Universidade de Brasília

Syntactic, Commutative and Associative Anti-Unification

Gabriela de Souza Ferreira

Advisor: Dr. Daniele Nantes Sobrinho

Departamento de Matemática
Universidade de Brasília

Dissertation submitted in partial fulfillment of the requirements for the degree of
Master in Mathematics

Brasília, October 11, 2022

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de
Matemática

Syntactic, Commutative and Associative Anti-Unification

Gabriela de Souza Ferreira*

Dissertação apresentada ao Departamento de Matemática da Universidade de Brasília, como parte dos requisitos para obtenção do grau de

MESTRA EM MATEMÁTICA

Brasília, 21 de outubro de 2022.

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

```
          de Souza Ferreira, Gabriela
dG118s       Syntactic, Commutative and Associative Anti-Unification
          / Gabriela de Souza Ferreira; orientador Daniele Nantes
          Sobrinho. -- Brasília, 2022.
             128 p.


             Dissertação(Mestrado em Matemática) -- Universidade de
          Brasília, 2022.


             1. Anti-Unification. 2. Equational Theory. 3.
          Associativity. 4. Commutativity. I. Nantes Sobrinho,
          Daniele, orient. II. Título.
```

I dedicate this work to my parents, Gleide and Jorge. I love you.

Acknowledgements

First, I would like to thank God. Because it was the Lord who sustained me in the most difficult days of this journey, who gave me courage and willpower, God made it possible for me to enter the program and complete it successfully. To Him be all glory and honor.

I would also like to thank my family, who supported me in every way possible. To my parents, Gleide and Jorge, for investing in my education and always encouraging me to pursue my dreams.To my brothers, Israel and Luana, for cooking for me and taking care of the household chores, thanks to you I was able to dedicate myself as much as possible to my studies. To my sister, Raquel, for always advising me and being my personal cheerleader. To my nephew Guilherm, who played with me and called me ever as possible, thank you for warm my heart. My grandmother, uncles and cousins who took such good care of me, always cheering me up and telling me good news.

I would like to thank my friends who shared an apartment with me, Caio and Ismael, I couldn't have made a better choice. Thank you also for studying with me during the first months of the master's degree, I am eternally grateful to you. In particular, I would like to thank my partner Gabriel, thank you for always supporting me, believing in me and helping me at all times.

I thank all the friends I made while studying at UnB. Thank you Ali, Andrés, Daniella, Edna, Guilherme, Katianny and Mirelly for studying with me, discussing, clarifying doubts and expanding my view of the contents. I would also like to thank all the other friends who didn't study with me but contributed to my emotional and physical well being by eating in the RU, chatting in the university or playing volleyball.

I take this opportunity to apologize for not always being the best correspondent and, for that, I would like to thank everyone who kept in touch with me, whether through messages, memes or songs. In particular, Bruna, Luan and Vanessa. You made my life lighter.

I am grateful to the Theory of Computation seminar group. It was a pleasure to participate in the weekly meetings, thank you for all the tips and ideas proposed.

I would like to thank the professors who were on my bench, Maurício Ayala, Daniel Ventura and mu supervisor Daniele Nantes for their corrections and constructive criticisms that greatly improve my work. In particular, thank my supervisor Daniele Nantes for all the corrections and suggestions. Thank you for your infinite patience with me. Without you, this work would not have been possible, I am eternally grateful.

I would like to thank all the teachers who worked for my academic development. In particular, Djair, Ivan, Edcarlos, James, Mateus and Sérgio. You made me love study mathematics, open my eyes for the possibilities of its study and encouraged me to still becoming better. Without you, I don't know I would be here. Thanks for all.

I would also like to thank all the professionals who work so hard so that the UnB math department continues to function even during the pandemic.

At least, I would like to thank the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior — CAPES for the financial support offered for a part-time development of this dissertation. The financial security promoted by government institutions are essential for the advancement of Brazilian research.

Resumo

Esta dissertação apresenta um estudo detalhado do Problema de Anti-Unificação, investigado originalmente por Plotkin, Popplestone e Reynolds no início dos anos 70. Este problema consiste em encontrar um termo que mantém a maior estrutura comum entre dois outros termos dados. Isto é, dados s e t, o problema consiste em encontrar um terceiro termo r, que tem uma noção de *maximalidade*, tal que existam σ_1 e σ_2 tais que $r\sigma_1 = s$ e $r\sigma_2 = t$. Tal termo r é chamado de *generalizador menos geral de s e t*.

Neste trabalho investigaremos o Problema de Anti-Unificação Sintático, isto é, quando consideramos a igualdade sintática entre os termos; e também dos Problemas de Anti-Unificação módulo Comutatividade (C) e Associatividade (A), isto é, quando o problema de anti-unificação considera as igualdades módulo C e módulo A, respectivamente. Em todos os casos, apresentamos um algoritmo para resolução do problema além de suas propriedades de terminação, correção e completude. A partir das propriedades de cada algoritmo, apresentaremos então as propriedades dos conjuntos de soluções de cada problema.

Palavras Chave: Anti-Unificação, Teorias Equacionais, Associatividade, Comutatividade.

Abstract

This dissertation presents a detailed study of the Anti-Unification Problem, originally investigated by Plotkin, Popplestone and Reynolds in the early 70's. This problem consists of finding a term that maintains the greatest common structure between two other given terms. That is, given s and t, the problem is to find a third term r, with a notion of *maximality* such that there are substitutions σ_1 and σ_2 such that $r\sigma_1 = s$ and $r\sigma_2 = t$. Such a term r is called the *least general generalizer of s and t*.

In this work we will investigate the Syntactic Anti-Unification Problem, that is, when we consider the syntactic equality between the terms; and also the Anti-Unification Problems modulo Commutativity (C) and Associativity (A), that is, when the anti-unification problem considers the equalities modulo C and modulo A, respectively. In all cases, we present an algorithm for solving the problem in addition to its termination, soundness and completeness properties. From the properties of each algorithm, we will then present the properties of the sets of solutions for each problem.

Key Words: Anti-Unification, Equational Theory, Associativity, Commutativity.

Table of contents

Introduction

Motivation and History. The *Anti-Unification Problem* (also, known as the Generalization Problem), consists of finding an expression that generalizes two given expressions s and t. Informally, it means that we are looking for an expression that keeps the most common structure of s and t. For example, if we take $s = h(a,h(a,a))$ and $t = h(h(b,b),b)$, then the solution of the anti-unification problem for s and t is $h(x,y)$. Notice that they have a common function symbol h at root position and they differ in their arguments and $h(x,y)$ express it. The expression $h(x,y)$ is called a *generalizer* of s and t. When there is no other generalizer that maintains more structure of s and t than r, then r is the *least general generalizer* of s and t.

Formally, the Anti-Unification Problem (AUP) is defined as: Given a pair of expressions s and t, find the *least general generalizer* (*lgg*) of s and t, i.e., an expression r for which there exists a pair of substitutions (θ_1,θ_2) such that $r\theta_1 = s$ and $r\theta_2 = t$, that is the least with such property. The precise notion of "least" is based on an ordering of solutions which will be defined later on (Definition 1.8).

The notions of generalizer and least general generalizers were initially presented by Plotkin [10], Popplestone [11] and Reynolds [12] in three different, but related, works. All three were published the in Machine Intelligence Journal. Despite talking about the same topic, these works had different goals. On the one hand, Reynolds was motivated by the idea of find a solution for a dual problem of Unification (given two expressions s and t, find a substitution σ such that $s\sigma = t\sigma$), which had been studied by Robinson [13] where he developed a classical unification algorithm; such dual problem of Unification is the Anti-Unification one. On the other hand, Plotkin [10] and Popplestone [11] were motivated by a previous study using unification for the development of automatic deduction that made them suppose that there should exist anti-unifiers that could be used for the development of inductive methods.

Two decades later, Baader [3] proved the existence of a solution for the Syntactic Anti-Unification Problem and also for its extensions, the Equational (E) Anti-Unification Problem (AUP$_E$) for the equational theories of associativity (A) and commutativity (C), namely, the

Associative Anti-Unification Problem (AUP_A) and Commutative Anti-Unification Problem (AUP_C). However, he did not established a way of finding such solutions.

Equational Anti-Unification. An Equational Anti-Unification Problem (AUP_E) consists of solving an anti-unification problem taking into account a fixed set of identities E (the equational theory): Given expressions s and t, find the least expression r for which there is a pair of substitutions σ_1 and σ_2 such that $r\sigma_1 \equiv_E s$ and $r\sigma_2 \equiv_E t$. This expression r is called a *E-least general generalizer* of s and t, and if r' is a generalizer of s and t less specific than r, it is called only *E-generalizer*. The AUP is the special case in which $E = \emptyset$.

Notice that when an equational theory is being considered, the analysis of the E-generalizers of s and t does not depend only in the fixed structures of the expressions s and t, it is necessary to consider all the expressions s' and t' such that $s \equiv_E s'$ and $t \equiv_E t'$ to ensure that the set of solutions given will be complete.

For instance, when we consider a theory A for associativity $\{h(h(x,y),z) \approx h(x,h(y,z))\}$, where h is an associative function symbol, the problem becomes Associative Anti-Unification Problem, that is: Given expressions s and t, find an expression r such that there exists a pair of of substitutions σ_1 and σ_2 such that $r\sigma_1 \equiv_A s$ and $r\sigma_2 \equiv_A t$. In this case, taking expressions $s = h(a,h(a,a))$ and $t = h(h(b,b),b)$, it follows that s and t anti-unify with $h(x,h(x,x))$, representing the common function symbols h of the structures of s and t as well as the equality modulo associativity of its arguments.

Depending on the set E, the size of the set of a complete set of solutions for the Equational Anti-Unification Problem varies: it could be finite (type *finitary*), or infinite (type *infinitary*), or empty type (*nullary*). The biggest issue to solve this kind of problem is to treat the anti-unification with the properties of each theory induced by its axioms in a proper way. Then, it is not possible to make a unique algorithm that is able to solve the Equational Anti-Unification Problems for a general equational theory. In conclusion, each problem has to be analyzed to obtain a specific algorithm to solve it.

In the last decade, Alpuente et. al. [1] proposed rule based anti-unification algorithms to solve the Anti-Unification Problem for syntactic theory, and for some equational theories including A and C.

Objective. We will present a detailed material about the Anti-Unification Problem considering the empty, commutative and associative theories. We will follow the method proposed by Alpuente et. al. in [1], exploring the rule based algorithms described for each theory.

Contributions. In the following we present a list summarizing the contributions of this dissertation:

1. Detailed proofs of the main theorems related to the rule-based algorithm, $\mathtt{AUnif}_E$, for solving AUP_E for $E = \emptyset$, C and A. More precisely, theorems related to termination, soundness and correctness of $\mathtt{AUnif}_\emptyset$, $\mathtt{AUnif}_C$ and $\mathtt{AUnif}_A$.

2. We presented some new examples to illustrate the concepts and facilitate the reader's understanding of these topics.

3. We proposed a new definition of associative pair of positions (Definition 4.5), fixing an imprecision in the original definition given by [1]. Furthermore, we presented a counter example of the Lemma 19 of [1] in Remark 4.3 that was used to prove completeness of $\mathtt{AUnif}_A$ in [1]. We replaced this lemma by three new ones (Lemmas 4.2, 4.3 and 4.4) in order to prove that the A-anti-unification algorithm is still complete.

4. We developed a rich appendix which contains detailed proofs of the lemmas used to prove the main theorem of each theory.

