overset{S}{\Longrightarrow}$
$\{x =^? h(y), y =^? v_1 + v_2, x_1 =^? h(v_1), v =^? h(v_2), v_2 =^? v_3 + v_4,$
$x_2 =^? h(v_3), x_3 =^? h(v_4)\}||\{\underbrace{(v_3, 0)}, \underbrace{(v_4, 0)}, (v_1, 1), (v_2, 1), (x, 0), (v.0), (y, 1),$
$(x_1, 0), (x_2, 0), (x_3, 0)\}||\emptyset$ $\overset{Uh}{\Longrightarrow}$
$\{\underbrace{x =^? h(y)}, y =^? v_1 + v_2, x_1 =^? h(v_1), v =^? h(v_2), v_2 =^? v_3 + v_4,$
$x_2 =^? h(v_3), x_3 =^? h(v_4)\}||\{(v_3, 1), (v_4, 1), (v_1, 1),$
$(v_2, 1), (x, 0), (v.0), (y, 1), (x_1, 0), (x_2, 0), (x_3, 0)\}||\emptyset$ $\overset{VEoc}{\Longrightarrow}$
$\{\underbrace{y =^? v_1 + v_2}, x_1 =^? h(v_1), v =^? h(v_2), v_2 =^? v_3 + v_4,$
$x_2 =^? h(v_3), x_3 =^? h(v_4)\}||\{(v_3, 1), (v_4, 1), (v_1, 1), (v_2, 1),$
$(x, 0), (v.0), (y, 1), (x_1, 0), (x_2, 0), (x_3, 0)\}||\{x \mapsto h(y)\}$ $\overset{VEoc}{\Longrightarrow}$
$\{\underbrace{x_1 =^? h(v_1)}, v =^? h(v_2), v_2 =^? v_3 + v_4, x_2 =^? h(v_3), x_3 =^? h(v_4)\}||$
$\{(v_3, 1), (v_4, 1), (v_1, 1), (v_2, 1), (x, 0), (v.0), (y, 1), (x_1, 0), (x_2, 0), (x_3, 0)\}||$
$\{x \mapsto h(v_1 + v_2), y \mapsto v_1 + v_2\}$ $\overset{VEoc}{\Longrightarrow}$
$\{\underbrace{v =^? h(v_2)}, v_2 =^? v_3 + v_4, x_2 =^? h(v_3), x_3 =^? h(v_4)\}||$
$\{(v_3, 1), (v_4, 1), (v_1, 1), (v_2, 1), (x, 0), (v.0), (y, 1), (x_1, 0), (x_2, 0), (x_3, 0)\}||$
$\{x \mapsto h(v_1 + v_2), y \mapsto v_1 + v_2, x_1 \mapsto h(v_1)\}$ $\overset{VEoc}{\Longrightarrow}$
$\{\underbrace{v_2 =^? v_3 + v_4}, x_2 =^? h(v_3), x_3 =^? h(v_4)\}||\{(v_3, 1), (v_4, 1), (v_1, 1),$
$(v_2, 1), (x, 0), (v, 0), (y, 1), (x_1, 0), (x_2, 0), (x_3, 0)\}||$
$\{x \mapsto h(v_1 + v_2), y \mapsto v_1 + v_2, x_1 \mapsto h(v_1), v \mapsto h(v_2)\}$ $\overset{VEoc}{\Longrightarrow}$
$\{\underbrace{x_2 =^? h(v_3)}, x_3 =^? h(v_4)\}||\{(v_3, 1), (v_4, 1), (v_1, 1), (v_2, 1), (x, 0), (v.0),$
$(y, 1), (x_1, 0), (x_2, 0), (x_3, 0)\}||\{x \mapsto h(v_1 + v_3 + v_4), y \mapsto v_1 + v_3 + v_4,$
$x_1 \mapsto h(v_1), v \mapsto h(v_3 + v_4), v_2 \mapsto v_3 + v_4, \}$ $\overset{VEoc}{\Longrightarrow}$
$\{\underbrace{x_3 =^? h(v_4)}\}||\{(v_3, 1), (v_4, 1), (v_1, 1), (v_2, 1),$
$(x, 0), (v.0), (y, 1), (x_1, 0), (x_2, 0), (x_3, 0)\}||\{x \mapsto h(v_1 + v_3 + v_4), y \mapsto v_1 + v_3 + v_4,$
$x_1 \mapsto h(v_1), v \mapsto h(v_3 + v_4), v_2 \mapsto v_3 + v_4, x_2 \mapsto h(v_3)\}$ $\overset{VEoc}{\Longrightarrow}$
$\emptyset||\{(v_3, 1), (v_4, 1), (v_1, 1), (v_2, 1), (x, 0), (v.0), (y, 1), (x_1, 0),$
$(x_2, 0), (x_3, 0)\}||\{x \mapsto h(v_1 + v_3 + v_4), y \mapsto v_1 + v_3 + v_4, x_1 \mapsto h(v_1), v \mapsto h(v_3 + v_4),$
$v_2 \mapsto v_3 + v_4, x_2 \mapsto h(v_3), x_3 \mapsto h(v_4)\}$