Related work. In 2020 Cerna and Kutsia [7] presented an counter example of the completeness of a modular order-sorted algorithm for the theory ACU (Unit) proposed by Alpuente at al in [1], one year later Alpuente et al. proposed a fix in [2]. However, this theory and associated algorithm is out of the scope of this dissertation and the problems presented here in the definition of associative pair of positions and Lemma 19 in [1] have not been fixed in [2].

Anti-unification algorithms are widely used in software engineering, involving machine learning, logic programming [9] and the development of inductive software for artificial intelligence [14]. Also, Anti-unification is used to identify and avoid clone in software [14, 6] and software misplacing [8]. In summary, these algorithms serves to reduce the time of development and maintenance of software, prevent bugs, degradation or even disruptions of codes. However, these applications are out of the scope of this dissertation.

Organization.

Chapter 1: In this chapter we will remember some basics notions that will be useful for the development of all the work, such as the definitions of Σ-term and how to represent their tree; substitutions and instances; equational theories and lexicographic and multiset orderings. Furthermore, we establish the main notion and definitions necessary to study the Anti-Unification Problem (Definition 1.16) for the syntactic theory, that is, the definitions of generalizer (Definition 1.14), least general generalizer (Definition 1.20) and the definition of Equational Anti-Unification Problem (Definition 1.23).

Chapter 2: We will study the Syntactic Anti-Unification Problem (AUP). In Section 2.1 we present a rule-based algorithm, `AUnif`, which outputs a least general generalizer for a given AUP. We also present examples to illustrate how to apply this algorithm and state some definitions necessary to study its properties such as termination (Theorem 2.1), confluence (Theorem 2.2) and correctness (Theorem 2.3). In Section 2.2 we study the properties of $\mathtt{AUnif}_{\emptyset}$, presenting detailed proofs of them and discussing the relevance of these properties to the solution of the problem.

Chapter 3: We will study the Commutative Anti-Unification Problem (AUP_C). In Section 3.1 we present some motivational examples that differs this problem from the syntactic one. In Section 3.2 we present a rule-based algorithm, $\mathtt{AUnif}_C$, to solve AUP_C. We establish some notion that are necessary to study its main properties such as termination (Theorem 3.1), soundness (Theorem 3.2) and correctness (Theorem 3.3). In Section 3.3 we focus on present detailed proofs of the properties of $\mathtt{AUnif}_C$.

Chapter 4: We will study the Associative Anti-Unification Problem (AUP_A). In Section 4.1 we present some examples of this problem, talking about the need of an specific algorithm to solve it. In Section 4.2 we present the rule-based algorithm, $\mathtt{AUnif}_A$, that solve this problem, showing how it works by the use of examples, also, we present some notions necessaries to study the main properties of $\mathtt{AUnif}_A$ such as termination (Theorem 4.1), soundness (Theorem 4.2) and completeness (Theorem 4.2). Finally, in the Section 4.3 we present detailed proofs of these properties.

Chapter 5 We conclude the dissertation with the principal consideration of the development of this work and we also propose some possibilities for future work.

Chapter 1

Background

In this chapter we present some basic notations and definitions that are necessary to study the Anti-Unification Problem. In Section 1.1 we introduce standard notions such as Σ-terms and their structures; orderings and finally we define what is an Anti-Unification Problem. In Section 1.2 we study equational theories, paying a special attention to commutative and associative theories; then we define the Equational Anti-Unification Problem. In the Section 1.3 we introduce the Abstract Reduction System and some important results involving this notion that are useful for understanding the rule-based algorithms defined in the next chapters.

1.1 Syntax

1.1.1 Σ-terms and their structures

The notions and definitions stated in this section are consistent with [4].

Fix a countable set $\mathcal{X} = \{w, x, y, z, \dots\}$ of variables and a finite signature

$$\Sigma = \{f_1 : m_1, \dots, f_k : m_k\}$$

of function symbols of arities $m_1, \dots, m_k$, where each m_i is a non-negative integer. The **set** $T(\mathcal{X},\Sigma)$ **of all Σ-terms** is defined inductively as follows:

- $\mathcal{X} \subset T(\mathcal{X},\Sigma)$ (every variable is term);
- for all $n \in \mathbb{N}$, $f : n \in \Sigma$ and all $t_1, \dots t_n \in T(\mathcal{X},\Sigma)$, we have $f(t_1, \dots, t_n) \in T(\mathcal{X},\Sigma)$ (applications of function symbols to terms yields terms).

We will call "Σ-terms" only "terms". When a term is composed only of a function symbol of arity 0, this term is called **constant function symbol**. We will use a, b, c, d to denote this kind of terms.

We will use the notation $\overline{t_n}$ to represent a list $t_1, \ldots, t_n$ of terms. Then, the term $f(t_1, \ldots, t_n)$ can be represent by $f(\overline{t_n})$.

Definition 1.1 (Set of variables of a term)**.** Given a term $t \in T(\mathcal{X}, \Sigma)$. The set of all variables that occur in t is denoted by $\mathcal{V}(t)$.

Definition 1.2 (Position of a term)**.** The set of positions of a term $t \in T(\mathcal{X}, \Sigma)$, denoted by $\mathtt{pos}(t)$, is a set of strings over positive integers. This set is defined inductively as follows:

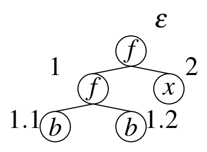
- if $t = x \in \mathcal{X}$, then $\mathtt{pos}(t) = \varepsilon$, where ε denotes the empty string;
- if $t = f(t_1, \ldots, t_n)$, then $\mathtt{pos}(t) = \{\varepsilon\} \cup \bigcup_{i=1}^{n} \{i.p \mid p \in pos(t_i)\}$

The position ε is called **root position** of t and the function symbol that appear in this position is denoted by $\mathtt{root}(t)$. Given $p, q \in \mathtt{pos}(t)$, we write $p < q$ if p is above (i.e, p is a prefix position of q, c.f. Definition 1.6) q; $p = q$ if p and q are the same position; and $p || q$ if both positions are incomparable.

Definition 1.3 (Subterms)**.** For $p \in \mathtt{pos}(t)$, the subterm s of t at position p is denoted by $t|_p = s$, and is defined by induction on the length of p as follows:

- $t|_\varepsilon = t$,
- if $t = f(t_1, \ldots, t_n)$, then $f(t_1, \ldots, t_n)|_{i.q} = t_i|_q$.

Example 1.1. For a signature $\Sigma = \{f : 2, b : 0\}$, the term $t = f(f(b,b),x) \in T(\mathcal{X}, \Sigma)$, can be represented as the following tree: The nodes represents the symbols in Σ and arrows connected function symbols to their arguments. We annotate the positions of t above each node.



Then, $\mathtt{pos}(t) = \{\varepsilon, 1, 1.1, 1.2, 2\}$, $t|_\varepsilon = f(f(b,b)x)$ and $t|_{1.1} = f(b,b)|_1 = b$.

Definition 1.4. . For $p \in \mathtt{pos}(s)$, we denote by $s[t]_p$ the term that is obtained from s by **replacing the subterm at position** p **by** t, i.e,

- $s[t]_\varepsilon = t$,
- $f(s_1,\dots,s_n)[t]_{i.q} = f(s_1,\dots,s_i[t]_q,\dots,s_n)$.

Definition 1.5 (Term size)**.** The **size of term** is given inductively as follows:

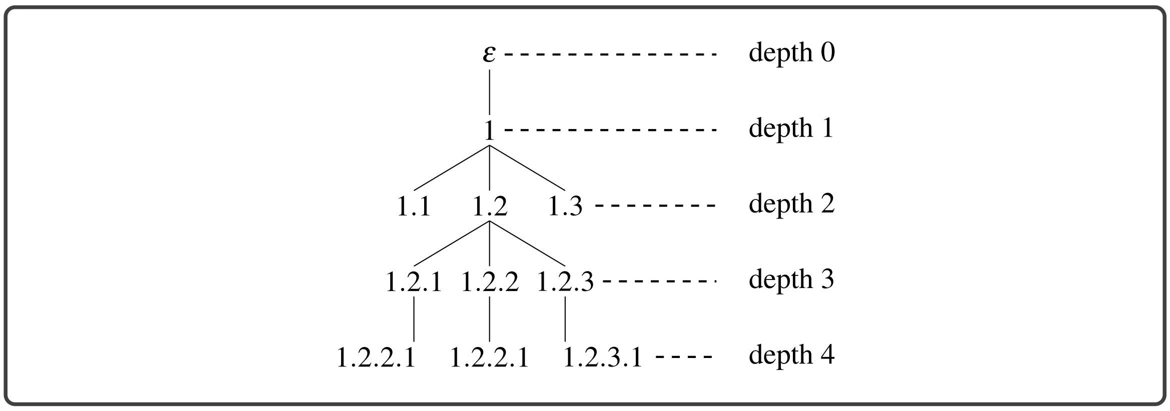
$$|x| = 1, \quad |a| = 1 \text{ and } |f(s_1,\dots,s_n)| = \sum_1^n |s_i| + 1.$$

Definition 1.6 (Depth, Prefix and Index of a position)**.**

- Given a position $p \in \texttt{pos}(s)$ the **depth of** p is the length of the position given inductively as follows:
  - $\texttt{depth}(\varepsilon) = 0$,
  - $\texttt{depth}(i.p) = 1 + \texttt{depth}(p)$
- Given a position $p \in \texttt{pos}(s)$ with $\texttt{depth}(p) = n$, $(p)^k$ is the **prefix** of the position p at depth $k \leq n$, i.e:
  - $(p)^0 = \varepsilon$
  - $(i.p)^k = i.(p)^{k-1}$, if $k > 0$.
- Given a position $p \in \texttt{pos}(s)$ with $\texttt{depth}(p) = n$, $(p)_k$ is the **index** of p at depth $k \leq n$, i.e:
  - $(i.p)_1 = i$
  - $(i.p)_{k+1} = (p)_k$

Example 1.2. Consider the positions $p = 1.2.1.1$ and $q = 1.2.3.1$ of one term s. It follows that:

- $\texttt{depth}(p) = \texttt{depth}(q) = 4$,
- $(p)^2 = (1.2.1.1)^2 = 1.(2.1.1)^1 = 1.2$,
- $(q)^2 = (1.2.3.1)^2 = 1.(2.3.1)^1 = 1.2$,
- $(p)_3 = (1.2.1.1)_{2+1} = (2.1.1)_{1+1} = (1.1)_1 = 1$,
- $(q)_3 = (1.2.3.1)_{2+1} = (2.3.1)_{1+1} = (3.1)_1 = 3$.



Lemma 1.1. Given $p = p_1 \dots p_n$, with $p_1, \dots, p_n \in \mathbb{N}$, a position such that $\texttt{depth}(p) = n > 0$. Then $(p)^n = p$ and $(p)_n = p_n$.

Proof. The proof proceeds by induction over the depth n of the position p.

Base Case: If $n = 1$, then $p = p_1$. By Definition 1.6 it follows that $(p)^1 = (p_1)^1 = p_1$ and $(p)_1 = (p_1)_1 = 1$.

Inductive Step: If $n > 1$, then $p = p_1, \dots, p_n$ it follows that

$$(p)^n = (p_1 \dots p_n)^n = p_1.(p_2 \dots p_n)^{n-1} \overset{(I.H.)}{=} p_1.p_2 \dots p_n = p, \text{ and}$$

$$(p)_n = (p_1 \dots p_n)_n = (p_2 \dots p_n)_{n-1} \overset{(I.H.)}{=} p_n.$$

□

The next lemma relates the prefix k of a position p with the same prefix of a position that is immediately bellow p, i.e, the prefix k of a position $p.i$ with $i = \mathbb{N}$.

Lemma 1.2. Given $p \in \texttt{pos}(s)$. If $p' = p.i \in \texttt{pos}(s)$ for some $i \in \mathbb{N}$ then $(p)^k = (p')^k$ for each $0 \leq k \leq \texttt{depth}(p)$.

Proof. The proof proceeds by induction over the depth n of the position p.

Base case: If $n = 0$ then $p = \varepsilon$ and $p' = i$. It follows by definition of prefix of a position that $(p)^0 = \varepsilon = (p')^0$. Then, the result follows trivially.

Inductive step If $n > 0$. Notice that we can write $p = p_1 \dots p_n$ for $p_1, \dots, p_n \in \mathbb{N}$. Then $p' = p_1 \dots p_n.p_{n+1}$ with $i = p_{n+1}$ a natural number. It follows that

$$\begin{aligned}(p)^k = (p_1 \dots p_n)^k = p_1(p_2 \dots p_n)^{k-1} \\ \overset{(I.H.)}{=} p1.(p_2. \dots .p_n.p_{n+1})^{k-1} = (p1. \dots p_n.p_{n+1})^k = (p')^k.\end{aligned}$$

□

Notice that given $\texttt{depth}(p) = n$ the result above cannot be extend to $k = n+1$ because $(p)^{n+1}$ is not defined.

Example 1.3. Take the position $p = 1.2.3$ and $p' = 1.2.3.4$ it follows that

- $(p)^0 = \varepsilon = (p')^0$,
- $(p)^1 = (1.2.3)^1 = 1 = (1.2.3.4)^1 = (p')^1$,
- $(p)^2 = (1.2.3)^{1+1} = 1.(2.3)^1 = 1.2 = 1.(2.3.4)^1 = (1.2.3.4)^{1+1} = (p')^2$,
- $(p)^3 = (1.2.3)^{2+1} = 1.(2.3)^2 = 1.2.3 = 1.(2.3.4)^2 = (1.2.3.4)^{2+1} = (p')^3$, and
- $(p')^4 = 1.(2.3.4)^{2+1} = 1.2(3.4)^{1+1} = 1.2.3.(4)^1 = 1.2.3.4 = p'$ and $(p)^4$ is not defined.

The next lemma relates the index k of a position p with the same index of a position immediately bellow p, i.e, the index k of the position $p.i$ for some $i \in \mathbb{N}$.

Lemma 1.3. Given $p \in \texttt{pos}(s)$. If $p' = p.1 \in \texttt{pos}(s)$ for some $i \in \mathbb{N}$ then $(p)^k = (p')^k$ for each $0 < k \leq \texttt{depth}(p)$.

Proof. The proof proceeds by induction over the depth n of p. When writing $p = p_1 \dots p_n$, with $p_1, \dots, p_n \in \mathbb{N}$, it follows that $p' = p_1 \dots p_n.p_{n+1}$ with $i = n+1$ a natural number.

Base Case: If $n = 1$, then $p = p_1$ and $p' = p_1.p_2$. It follows by Definition 1.6 that $(p)^1 = (p_1)^1 = 1 = (p_1.p_2)^1 = (p')^2$.

Inductive Step: If $n > 1$, then $p = p_1 \dots p_n$ and $p' = p_1 \dots p_n p_{n+1}$. For $0 < k \leq \texttt{depth}(p)$ it follows that

$$(p)_k = (p_1 \dots p_n)_k = (p_2 \dots p_n)_{k-1}$$
$$\overset{(I.H.)}{=} (p_2 \dots p_n.p_{n+1})_{k-1} = (p_1 \dots p_n.p_{n+1})_k = (p')_k.$$

□

Again, the result cannot be extended to $k > \texttt{depth}(p)$ because $(p)_k$ will be not defined.

Example 1.4. Let $p = 1.2.3$ and $p' = 1.2.3.4$ it follows that

- $(p)_1 = (1.2.3)_1 = 1 = (1.2.3.4)_1$,

- $(p)_2 = (1.2.3)_2 = (2.3)_1 = 2 = (2.3.4)_1 = (1.2.3.4)_2 = (p')_2$,
- $(p)_3 = (2.3)_2 = (3)_1 = 3 = (3.4)_1 = (2.3.4)_2 = (1.2.3.4)_3 = (p')_3$, and
- $(p')_4 = 4$ while $(p)_4$ is not defined.

Definition 1.7 (Substitution)**.**

1. A **substitution** $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ is a finite mapping from variables to terms.
2. We write $\mathtt{dom}(\sigma) = \{x \in \mathcal{X}; x\sigma \neq x\}$ to denote the **domain** of a substitution σ and $\mathtt{vran}(\sigma) = \bigcup\{\mathcal{V}(x\sigma) \mid x\sigma \neq x\}$ to denote its **set of range variables**.
3. A substitution is called a **variable renaming** if its restriction to $\mathcal{X}$ is a permutation of $\mathcal{X}$.
4. The **identity substitution** has empty domain and this substitution is represented by id.
5. The **composition of substitutions** is written from left to right, thus $\sigma_1\sigma_2$ means apply σ_1 first and σ_2 after.
6. To denote a **n-upla of substitutions** we write $\overline{\sigma} = (\sigma_1, \ldots, \sigma_n)$ where

$$\lambda\overline{\sigma} = (\lambda\sigma_1, \ldots, \lambda\sigma_n) \text{ and } \overline{\sigma}\lambda = (\sigma_1\lambda, \ldots, \sigma_n\lambda).$$

The application of $\overline{\sigma}$ to a term t is defined as $t\overline{\sigma} = (t\sigma_1, \ldots, t\sigma_n)$.

Definition 1.8 (Instance)**.** Given terms s and r, s is an instance of r iff exists a substitution θ such that $r\theta = s$. The notation is $r \leq s$

In the illustrations we will use a vertical arrow "$\downarrow$" with r in the top and s below, to mean $r < s$. To denote the substitution σ such that $r\sigma = s$ it appears as a label in the arrow. As is illustrated below:

$$\begin{tikzcd} r \arrow[d, "\sigma"'] \\ s \end{tikzcd}$$

Example 1.5. Given terms $r = f(x,y)$, $s = f(a, f(b,b))$, and $t = f(a, f(a,b))$,

- taking the substitution $\theta_1 = \{x \mapsto a, y \mapsto f(b,b)\}$ it follows that $r\theta_1 = s$. Therefore, s is an instance of r;

- taking the substitution $\theta_2 = \{x \mapsto a, y \mapsto f(a,b)\}$ it follows that $r\theta_2 = t$. Therefore, t is an instance of r.

Remark 1.1. When $s < t$ it is also common to say "s is more general than t", or, equivalently, "t is less general than s".

1.1.2 Orders

In this section, we introduce some order relations that will be used to prove the termination of the rule-based algorithms defined in the next chapters.

Definition 1.9 (Lexicographic order)**.** Given two strict orders, i.e, two transitive, asymmetric and irreflexive orders, $(A, >_A)$ and $(B, >_B)$, the lexicographic product $>_{A\times B}$ on $(A \times B)$ is defined by

$$(x,y) >_{A\times B} (x',y') :\text{ iff } (x >_A x') \vee (x = x' \wedge y >_B y').$$

Example 1.6. Given the usual order $>$ over naturals, $(1,1) >_{\mathbb{N}\times\mathbb{N}} (0,2)$.

Theorem 1.1 (Theorem 2.4.2)**.** The lexicographic product of two terminating relation is again terminating.

Proof. To see the proof of this theorem, [4]. □

Definition 1.10 (Multiset)**.** A **multiset** M over a set A is a function $M : A \longrightarrow \mathbb{N}$. Intuitively, $M(x)$ is the number of copies of $x \in A \in M$.

Notation: We use the standard notation $\{\{a,a,b\}\}$ as an abbreviation of the function $\{a \mapsto 2, b \mapsto 1, c \mapsto 0\}$ over the set A.

Definition 1.11 (Finite Multiset)**.** A multiset M is finite if there are only finitely many x such that $M(x) > 0$. Let $\mathcal{M}(A)$ denote the set of all finite multisets over A.

In this paper we will use only finite multisets.

Definition 1.12 (Basic operations in $\mathcal{M}(A)$)**.**

Element: $x \in M$ iff $M(x) > 0$.

Inclusion: $M \subseteq N$ iff $\forall x \in A. M(x) \leq N(x)$.

Union: $(M \cup N)(x) := M(x) + N(x)$.

Difference: $(M - N)(x) := M(x) \dot{-} N(x)$, where $m \dot{-} n$ is $(m - n)$ if $m \leq n$ and is 0 otherwise.

Definition 1.13 (Multiset order)**.** Given one strict order $>$ on a set A, we define the corresponding **multiset order** $>_{mul}$ on $\mathcal{M}(A)$ as follows: $M >_{mul} N$ if, and only if, there exist $X, Y \in \mathcal{M}(A)$ such that

- $\emptyset \neq X \subseteq M$,
- $N = (M - X) \cup Y$, and
- $\forall y \in Y. \exists x \in X. x > y$.

Example 1.7. Given the standard order $>$ over naturals, then $\{\{0\}\} >_{mul} \{\{0,0,0\}\}$.

Theorem 1.2 (Theorem 2.5.5 in [4])**.** the multiset order $>_{mul}$ is terminating iff $>$ is terminating.

Proof. To see the proof of this theorem, consult [4]. □

1.1.3 Anti-Unification Problem

The Definition 1.8 gives us a way to compare two terms s and t, ordering then according to their images under substitutions. Thanks to instantiation ordering that is a partial order on terms, it is possible to organize terms in chains, starting with more generals to least general ones. The notions and notations in this section are consistent with [1].

Definition 1.14 (Generalizer)**.** Given two terms $s, t \in T(\mathcal{X}, \Sigma)$. A **generalizer** of s and t is a term $r \in T(\mathcal{X}, \Sigma)$ for which there exists a pair of substitutions $\overline{\theta} = (\theta_1, \theta_2)$ such that $r\theta_1 = s$ and $r\theta_2 = t$. In other words, r is a generalizer of s and t iff both s and t are instances of r. The set of generalizers of s and t is denoted by $\mathtt{gen}(s,t)$.

Example 1.8 (Continuation of Example 1.5)**.** Given terms $r = f(x,y), s = f(a, f(b,b))$ and $t = f(a, f(a,a))$. The Example 1.5 shows that $r\theta_1 = s$ and $r\theta_2 = t$, i.e, $r < s$ and $r < t$. Therefore, r is a generalizer of s and t.

Example 1.9. Taking terms $s = f(f(x,y), f(c,z)), t = f(f(a,b), f(c,z))$ and the substitution $\lambda = \{x \mapsto a, y \mapsto b\}$ it follows $s\lambda = t$, then $s < t$. Notice that, taking the identity substitution id it follows that the pair of substitutions (id, λ) satisfies $(id, \lambda)s = (s,t)$. Then, s is an generalizer of t and itself.