The solution set for the problem $\{h(y) =^? x_1 + x_2 + x_3\}$ *is*

$$\sigma = \{y \mapsto v_1 + v_3 + v_4, x_1 \mapsto h(v_1), x_2 \mapsto h(v_3), x_3 \mapsto h(v_4)\}$$

Termination, Correctness and Completeness

In [11], Eeralla and Lynch introduced an algorithm to apply their inference rules and proved its termination, correctness and completeness.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This work studies different problems related to equational reasoning modulo equational theories. Problems, as unification, asymmetric unification, disunification are addressed considering operators with algebraic equational properties such as associativity (A), commutativity (C), nilpotence (N), existence of unity (U) and homomorphisms (h).

Initially, we study the paper by Veena Ravishankar, Paliath Narendran and Kimberly A. Gero ([7]) that shows that asymmetric unification seems to be harder than disunification, as the solvability of asymmetric unification modulo $ACUN$ is NP-hard while disunification modulo $ACUN$ is in P. Also, we study the case of ground disunification and ground asymmetric unification modulo $ACUNh$ for which the former is in P while the latter is NP-hard.

Afterwards, we study Zhiqiang Liu PhD thesis and his work with Serdar Erbatur on asymmetric unification for the theory XOR with uninterpreted function symbols ([1], [23]). The theory XOR has the properties of $ACUN$, and is presented by an equational axiomatization for AC and rewriting rules for the UN. We studied the algorithm by Zhiqiang Liu for the asymmetric unification problem modulo the theory XOR. The asymmetric unification was first investigated by Serdar Erbatur in his work [23] about the decidability of asymmetric unification. Erbatur shows that how variant based unification can be adapted to obtain an asymmetric unification algorithm. Erbatur used the technique to develop an asymmetric unification algorithm by converting the symmetric algorithm to an asymmetric one.

Finally, we study unification modulo the theories $ACUh$ and ACh. First, we study Paliath Narendran's proofs of the undecidability of the problem of unification modulo these equational theories ([9]). We expand all details in Narendran's proof on the undecidability for the case of $ACUh$, and also develop the mentioned adaptation of the proof for

the case of the undecidability of unification modulo ACh. The development of such complete proof contributes to the understanding of the complexity of the unification problem modulo these theories. Second, we studied a bounded unification algorithm modulo ACh developed by Ajay Kumar Eeralla and Christopher Lynch ([11]) that addresses the problem of unification on this equational theory providing solutions that are *bounded* meaning that the solutions cannot allow chained compositions of the homomorphic operator.

6.2 Future Work

The theory of equational reasoning (matching, unification, disunification, assymetric unification) modulo the studied and other equational theories is complex and deserve special attention. For future work, it would be interesting to present detailed proofs on correctness and completeness of the algorithm proposed by Eeralla and Lynch in [11]. Also, it would be of interest understanding how this approach may be adapted for the case of unification modulo the theory $ACUh$.

Another possible future work is to study the adaptation of the already existing algorithms for the other equational problems such as matching, unification, asymmetric unification and disunification problems.

References

[1] Liu, Zhiqiang: *Dealing Efficiently with Exclusive OR, Abelian Groups and Homomorphism in Cryptographic Protocol Analysis.* Ph.D. Thesis, Clarkson University, 2012. https://people.clarkson.edu/~clynch/papers/Dissertation_of_Zhiqiang_Liu.pdf. 1, 3, 34, 36, 37, 41, 42, 45, 48, 86

[2] Katz, Jonathan and Yehuda Lindell: *Introduction to Modern Cryptography.* Chapman and Hall/CRC Press, 2007. https://doi.org/10.1201/b17668. 1

[3] Dolev, Danny and Andrew Chi Chih Yao: *On the security of public key protocols.* IEEE Transactions on Information Theory, 29(2):198–208, 1983. https://doi.org/10.1109/tit.1983.1056650. 1

[4] Lowe, Gavin: *Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR.* In *Proc. Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems TACAS*, volume 1055 of *LNCS*, pages 147–166. Springer, 1996. https://doi.org/10.1007/3-540-61042-1_43. 1

[5] Ayala-Rincón, Mauricio, Maribel Fernández, and Daniele Nantes-Sobrinho: *Intruder deduction problem for locally stable theories with normal forms and inverses.* Theor. Comput. Sci., 672:64–100, 2017. https://doi.org/10.1016/j.tcs.2017.01.027. 2

[6] Escobar, Santiago, Catherine A. Meadows, José Meseguer, and Sonia Santiago: *Symbolic Protocol Analysis with Disequality Constraints Modulo Equational Theories.* In *Programming Languages with Applications to Biology and Security - Essays Dedicated to Pierpaolo Degano on the Occasion of His 65th Birthday*, volume 9465 of *LNCS*, pages 238–261. Springer, 2015. https://doi.org/10.1007/978-3-319-25527-9_16. 2