Example 1.10 (The trivial generalizer)**.** Every pair of terms s and t have a generalizer. In fact, given s and t and taking the pair of substitution $\overline{\theta} = (\theta_1, \theta_2) = (\{x \mapsto s\}, \{x \mapsto t\})$ for a new variable x, it follows that $x\overline{\theta} = (s,t)$. Therefore, x is a generalizer of s and t, as is nicely illustrated below:

$$\begin{tikzcd}
 & x \arrow[dl, "\theta_1"'] \arrow[dr, "\theta_2"] & \\
s & & t
\end{tikzcd}$$

In conclusion, the generalization problem always have the variable x as a trivial solution. Since there is no term more general than a variable, x is the most general generalizer of any terms s and t. Therefore, a more interesting problem is looking for the *least general generalization* of s and t (or the *most specific generalizer*). To formalize it we state the next definition:

Definition 1.15. Let Σ be a signature and terms s and $t \in T(\mathcal{X},\Sigma)$. We define the **a least general generalization** of s and t as the greatest lower bound generalizer of s and t (see Figure 1.1). In other words:

$$\mathtt{lgg}(s,t) = \{r \in \mathtt{gen}(s,t) \mid r' \leq r, \forall r' \in \mathtt{gen}(s,t)\}$$

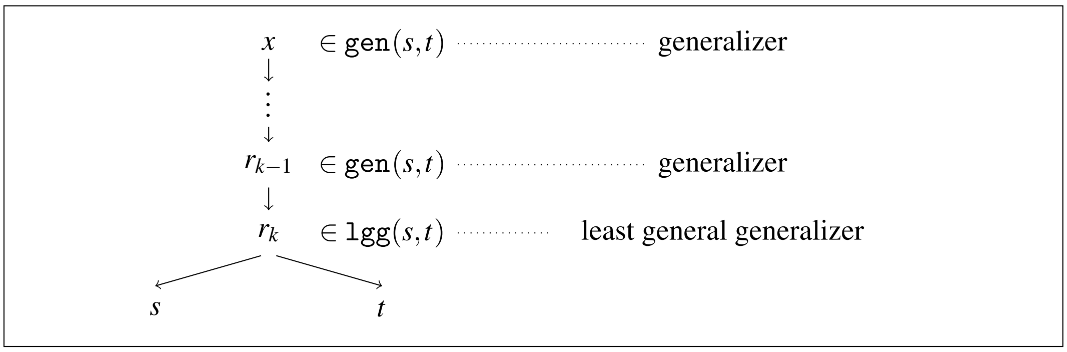


Fig. 1.1 Least general generalizer of s and t

Definition 1.16 (Anti-Unification Problem — AUP)**.** Given $s,t \in T(\mathcal{X},\Sigma)$, the Anti-Unification Problem (AUP) of s and t, denoted by $\mathcal{A}\langle s,t\rangle$, has as solution a term r such that $r \in \mathtt{lgg}(s,t)$.

Example 1.11 (Continuation of Example 1.8)**.** Given $s,t \in T(\mathcal{X},\Sigma)$, where $s = f(a, f(b,b))$ and $t = f(a, f(a,b))$. The Example 1.8 and the Example 1.5 yields that $x, f(x,y) \in \mathtt{gen}(s,t)$.

Besides, as taking the term $f(a, f(z,w))$ and the pair of substitutions $\overline{\theta} = (\theta_1, \theta_2)$ with $\theta_1 = \{z \mapsto b, w \mapsto b\}$ and $\theta_2 = \{z \mapsto b, w \mapsto a\}$, one has $f(a, f(z,w))\theta_1 = s$ and $f(a, f(z,w))\theta_2 = t$, i.e, $f(a, f(z,w)) \in \mathtt{gen}(s,t)$.

Now, notice that $f(x,y) < f(a,f(z,w))$. In other words, $f(a,f(z,w))$ is less general than $f(x,y)$. In fact it is a least general generalizer of s and t, i.e. $f(a,f(z,w)) \in \mathtt{lgg}(s,t)$.

There is a chain of generalizers of s and t, organized by the instantiation order. The chain begins with the trivial generalizer x and goes to the other generalizers, until it reaches in the least general generalization, as bellow:

$$\begin{cases} x < f(x,y) < f(a,f(z,w)) < s \\ x < f(x,y) < f(a,f(s,w)) < t. \end{cases}$$

By observing the term structure of s and t and comparing it with the term structure of $f(x,y)$ and $f(a,f(z,w))$, it is easy to see the least general generalizer $f(a,f(z,w))$ has more in common with s and t than the generalizer $f(x,y)$. More precisely, $f(a,f(z,w))$ preserves the common function symbol f in the root of s and t and also the common function symbols a and f in the arguments of s and t, while $f(x,y)$ only preserves the root symbol f.

1.2 Equational theories

An equational theory is a pair (Σ, E), where Σ is a signature and E a set of equations between Σ-terms, denoted $s \approx t$, that are axioms of the theory. Then, the syntactic theory is just a equational theory with no axioms, i.e, $E = \emptyset$. The equational theories with the commutative and the associative properties are, respectively, defined by the following axioms:

$$C = \{g(x,y) \approx g(y,x)\}$$

$$A = \{h(h(x,y),z) \approx h(x,h(y,z))\}.$$

Let E be an equational theory, we denote by Σ_E the signature for which the axioms of E holds for every function symbol $f \in \Sigma_E$.

We say a function symbol f is commutative (respectively, associative) to mean that the axiom of C (respectively A) hold for this function symbol. Notation:

- $\Sigma_\emptyset$ denote the signature with only free function symbols,
- Σ_A denote the signature with only associative function symbols,
- Σ_C denote the signature with only commutative function symbols.
- $\Sigma_{\emptyset\cup C} = \Sigma_\emptyset \cup \Sigma_C$. Similarly, for $\Sigma_{\emptyset\cup A}$.

From now on we will omit Σ to denote an equational theory (Σ, E), writing only E.

For an equational theory E, the **equality induced by the axioms of** E is denoted as $=_E$. For example, given $g \in \Sigma_C$, $f : 2, a : 0, b : 0 \in \Sigma_\emptyset$ and $g(f(a,b), f(a,x)) \in T(\mathcal{X}, \Sigma_{\emptyset \cup C})$, then $g(f(a,b), f(a,x)) =_C g(f(a,x), f(a,b))$. Notice that since f is not an function commutative function symbol then $g(f(a,b), f(a,x)) \neq_C g(f(b,a), f(x,a))$.

Definition 1.17 (E-instantiation)**.** Given a theory E, s is an E**-instance** of r iff there exists a substitution θ such that $r\theta =_E s$, denoted as $r \leq_E s$. When θ is not the identity substitution we write $r <_E s$. When the theory E is empty, we omit the parameter E and write only $r \leq s$.

Example 1.12 (C-instances)**.** Let $g \in \Sigma_C$, $s = g(a,b) \in \Sigma_{\emptyset \cup C}$. Then,

- with $\theta_1 = \{x \mapsto s\}$, it follows that $x\theta_1 = s$, therefore x is a C-instance of s,
- with $\theta_2 = \{x_1 \mapsto a, x_2 \mapsto b\}$, it follows that $g(x_1,x_2)\theta_2 = s$, therefore s is a C-instance of $g(x_1,x_2)$,
- with $\theta_3 = \{x_2 \mapsto b\}$, it follows that $g(a,x_2)\theta_3 = s$, then s is a C-instance of $g(a,x_2)$,
- with $\theta_4 = \{x_1 \mapsto b\}$, it follows that $g(x_1,a)\theta_4 = g(b,a) =_C g(b,a) = s$, therefore s is a C-instance of $g(x_1,a)$.

Hence, s is an instance modulo C of the terms $x, g(x_1,x_2), g(a,x_2),$ and $g(x_1,a)$.

Example 1.13 (A-instances)**.** Given $h \in \Sigma_A$, and Taking terms $s = h(f(a,a), h(a,b))$, $t = h(h(f(b,b),a)b)$ and $r = h(f(x,x), h(a,b)) \in T(\mathcal{X}, \Sigma_{\emptyset \cup A})$,

- with $\theta_1 =_A \{x \mapsto a\}$, it follows that $r\theta_1 =_A h(h(f(a,a),a),b) = s$, therefore s is an A-instance of r;
- with $\theta_2 =_A \{x \mapsto b\}$, it follows that $r\theta_2 =_A h(h(f(b,b),a),b) =_A h(f(b,b),h(a,b)) =_A t$, therefore t is an A-instance of r.

Definition 1.18 (Equivalency induced by instantiation)**. The equivalence relation** $\equiv_E$ is defined as $s \equiv_E t$ iff $s \leq_E t$ and $t \leq_E s$. When the theory is empty, we omit the suffix E.

Remark 1.2. When $E = \emptyset$, $s \equiv t$ means that s is equal to t, except for variable renaming.

Example 1.14. If $x, y \in \mathcal{X}$, i.e, x and y are variables, it is easy to see $x\{x \mapsto y\} = y$ and $y\{y \mapsto x\} = x$. Then, $x < y$ and $y < x$ yields that $x \equiv y$.

Example 1.15.

- Let $g \in \Sigma_C$, it follows that $g(a,x) \equiv_C g(y,a)$. Because for $\theta_1 = \{x \mapsto y\}$ it follows that $g(a,x)\theta_1 = g(a,y) =_C g(y,a)$ and for $\theta_2 = \{y \mapsto x\}$, $g(y,a)\theta_2 = g(x,a) =_C g(a,x)$.
- Let $h \in \Sigma_A$, it follows that $h(h(a,b),h(c,c)) \equiv_A h(h(h(a,b),c),c)$.

In general the relations $=_E$ and $\equiv_E$ does not coincide. Actually, $=_E \subseteq \equiv_E$. Notice that if $s =_E t$, then $s\,id =_E t$ and $t\,id = s$, which implies that $s \leq_E t$ and $t \leq_E s$, therefore $s \equiv_E t$. For instance, $g(a,x) =_C g(x,a) \neq_C g(y,a)$ but Example 1.15 yields that $g(a,x) \equiv_C g(y,a)$. In this case these terms differ modulo $=_C$ by a difference of variables.

1.2.1 Equational Anti-Unification

In this section we will to extend the definition of the Anti-Unification Problem taken into a certain equational theory E. Definitions and notations are mostly consistent with [1].

Definition 1.19. A **minimal and complet set of generalizations** of two terms s and t is the set $\mathcal{G}$ with the following three properties:

1. Each element of $\mathcal{G}$ is an generalization of s and t (soundness of $\mathcal{G}$).
2. For each generalization r' of s and t, there exists $r \in \mathcal{G}$ such that $r' \leq_E r$, i.e, r is less general than r' modulo E (completeness of $\mathcal{G}$).
3. No two distinct elements of $\mathcal{G}$ are $\leq_E$-comparable: If $r_1, r_2 \in \mathcal{G}$ such that $r_1 \leq_E r_2$, then $r_1 \equiv_E r_2$ (minimality of $\mathcal{G}$).

Definition 1.20 (Complete set of generalizers)**.** Let $s,t \in T(\mathcal{X},\Sigma_E)$, a complete set of generalizers modulo E, denoted by $\texttt{gen}_E(s,t)$, is defined as follows:

$$\texttt{gen}_E(s,t) = \{r \mid \exists u, u' \text{ such that } s =_E u, t =_E u', r \in \texttt{lgg}(u,u')\}.$$

Definition 1.21. Given $s,t \in T(\mathcal{X},\Sigma_E)$, then all possible set of least general generalizers of s and t is defined as follows

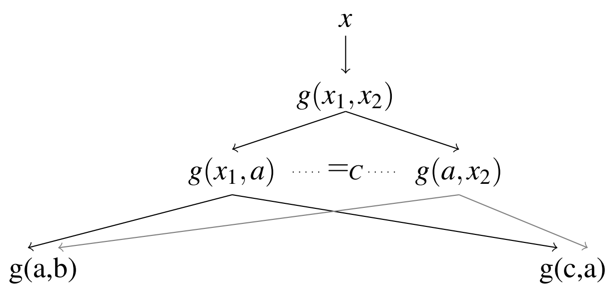
$$\texttt{CGM}_E(s,t) = \left\{ G \subseteq \texttt{gen}_E(s,t) \;\middle|\; \begin{array}{l} G \text{ is a complete } E\text{- generalization of } \texttt{gen}_E(s,t) \\ \text{and } E\text{-minimal} \end{array} \right\}$$

Definition 1.22. Given $s,t \in T(\mathcal{X},\Sigma_E)$ a set $\texttt{lgg}_E(s,t)$ of least general generalizers of s and t modulo E is defined by choosing $G \in \texttt{CGM}_E(s,t)$.