[7] Veena Ravishankar, Paliath Narendran and Kimberly A. Gero: *Asymmetric Unification and Disunification.* CoRR, abs/1706.05066, 2017. http://arxiv.org/abs/1706.05066, Published in 2019 in LNCS, vol 11560 https://doi.org/10.1007/978-3-030-22102-7_23. 2, 3, 10, 17, 18, 19, 20, 21, 24, 25, 26, 28, 29, 33, 86

[8] Liu, Zhiqiang and Christopher Lynch: *Efficient General Unification for XOR with Homomorphism.* In *Proc. 23rd International Conference on Automated Deduction CADE*, volume 6803 of *LNCS*, pages 407–421. Springer, 2011. https://doi.org/10.1007/978-3-642-22438-6_31. 3

[9] Narendran, Paliath: *Solving Linear Equations over Polynomial Semirings*. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science LICS*, pages 466–472. IEEE Computer Society, 1996. https://doi.org/10.1109/LICS.1996.561463. 3, 4, 50, 51, 52, 54, 55, 57, 58, 59, 60, 61, 62, 64, 86

[10] Nutt, Werner: *Unification in Monoidal Theories*. In *Proc. 10th International Conference on Automated Deduction CADE*, volume 449 of *LNCS*, pages 618–632. Springer, 1990. https://doi.org/10.1007/3-540-52885-7_118. 3, 4, 59

[11] Eeralla, Ajay Kumar and Christopher Lynch: *Bounded ACh Unification*. CoRR, abs/1811.05602, 2018. http://arxiv.org/abs/1811.05602, Published in *Math. Struct. in Comp. Science* Volume 30(6):664-682 (https://doi.org/10.1017/S0960129520000183), 2020. 3, 4, 51, 64, 65, 66, 67, 69, 70, 71, 72, 73, 78, 79, 80, 85, 87

[12] Baader, Franz and Wayne Snyder: *Unification Theory*. In *Handbook of Automated Reasoning*, volume 1, chapter Eighth, pages 445–532. MIT Press, North Holland, Elsevier Science Publisher, 2001. https://en.wikipedia.org/wiki/Handbook_of_Automated_Reasoning. 5, 8

[13] Baader, Franz: *The Theory of Idempotent Semigroups is of Unification Type Zero*. J. Autom. Reason., 2(3):283–286, 1986. https://doi.org/10.1007/BF02328451. 8

[14] Schmidt-Schauß, Manfred: *Unification under Associativity and Idempotence is of Type Nullary*. J. Autom. Reason., 2(3):277–281, 1986. https://doi.org/10.1007/BF02328450. 8

[15] Baader, Franz and Tobias Nipkow: *Term rewriting and all that*. Cambridge University Press, 1998. https://doi.org/10.1017/CBO9781139172752. 8, 9

[16] Comon, Hubert and Pierre Lescanne: *Equational Problems and Disunification*. J. Symb. Comput., 7(3/4):371–425, 1989. https://doi.org/10.1016/S0747-7171(89)80017-3. 10, 11

[17] Manber, Udi: *Introduction to algorithms - a creative approach*. Addison-Wesley, 1989. 11, 12

[18] Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein: *Introduction to Algorithms*. MIT Press, 3rd edition, 2009. https://mitpress.mit.edu/books/introduction-algorithms-third-edition. 12

[19] Wikipedia contributors: *NP-hardness*. https://en.wikipedia.org/w/index.php?title=NP-hardness&oldid=957633158, 2020. Wikipedia, The Free Encyclopedia. Accessed 29-July-2020. 12

[20] Baader, Franz and Klaus U. Schulz: *Combination Techniques and Decision Problems for Disunification*. Theor. Comput. Sci., 142(2):229–255, 1995. https://doi.org/10.1016/0304-3975(94)00277-0. 21

[21] Guo, Qing, Paliath Narendran, and David A.Wolfram: *Complexity of Nilpotent Unification and Matching Problems*. Inf. Comput., 162(1-2):3–23, 2000. https://doi.org/10.1006/inco.1999.2849. 22

[22] Vardi, Moshe Y. and Thomas Wilke: *Automata from logics to algorithms*. In *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, pages 629–736. Amsterdam University Press, 2008. 24

[23] Erbatur, Serdar, Santiago Escobar, Deepak Kapur, Zhiqiang Liu, Christopher Lynch, Catherine A. Meadows, José Meseguer, Paliath Narendran, Sonia Santiago, and Ralf Sasse: *Asymmetric Unification: A New Unification Paradigm for Cryptographic Protocol Analysis*. In *Proc. 24th International Conference on Automated Deduction CADE*, volume 7898 of *LNCS*, pages 231–248. Springer, 2013. https://doi.org/10.1007/978-3-642-38574-2_16. 34, 35, 37, 38, 39, 40, 42, 43, 48, 86