Definition 1.23 (Equational Anti-Unification Problem — AUP$_E$)**.** Given $s,t \in T(\mathcal{X},\Sigma_E)$, the Equational Anti-Unification Problem of s and t, denoted by $\mathcal{A}_E\langle s,t\rangle$, has as a solution a minimal and complete set of least general generalizers modulo E of s and t, i.e, $\mathtt{lgg}_E(s,t)$. As usual, when the theory E is empty, we omit the suffix E.

Remark 1.3. The Syntactic Anti-Unification Problem presented in Section 1.1.3 is not a different problem than what was presented here in Definition 1.16. Actually, the Syntactic Anti-Unification Problem is just a special case of Equational Anti-Unification obtained as take $E = \emptyset$.

Example 1.16 (Set of C-least general generalizer)**.** Let $g \in \Sigma_C$, $s = g(a,b)$ and $t = g(c,a)$. The following terms instantiates both s and t: x, $g(x_1,x_2)$, $g(x_1,a)$, $g(a,x_2)$, as is illustrated bellow. It follows that they are in the set of C-generalizers of s and t. Notice that $g(x_1,a)$ and $g(a,x_2)$ are maximal terms of this set, i.e, they are a complete C- generalization of $\mathtt{gen}_C(s,t)$. However, since $g(x_1,a)\{x_1 \mapsto x_2\} =_C g(a,x_2)$ and $g(a,x_2)\{x_2 \mapsto x_1\} =_C g(x_1,a)$ it follows that $g(x_1,a) \equiv_C (a,x_2)$. In other words, $g(x_1,a)$ and $g(a,x_2)$ are in the same equivalence class induced by the instantiation preorder. Therefore, to obtain a minimal set of $\mathtt{gen}_C(s,t)$, one of these terms has to be chosen to represent this set. As choosing $g(x_1,a)$, it follows that $\mathtt{lgg}_C(s,t) = \{g(x_1,a)\}$ is a complete and minimal set of generalizers of s and t.



1.3 Abstract Reduction Systems

The notions, definitions and results presented here follows the approach of Baader [4].

We will use reduction as synonymous of a stepwise execution of a computation. In this work we are interested in building some algorithms to solve the Anti-Unification problem. Thus, reductions $a_o \Longrightarrow a_1 \Longrightarrow \cdots \Longrightarrow a_n$ will be used to express n steps of the execution of a procedure.

Definition 1.24 (Abstract Reduction System)**.** An Abstract Reduction System is a pair $(A, \Longrightarrow)$, where the reduction $\Longrightarrow$ is a binary relation on the set A, i.e., $\Longrightarrow \subseteq A \times A$. Instead of $(a,b) \in \Longrightarrow$ we write $a \Longrightarrow b$.

> *Remark* 1.4. In the literature it is common to use "$\longrightarrow$" to denote a reduction. Nonetheless, to avoid confusion with the notation used in the diagrams that illustrate the instantiating relation between terms we will use a double arrow " $\Longrightarrow$ " instead of "$\longrightarrow$".

Definition 1.25 (Composition of relations)**.** Given two relations $R \subset A \times B$ and $S \subset B \times C$, their composition is defined by

$$R \circ S = \{(x,y) \in A \times C \mid \exists y \in B.(x,y) \in R \wedge (y,z) \in S\}$$

Based on the composition of relation $\Longrightarrow_R$ we define:

$$\begin{array}{lll}
\overset{0}{\Longrightarrow} & := \{(x,x) \mid x \in A\} & \text{identity} \\
\overset{1}{\Longrightarrow} & := \Longrightarrow & \text{one step of the reduction} \\
\overset{i+1}{\Longrightarrow} & := \overset{i}{\Longrightarrow} \circ \overset{1}{\Longrightarrow} & (i+1)\text{-fold composition, } i > 0 \\
\overset{+}{\Longrightarrow} & := \bigcup\limits_{i>0} \overset{i}{\Longrightarrow} & \text{transitive closure} \\
\overset{*}{\Longrightarrow} & := \overset{+}{\Longrightarrow} \cup \overset{0}{\Longrightarrow} & \text{reflexive transitive closure.}
\end{array}$$

Sometimes it will be useful to specify in how many steps one term reduces to other. Then, to write that a reduces to b in n steps, with $n \in \mathbb{N}$, we write $a \overset{n}{\Longrightarrow} b$ induced by the above definition.

Definition 1.26 (Normal form)**.** x is in normal form iff there is no y such that $x \Longrightarrow y$. Also, y is a normal form of x iff $x \overset{*}{\Longrightarrow} y$ and y is in normal form.

Definition 1.27 (Termination)**.** A reduction relation is called **terminaning** if there is no infinite chain $a_0 \Longrightarrow a_1 \Longrightarrow \dots$.

Since we are using reduction as a synonymous of a step-wise execution of a computation, the notion of terminating reduction is paramount for ensure that this computation ends. On other hand, the normal forms of a reduction corresponds to elements in which this computation stops.

When we are studying a problem that only have one solution, (as likes happens to the Syntactic Anti-Unification problem presented in the Chapter 2,) it is interesting to analyse if the procedure developed to solve this problem has only one normal form. To do it, we use the notion of confluence presented in the following.

Definition 1.28 (Confluence)**.** A reduction relation is called **confluent** iff for all x such that $x \overset{*}{\Longrightarrow} y_1$ and $x \overset{*}{\Longrightarrow} y_2$ there exists z such that $y_1 \overset{*}{\Longrightarrow} z$ and $y_2 \overset{*}{\Longrightarrow} z$. A diagram illustrating this property is presented in Figure 1.2.

Unfortunately is not always easy to prove that a reduction is confluent. However, when a relation is terminating there exists an easier way to prove confluence. To present it is necessary first present the notion of locally confluence, given bellow.

Definition 1.29 (Locally Confluent)**.** A reduction is locally confluent iff $x \Longrightarrow y_1$ and $x \Longrightarrow y_2$ then there exist a term z such that $y_1 \overset{*}{\Longrightarrow} z$ and $y_2 \overset{*}{\Longrightarrow} z$. An diagram illustrating this property is presented in Figure 1.2 bellow.

$$\begin{tikzcd}
& x \arrow[dl, Rightarrow, "*"'] \arrow[dr, Rightarrow, "*"] & \\
y_1 \arrow[dr, Rightarrow, "*"'] & & y_2 \arrow[dl, Rightarrow, "*"] \\
& z &
\end{tikzcd}$$

Confluent

$$\begin{tikzcd}
& x \arrow[dl, Rightarrow, "1"'] \arrow[dr, Rightarrow, "1"] & \\
y_1 \arrow[dr, Rightarrow, "*"'] & & y_2 \arrow[dl, Rightarrow, "*"] \\
& z &
\end{tikzcd}$$

Locally Confluent

Fig. 1.2

The next lemma is useful to prove confluence of terminating relations.

Lemma 1.4 (Newman's Lemma c.f. Lemma 2.7.2 in [4])**.** A terminating relation is confluent if its is locally confluent.

When a relation is not confluent then it must have more than one normal form. Sometimes it is not easy to say how many normal forms an term could have. However, the next notions are useful to verify if the number of normal forms is limited, even if a maximum measure to limit this amount of normal forms is not known.

Definition 1.30 (Successors of a term)**.** Given an abstract reduction system $(A, \Longrightarrow)$.

1. b is a **direct successor** of a iff $a \overset{1}{\Longrightarrow} b$.

2. b is a **successor** of a iff $a \overset{n}{\Longrightarrow} b$ for some $n \in \mathbb{N}$.

Definition 1.31 (Finitely branching, globally finite)**.**

1. A relation is **finitely branching** if each element has only finitely many direct successors.
2. A relation is **globally finite** if each element has only finitely many successors.

Lemma 1.5 (Lemma 2.2.4 in [4])**.** A finitely branching relation is globally finite if it is terminating.

Therefore, if a relation is terminating and finitely branching then it must have a finite number of normal forms, since each normal form of a term is also a successor of this term.

Chapter 2

Syntactic Anti-Unification Problem

The Anti-Unification Problem, also called generalization problem, consists in: given two terms s and t, to find a term r such that represent as most as possible both term structure's of s and t. In this chapter we will follow the method proposed by Alpuente et. al. in [1] to solve the Syntactic Anti-Unification Problem.

In Section 2.1 some notions will be initially established in order to define the rule-based anti-unification algorithm $\mathtt{AUnif}_\emptyset$, presented in Figure 2.2, introduced originally in [1]. We will discuss each of the inference rules proposed; how to interpret normal forms of the algorithm and computed solutions of an anti-unification problem $\mathcal{A}\langle s,t\rangle$; how to obtain least general generalizers and also substitutions that maps this generalizer to s and t; and present some examples.

In the Section 2.2 we will study the properties of $\mathtt{AUnif}_\emptyset$ such as termination (Theorem 2.1), confluence (Theorem 2.2) and correctness (Theorem 2.3). We will present more detailed proofs than [1], and also a new lemma (Lemma 2.8) which shows that for the term r outputted by $\mathtt{AUnif}_\emptyset$, it can also gives a way to mapping r to s and t, as is mentioned above.

2.1 $\mathtt{AUnif}_\emptyset$: a rule-based algorithm for anti-unification

In this section we will present a rule based anti-unification algorithm called $\mathtt{AUnif}_\emptyset$, with rules given in Figure 2.2. Given a pair of terms s and t, the goal of $\mathtt{AUnif}_\emptyset$ is output a solution for $\mathcal{A}\langle s,t\rangle$, i.e, a solution for the anti-unification problem between s and t. The idea of the algorithm $\mathtt{AUnif}_\emptyset$ is to recursively simplify the initial problem $\mathcal{A}\langle s,t\rangle$ by application of the simplification rules, obtaining new anti-unification problems, until $\mathcal{A}\langle s,t\rangle$ being completely solved. To describe this anti-unification algorithm precisely it is necessary to establish some auxiliary notations.

Definition 2.1 (Constraint)**.** In the following, a constraint $\{x : s \triangleq t\}$ is composed of a pair of terms s and t and a fresh variable x. This constraint is used to represent the anti-unification problem $\mathcal{A}\langle s,t\rangle$ and the variable x represents a trivial generalizer of s and t.

Definition 2.2 (Index Variable, Set of Index Variable.)**.** Let $\{x : s \triangleq t\}$ being a constraint, we call x an index variable or a variable at the index position of the constraint. Given a set of constraints Q; the set of index variables is defined as

$$\texttt{Index}(Q) = \{x \in \mathcal{X} \mid \text{ exists } \{x : s \triangleq t\} \text{ in } Q\}.$$

Definition 2.3 (Configurations)**.** Configurations are triples of the form $\langle P \mid S \mid \sigma\rangle$, where P is a set of constraints to be solved, S is a set of constraints that are solved and σ is a computed substitution.

The anti-unification algorithm $\texttt{AUnif}_\emptyset$ will operate in configurations. A configuration of the form $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ is called initial configuration and a configuration of the form $\langle\emptyset \mid S \mid \sigma\rangle$ is called a final configuration.

The initial configuration gives $x\,id = x$, the trivial generalizer of s and t. The goal of $\texttt{AUnif}_\emptyset$ is to find a more specific one (least general), given by $x\sigma$, with σ being the substitution of the final configuration and x the index variable of the initial configuration.

Idea of the algorithm: $\texttt{AUnif}_\emptyset$

Input: $\mathcal{A}_\emptyset\langle s,t\rangle$.

$$\begin{tikzcd}
& \texttt{AUnif}_\emptyset - \text{rules} \arrow[d] & \\
\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \arrow[r, Rightarrow] & \ldots \arrow[r, Rightarrow] & \langle\emptyset \mid S \mid \sigma\rangle \\
\texttt{initial state} \arrow[u] & & \texttt{final state} \arrow[u]
\end{tikzcd}$$

Output: $\texttt{lgg}(s,t) = x\sigma$.

Fig. 2.1 Idea of the algorithm: $\texttt{AUnif}_\emptyset$

More precisely, given $\mathcal{A}\langle s,t\rangle$, the anti-unification algorithm $\texttt{AUnif}_\emptyset$ starts with the initial configuration $\langle x : s \triangleq t \mid \emptyset \mid id\rangle$, applies the $\texttt{AUnif}_\emptyset$ simplification rules (in Figure 2.2) on configurations until the rules cannot be applied anymore, obtaining a final configuration $\langle\emptyset \mid S \mid \sigma\rangle$, then the algorithm output $\texttt{lgg}(s,t) = x\sigma$ and also the pair of substitutions

$\overline{\theta} = (\theta_1, \theta_2)$, that is described in Lemma 2.8, such that $x\sigma\overline{\theta} = (s,t)$. The proof that $x\sigma$ is a $\mathtt{lgg}(s,t)$ is given later on, in Theorem 2.3.

In order to easier readability, fix $s,t \in T(\mathcal{X}, \Sigma_\emptyset)$ and $f : n \in \Sigma_\emptyset$.

(Dec) **: Decompose:**

$$\langle P \cup \{x : f(\overline{s_n}) \triangleq f(\overline{t_n})\} \mid S \mid \sigma\rangle \Longrightarrow \left\langle P \cup \begin{Bmatrix} x_1 : s_1 \triangleq t_1, \\ \vdots \\ x_n : s_n \triangleq t_n \end{Bmatrix} \mid S \mid \sigma\{x \mapsto f(\overline{x_n})\}\right\rangle$$

where $x_1, \ldots, x_n$ are fresh variables.

(Sol)**: Solve:** If $\mathtt{root}(s) \neq \mathtt{root}(t)$ and there is no constraint $\{y : s \triangleq t\} \in S$

$$\langle P \cup \{x : s \triangleq t\} \mid S \mid \sigma\rangle \Longrightarrow \langle P \mid S \cup \{x : s \triangleq t\} \mid \sigma\rangle$$

(Rec)**: Recover:** If $\mathtt{root}(s) \neq \mathtt{root}(t)$

$$\langle P \cup \{x : s \triangleq t\} \mid S \cup \{y : s \triangleq t\} \mid \sigma\rangle \Longrightarrow \langle P \mid S \cup \{y : s \triangleq t\} \mid \sigma\{x \mapsto y\}\rangle$$

Fig. 2.2 `AUnif`$_\emptyset$ Simplification Rules

We now describe each rule of the algorithm.

(Dec)**:** Given a constraint $\{x : f(s_1, \ldots, s_n) \triangleq f(t_1, \ldots, t_n)\}$, we have to compute the generalization between s_1 and t_1, s_2 and t_2 and so on, until comparing all the arguments of $f(s_1, \ldots, s_n)$ and $f(t_1, \ldots, t_n)$. Basically, (Dec) delete the constraint

$$\{x : f(s_1, \ldots, s_n) \triangleq f(t_1, \ldots, t_n)\}$$

and replace it with n new constraints $\{x_1 : s_1 \triangleq t_1, \ldots, x_n : s_n \triangleq t_n\}$ and creates a new substitution $\sigma = \{x \mapsto f(x_1, \ldots, x_n)\}$, where $x_1, \ldots, x_n$ are the corresponding index variable of each new created constraint.

Calling $s = f(s_1, \ldots, s_n)$ and $t = f(t_1, \ldots, t_n)$, notice that x is a generalizer of s and t, but $x\sigma = f(x_1, \ldots, x_n)$ is still a generalizer of s and t and preserve more structure of both terms than x, i.e. $x < f(x_1, \ldots, x_n)$. In conclusion, the application of Decompose rule gives a more specific generalizer of s and t.

(Sol)**:** Given a constraint $\{x : s \triangleq t\}$, such that $\mathtt{root}(s) \neq \mathtt{root}(t)$. It is easy to see that s and t only have the trivial generalizer. Then, (Sol) moves the constraint $\{x : s \triangleq t\}$

from the set to unsolved constraints and does not change the computed substitution, since a substitution cannot change the head symbol of s or t.

(Rec): Given a unsolved constraint $\{x : s \triangleq t\}$ such that $\texttt{root}(s) \neq \texttt{root}(t)$, and a solved constraint $\{y : s \triangleq t\}$. Then, it is easy to see that s and t only have trivial generalizers. As there is already a solved constraint with index variable y representing the same anti-unification problem as $\{t : x \triangleq s\}$; (Rec) delete the constraint $\{x : s \triangleq t\}$ of the set of unsolved constraints, do not change the set of solved constraints and add the substitution $\sigma = \{x \mapsto y\}$.

Remark 2.1. Notice that we could have added a rule to simplify the computation of trivial generalizers instead of exhaustively go through all the steps,

$$\textit{\textbf{(Triv)}}\textbf{: Trivial: } \langle P \cup \{x : s \triangleq s\} \mid S \mid \sigma\rangle \Longrightarrow \langle P \mid S \mid \sigma\{x \mapsto s\}\rangle.$$

For example, given $s = f(a) = t$, it is easy to see that the least general generalizer of $f(a)$ and $f(a)$ is itself. The $\texttt{AUnif}_\emptyset$ will need two steps of rule application to obtain the result, as is described below

$$\begin{array}{c}
\langle \{x : f(a) \triangleq f(a)\} \mid \emptyset \mid id\rangle \\
{\scriptstyle (Dec)} \Downarrow \\
\langle \{y : a \triangleq a\} \mid \emptyset \mid \underbrace{\{x \mapsto f(y)\}}_{\sigma_1}\rangle \\
{\scriptstyle (Dec)} \Downarrow \\
\langle \emptyset \mid \emptyset \mid \underbrace{\sigma_1\{y \mapsto a\}}_{\sigma_2}\rangle
\end{array}$$

on the other hand, the rule special rule $(Triv)$ would obtain the result in just one step

$$\langle \{x : f(a) \triangleq f(a)\} \mid \emptyset \mid id\rangle \Longrightarrow_{(Triv)} \{x \mapsto f(a)\}.$$

Notice that if s have more arguments, then $\texttt{AUnif}_\emptyset$ will need a bigger number of steps for stop, while $(Triv)$ always output the solution in just one step.
Since the approach of this work is to study one algorithm that solve the Anti-Unification Problem, $\texttt{AUnif}_\emptyset$ simplifications rules are enough to deal this purpose. Therefore, it is not necessary to add to $\texttt{AUnif}_\emptyset$ a especial rule to applies in initial configurations of the form $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ with $s = t$, since the algorithm will output a correct substitution σ such that $x\sigma = s = t$.

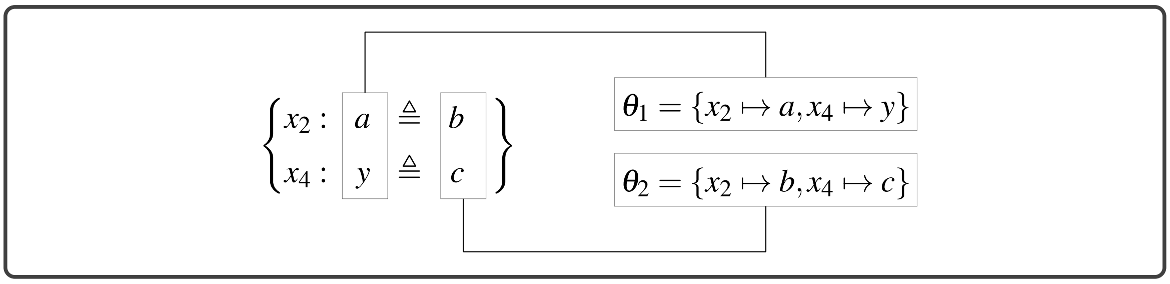
We will now illustrate the algorithm by showing some examples.

Example 2.1. Consider the problem $\mathcal{A}\langle s,t\rangle$, where $s = f(f(a,y),a)$ and $t = f(f(b,c),b)$. The `AUnif`$_\emptyset$ computes as follows:

$$\begin{array}{ll}
\langle\{x : f(f(a,y),a) \triangleq f(f(b,c),b)\} \mid \emptyset \mid id\rangle, & \text{initial configuration} \\
\Big\Downarrow (Dec) & \\
\left\langle\left\{\begin{array}{l} x_1 : f(a,y) \triangleq f(b,c) \\ x_2 : a \triangleq b \end{array}\right\} \mid \emptyset \mid \underbrace{\{x \mapsto f(x_1,x_2)\}}_{\sigma_1}\right\rangle, & x_1, x_2 \text{ fresh variables} \\
\Big\Downarrow (Sol) & \\
\langle\{x_1 : f(a,y) \triangleq f(b,c)\} \mid \{x_2 : a \triangleq b\} \mid \sigma_1\rangle & \\
\Big\Downarrow (Dec) & \\
\left\langle\left\{\begin{array}{l} x_3 : a \triangleq b \\ x_4 : y \triangleq c \end{array}\right\} \mid \{x_2 : a \triangleq b\} \mid \underbrace{\sigma_1\{x_1 \mapsto f(x_3,x_4)\}}_{\sigma_2}\right\rangle, & \begin{array}{l} x_2, x_3 \text{ fresh variables and} \\ \sigma_2 = f(f(x_3,,x_4),x_2) \end{array} \\
\Big\Downarrow (Rec) & \\
\langle\{x_4 : y \triangleq c\} \mid \{x_2 : a \triangleq b\} \mid \underbrace{\sigma_2\{x_3 \mapsto x_2\}}_{\sigma_3}\rangle, & \sigma_3 = \{x \mapsto f(f(x_2,x_4),x_2)\} \\
\Big\Downarrow (Dec) & \\
\left\langle\emptyset \mid \left\{\begin{array}{l} x_2 : a \triangleq b \\ x_4 : y \triangleq c \end{array}\right\} \mid \sigma_3\right\rangle, & \text{final configuration}
\end{array}$$

Therefore, `AUnif`$_\emptyset$ output $x\sigma_3 = f(f(x_2,x_3),x_2)$ that will be proven to be the least general generalizer in Theorem 2.3. To obtain $\overline{\theta} = (\theta_1,\theta_2)$ such that $x\sigma\theta_1 = s$, and $x\sigma\theta_2 = t$ it is enough to observe the set of solved constraints of the final configuration. The construction is as follows.

The right side of the solved constraints gives $\theta_1 = \{x_2 \mapsto a, x_4 \mapsto y\}$ and the left side gives $\theta_2 = \{x_2 \mapsto b, x_4 \mapsto c\}$, as is shown in the figure below:


$x_2 : a \triangleq b$
$x_4 : y \triangleq c$
$\theta_1 = \{x_2 \mapsto a, x_4 \mapsto y\}$
$\theta_2 = \{x_2 \mapsto b, x_4 \mapsto c\}$

In fact,

$$x\sigma\theta_1 = f(f(x_2,x_4),x_2)\theta_1 = f(f(a,y),a) = s,$$
$$x\sigma\theta_2 = f(f(x_2,x_4),x_2)\theta_2 = f(f(b,c),b) = t.$$

Conflict position and pair.

The concept of conflict pair, given below, is important to identify where s and t diverges in their structures. Since $\mathtt{lgg}(s,t)$ is the term that have the most structure in common with s and t, it is useful to know this concept.

Definition 2.4. Given terms $s,t \in T(\mathcal{X},\Sigma_\emptyset)$,

a) A position $p \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$ is called a **conflict position** of s and if

$$\mathtt{root}(s|_p) \neq \mathtt{root}(t|_p)$$

and for all $q < p$, $\mathtt{root}(s|_q) = \mathtt{root}(t|_q)$;

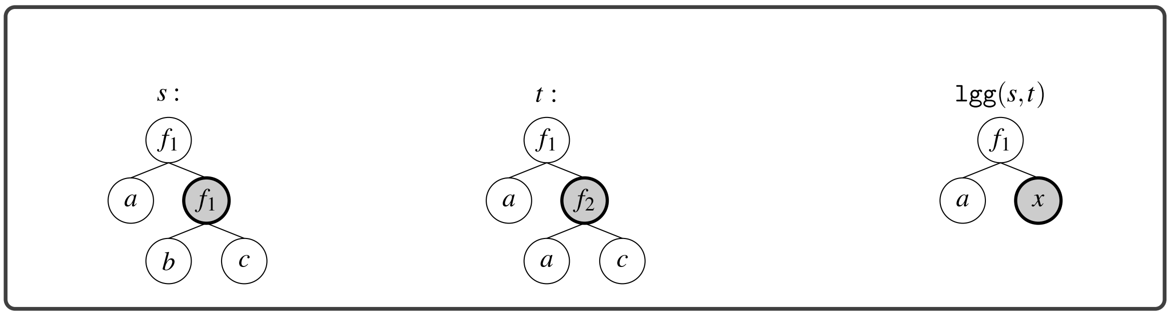
b) The pair (u,v) is called a **conflict pair** of s and t if there exists at least one conflict position p of t such that $u = s|_p$ and $v = t|_p$.

Example 2.2. Let $f_1, f_2 \in \Sigma_\emptyset$ two binary function symbols. Take $s = f_1(a, f_1(b,c))$ and $t = f_1(a, f_2(a,c))$, then $p = 2$ is the unique conflict position of s and t and

$$(u,v) = (f_1(b,c), f_2(a,c))$$

is the respective conflict pair of s and t.

The conflict position identifies where s and t diverges in their structure. Note that there is no non-trivial generalizer for u and v. Then, the bottom of the position 2 of a least general generalizer of s and t must be a variable, as is it illustrated bellow.



The idea to determine the $\mathtt{lgg}(s,t)$ is by comparing the root symbols of each pair of subterms $(s|_p, t|_p)$ of s and t, with $p \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$, going deeper in their term tree structure until we find a conflict pair of subterms. Moreover, $(s|_p, t|_p)$ being a conflict pair of positions means there is no reason to analyse a position $q > p$, since s and t structures diverge before it. For instance, note that $2.2 \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$ and $s|_{2.2} = t|_{2.2} = c$, but there is no generalizer r of s and t such that $2.2 \in \mathtt{pos}(r)$, due to the conflict position $2 < 2.2$.

The measure defined bellow it is useful to prove termination of $\mathtt{AUnif}_{\emptyset}$ in the next section.

Definition 2.5. Given a configuration $\langle P \mid S \mid \sigma \rangle$ we define the following measure

$$m(\langle P \mid S \mid \sigma \rangle) = |P| = \sum_{\{x:s \triangleq t\}} |s| + |t|$$

where $|s|$ denotes the size of terms s, i.e.,the number of symbols in s.

2.2 Properties of `Aunif`

Now, in this section we present the main properties of the $\mathtt{AUnif}_{\emptyset}$ algorithm, such as termination (Theorem 2.1), confluence of the rules (Theorem 2.2) and correctness (Theorem 2.3). These results were initially proven in [1], here we present more detailed versions of the proofs. Further, let $\mathcal{A}\langle s,t \rangle$ being an AUP, we and extend the results of correctness and completeness to the set of unsolved constraints S of the final configuration of $\mathtt{AUnif}_{\emptyset}(s,t)$, showing its mapping from the computed solutions $x\sigma$ to s and t. We also present some examples.

Theorem 2.1 (Termination of $\mathtt{AUnif}_{\emptyset}$ [1])**.** Let $\mathcal{A}\langle s,t \rangle$ be an AUP. Every derivation of $\mathtt{AUnif}_{\emptyset}$ starting from $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle$ terminates in a final configuration of the form $\langle \emptyset \mid S \mid \sigma \rangle$.

Proof. Consider the measure $m(\langle P \mid S \mid \sigma \rangle)$ given in the Definition 2.5.We will show that if

$$C_1 = \langle P_1 \mid S_1 \mid \sigma_1 \rangle \Longrightarrow_{\mathtt{AUnif}_{\emptyset}} \langle P_2 \mid S_1 \mid \sigma_2 \rangle = C_2,$$

then $m(C_1) > m(C_2)$.

We proceed by analysing each rule:

Case *(R) = (Dec)*: If the rule applied is Decompose, we have

$$P_1 = P \cup \{x : f(\overline{s_n}) \triangleq f(\overline{t_n})\} \text{ and}$$
$$P_2 = P \cup \{x_1 : s_1 \triangleq t_1, \ldots, x_n : s_n \triangleq t_n\}.$$

Then, $m(C_1)$ and $m(C_2)$ are as follows:

$$\begin{aligned} m(C_1) &= |P_1| \\ &= |P| + |\{f(s_1,\dots,s_n), f(t_1,\dots,t_n)\}| \\ &= |P| + |f(s_1,\dots,s_n)| + |f(t_1,\dots,t_n)| \\ &= |P| + \sum_{i=1}^{n}(|s_i| + |t_i|) + 2 \\ &> |P| + \sum_{i=1}^{n}(|s_i| + |t_i|) \\ &= |P_2| = m(C2) \end{aligned}$$

Case *(Sol)* and *(Rec)* : If the rule applied is Solve or Recover we have $P_1 = P \cup \{x : s \triangleq t\}$ and $P_2 = P$, then the result follows immediately.

□

Since the $\texttt{AUnif}_\emptyset$ rules applications are non-deterministic it is necessary to prove confluence. The next lemmas are presented to easier the confluence proof easier.

Lemma 2.1 (Uniqueness of generalization variables [1])**.** Let $\mathcal{A}\langle s,t\rangle$ be an AUP. If there is a derivation $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$ then y does not appear in any constraint in P or S.

Proof. The complete proof can be found in Appendix A, Lemma A.1. □

Remark 2.2. The $\texttt{AUnif}_\emptyset$ algorithm is non-deterministic. Multiple rules can be applied in a configuration in each step of computation. Notice, however, that the left-hand sides of the rules in Figure 2.2 do not clash, that is, it is not possible to apply different rules in the same constraint.
For example, given a configuration $C = \langle\{x_1 : s_1 \triangleq t_1, x_2 : s_2 \triangleq t_2\} \mid S \mid \sigma\rangle$ there are 2 different ways to apply a rule from $\texttt{AUnif}_\emptyset$ in C, one for each unsolved constraint. That is

$$\begin{tikzcd} & C \arrow[dl, Rightarrow, "(R_1)"'] \arrow[dr, Rightarrow, "(R_2)"] & \\ C_1 & & C_2 \end{tikzcd}$$

where (R_1) is a rule applied to contraint $\{x_1 : s_1 \triangleq t_1\}$ and (R_2) is a rule applied to constraint $\{x_1 : s_2 \triangleq t_2\}$. Also, $C_1 = \langle P_1 \cup \{x_2 : s_2 \triangleq t_2\}\rangle$ and $C_2 = \langle P_2 \cup \{x_1 : s_1 \triangleq t_1\}\rangle$. Likewise, it does not matter which rules are first applies in C, since the set of rules of $\texttt{AUnif}_\emptyset$ is confluent, as will be proved in Theorem 2.2.

Theorem 2.2 (Confluence of $\mathtt{AUnif}_\emptyset$ [1])**.** Let $\mathcal{A}\langle s,t\rangle$ be an AUP. Then every derivation $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle\emptyset \mid S \mid \sigma\rangle$ is such that $\langle\emptyset \mid S \mid \sigma\rangle$ is unique up to variable renaming.

Proof. Given $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \Longrightarrow^*_{\mathtt{AUnif}_\emptyset} \langle P \mid S \mid \sigma\rangle = C$. Suppose at this step we have a bifurcation

$$\begin{tikzcd}
 & C \arrow[dl, Rightarrow, "(R_1)"'] \arrow[dr, Rightarrow, "(R_2)"] & \\
C_1 & & C_2
\end{tikzcd}$$

By Newman's Lemma, once $\mathtt{AUnif}_\emptyset$ is terminating, to prove confluence is enough to prove local confluence, i.e., it is sufficient show that exist C_3 such that

$$\begin{tikzcd}
C_1 \arrow[dr, Rightarrow, "*", "\mathtt{AUnif}_\emptyset"'] & & C_2 \arrow[dl, Rightarrow, "*"', "\mathtt{AUnif}_\emptyset"] \\
 & C_3 &
\end{tikzcd}$$

The proof proceeds by analysing the rule (R_1) applied in $C = \langle P \mid S \mid \sigma\rangle$.

Case $(\boldsymbol{R}_1)$ = *(Dec)*:

Then $P = P_1 \cup \{y : f(s_1,\dots,s_n) \triangleq f(t_1,\dots,t_n)\}$. From Remark 2.2 we already notice that if a rule (R_2) can be applied in P, it is because it was applied in another constraint, i.e., in a constraint from P_1. Then (R_2) is an $\mathtt{AUnif}_\emptyset$'s reduction rule that acts in some constraint from P_1. It follows that

$$C_1 = \langle P_1 \cup \{y_1 : s_1 \triangleq t_1,\dots,y_n : s_n \triangleq t_n\} \mid S \mid \sigma\sigma_1\rangle,$$
$$C_2 = \langle P_1' \cup \{y : f(s_1,\dots,s_n) \triangleq f(t_1,\dots,t_n)\} \mid S' \mid \sigma\sigma_2\rangle,$$

where $\sigma_1 = \{y \mapsto f(\overline{y_n})\}$ and σ_2 is a substitution computed by (R_2).

Now applying (R_2) in P_1 from C_1 we obtain the configuration C_3 given as follows:

$$C_3 = \langle P_1' \cup \{y_1 : s_1 \triangleq t_1,\dots,y_n : s_n \triangleq t_n\} \mid S' \mid \sigma\sigma_1\sigma_2\rangle$$

Applying (Dec) in C_2 we obtain the configuration C_4 given as follows:

$$C_4 = \langle P_1' \cup \{y_1 : s_1 \triangleq t_1,\dots,y_n : s_n \triangleq t_n\} \mid S' \mid \sigma\sigma_2\sigma_1\rangle$$

The Lemma 2.1 implies that the index variables $y, y_1,\dots,y_n$ does not appear in any constraint in P_1. Furthermore, Lemma 2.1 also hence that any index variables of the constraints in P_1 appears in any constraint of $s_1,\dots,s_n$ and $t_1,\dots,t_n$. The Figure 2.2 yields that the domain and the range of variables of each substitution created by an application of each rule of $\mathtt{AUnif}_{\emptyset}$ depends on the constraint which this rule was applied and the constraints which this rules created, then

$$(\mathtt{dom}(\sigma_1)\cup v\mathtt{ran}(\sigma_1))\cap(\mathtt{dom}(\sigma_2)\cup \mathtt{vran}(\sigma_2))=\emptyset.$$

Therefore, $\sigma\sigma_1\sigma_2=\sigma\sigma_2\sigma_1$. It follows that $C_3=C_4$.

Then:

$$\begin{tikzcd}
& \langle P_1\cup\{y: f(\overline{s_n})\triangleq f(\overline{t_n})\}\mid S\mid\sigma\rangle \arrow[dl, Rightarrow, "(Dec)"'] \arrow[dr, Rightarrow, "(R_2)"] & \\
C_1=\langle P_1\cup\{y_1: s_1\triangleq t_1,\dots,y_n: s_n\triangleq t_n\}\mid S\mid\sigma\sigma_1\rangle \arrow[dr, Rightarrow, "(R_2)"'] & & \langle P_1'\cup\{y: f(\overline{s_n})\triangleq f(\overline{t_n})\}\mid S'\mid\sigma\sigma_2\rangle=C_2 \arrow[dl, Rightarrow, "(Dec)"] \\
& C_3=\langle P_1'\cup\{y_1: s_1\triangleq t_1,\dots,y_n: s_n\triangleq t_n\}\mid S'\mid\sigma\sigma_1\sigma_2\rangle=C_4 &
\end{tikzcd}$$

Case (R_1) = *(Rec)*:

Then $P=P_1\cup\{y: s\triangleq t\}$ and $S=S_1\cup\{z: s\triangleq t\}$. Since the case where (Dec) applies in C was done in the previous case, then we can suppose $(R_2)=(Sol)$ or $(R_2)=(Rec)$. The rule (R_2) will act in some constraint in the set of unsolved constraints P_1 of the configuration C_1.

$$\begin{aligned}
C_1&=\langle P_1\mid S_1\cup\{z: s\triangleq t\}\mid\sigma\sigma_1\rangle,\\
C_2&=\langle P_1'\cup\{y: s\triangleq t\}\mid S_1\cup\{z: s\triangleq t\}\cup S_2\mid\sigma\sigma_2\rangle,
\end{aligned}$$

where $\sigma_1=\{y\mapsto z\}$ and σ_2 is a variable renaming given by (R_2) or the identity substitution, depending on if (R_2) is (Rec) or (Sol) and S_2 is some possible constraint added to the set of unsolved constraints by the application of rule (R_2) in C.

Now, applying (R_2) in C_1 we obtain the configuration C_3 given as follows:

$$C_3=\langle P_1'\mid S_1\cup S_2\cup\{z: s\triangleq t\}\mid\sigma\sigma_1\sigma_2\rangle.$$

Applying (Rec) in C_2 we obtain the configuration C_4 given as follows:

$$C_4=\langle P_1'\mid S_1\cup S_2\cup\{z: s\triangleq t\}\mid\sigma\sigma_2\sigma_1\rangle.$$

If $(R_2) = (Sol)$, then $\sigma_2 = id$ and $C_3 = C_4$ follows immediately. If $(R_2) = (Rec)$, then Lemma 2.1 yields $\sigma\sigma_1\sigma_2 = \sigma\sigma_2\sigma_1$, then $C_3 = C_4$. In both cases we obtain

$$C_1 \Longrightarrow C_3 \Longleftarrow C_2.$$

Finally,

Case *(R*$_1$*)= (R*$_2$*) = (Sol)* :

Since the cases of (Dec) and (Rec) applications were analysed, remains only the case where $C \Longrightarrow_{(Sol)} C_1$ and $C \Longrightarrow_{(Sol)} C_2$. Since (Sol) has a condition under the set of solved constraints, and the application of this rule changes this set, then this case is divided in two:

1. $P = P_1 \cup \{y : s_1 \triangleq t_1, z : s_2 \triangleq t_2\}$. Without loss of generality, we obtain the configurations C_1 and C_2 as follows:

$$C_1 = \langle P_1 \cup \{z : s_2 \triangleq t_2\} \mid S \cup \{y : s_1 \triangleq t_1\} \mid \sigma\rangle,$$
$$C_2 = \langle P_1 \cup \{y : s_1 \triangleq t_1\} \mid S \cup \{z : s_2 \triangleq t_2\} \mid \sigma\rangle.$$

Applying (Sol) in C_1 and C_2, both reduces to the configuration given as follows:

$$\langle P_1 \mid \{y : s_1 \triangleq t_1, z : s_2 \triangleq t_2\} \mid \sigma\rangle$$

and the results follows.

2. $P = P_1 \cup \{y : s \triangleq t; z : s \triangleq t\}$. Without loss of generality, we obtain the configurations C_1 and C_2 as follows:

$$C_1 = \langle P_1 \cup \{z : s \triangleq t\} \mid \{y : s \triangleq t\} \mid \sigma\rangle,$$
$$C_2 = \langle P_1 \cup \{y : s \triangleq t\} \mid \{z : s \triangleq t\} \mid \sigma\rangle.$$

Now, applying (Rec) in C_1 we obtain the configuration C_3 given as follows:

$$C_3 = \langle P' \mid S \cup \{y : s \triangleq t\} \mid \sigma\sigma_1\rangle.$$

Applying (Rec) in C_2 we obtain the configuration C_4 given as follows:

$$C_4 = \langle P' \mid S \cup \{z : s \triangleq t\} \mid \sigma\sigma_2\rangle,$$

where $\sigma_1 = \{z \mapsto y\}$ and $\sigma_2 = \{y \mapsto z\}$. Therefore $C_3 = C_4$ up to variable renaming.

□

Now we will analyse the completeness property of the algorithm $\texttt{AUnif}_\emptyset$. For that we will present some lemmas to simplify the proof of Theorem 2.3 for correctness.

Lemma 2.2 (Range of Substitutions [1]). Given terms $s,t \in T(\mathcal{X},\Sigma_\emptyset)$ and a fresh variable x such that $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \mid S \mid \sigma\rangle$. Then, $\texttt{Index}(P \cup S) \subseteq \texttt{vran}(\sigma) \cup \{x\}$ and $\texttt{vran}(\sigma) \subseteq \mathcal{V}(x\sigma)$.

Proof. The complete proof can be found in Appendix A, Lemma A.2. □

Lemmas 2.3 and 2.4 below, establish auxiliary properties that are useful to understand the definition of conflict pair. Given the problem $\mathcal{A}\langle s,t\rangle$, these lemmas relate the set of unsolved constraints and also the set of solved constraints of each configuration obtained during a derivation of $\texttt{AUnif}_\emptyset$.

Lemma 2.3 (Lemma 3 in [1]). Let $\mathcal{A}\langle s,t\rangle$ be an AUP. There is a derivation of the form $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle\{y : u \triangleq v\} \cup P \mid S \mid \sigma\rangle$ if, and only if, there exists a position $p \in \texttt{pos}(s) \cap \texttt{pos}(t)$ such that $s|_p = u$, $t|_p = v$, and for all position $p' < p$, $\texttt{root}(s|_{p'}) = \texttt{root}(t|_{p'})$.

Proof. The complete proof can be found in Appendix A, Lemma A.3

□

Remark 2.3. Notice that the constraints $\{x : s \triangleq t\}$ and $\{x : t \triangleq s\}$ represent the same problem $\mathcal{A}\langle s,t\rangle$. However, the initial configuration $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ fixes the sides of the subterms of s and t in each constraint produced by the application of algorithm's rules, i.e, the left side of each constraint will be a subterm of s and the right side of each constraint will be a subterm of t, as it was proved by Lemma 2.3. If the sides of subterms are changed during the algorithm application, then the computed pair of substitutions $\overline{\theta}$ that should map $x\sigma$ to s and t may be such that $x\sigma\,\overline{\theta} \neq (s,t)$, i.e, the mapping does not produce a generalizer. Then, the order which each term appear in a constraint matters.

The Lemma bellow uses the definition of conflict position and pair given in Definition 2.4.

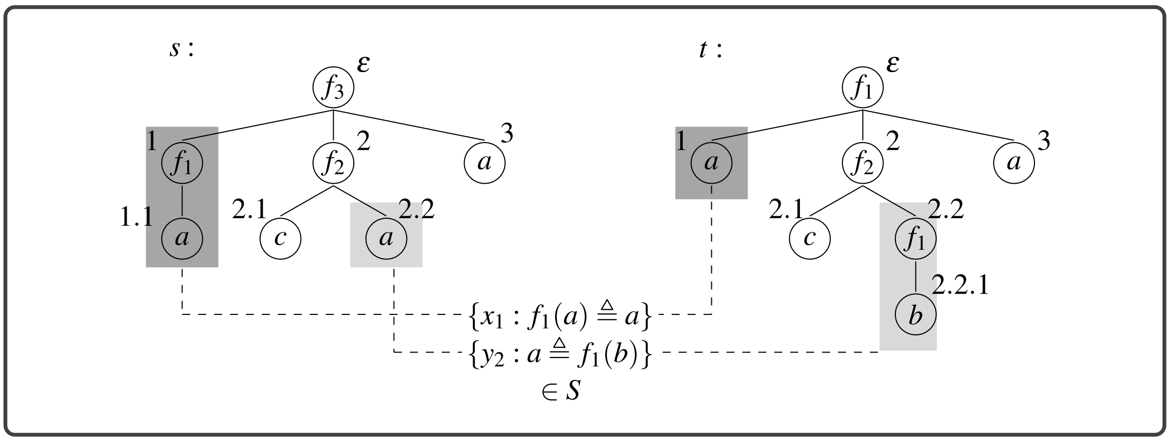
Lemma 2.4 (Lemma 4 in [1]). Let $\mathcal{A}\langle s,t\rangle$ be an AUP. There is a derivation of the form $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \mid \{y : u \triangleq v\} \cup S \mid \sigma\rangle$ if, and only if, there exists a conflict position p of s and t such that $s|_p = u$ and $t|_p = v$.

Proof. The complete proof can be found in Appendix A, Lemma A.4 □

Example 2.3. As taking $\{f_3 : 3, f_2 : 2, f_1 : 1\} \subseteq \Sigma_\emptyset$ and terms $s = f_3(f_1(a), f_2(c,a), a)$ and $t = f_3(a, f_2(c, f_1(b)), a)$, therefore there are two conflict positions in s and t:

1. $p_1 = 1$, and its respective conflict pair of subterms is $(f_1(a), a)$, and
2. $p_2 = 2.2$, and its respective conflict pair of subterms is $(a, f_1(b))$.

Then by Lemma 2.4 there exists a derivation $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \mid S \mid \sigma\rangle$ and fresh variables x_1, y_2 such that $\{x_1 : f_1 \triangleq a,\ y_2 : a \triangleq f_1(b)\} \in S$. In other words, the Lemma 2.4 says that for each conflict pair of s and t there exists a reduction of $\texttt{AUnif}_\emptyset$ such that the constraint representing the Anti-Unification Problem between the subterms of s and t given by this conflict pair is already solved, as is showed in the figure bellow.



In fact, applying (Dec) in the initial configuration, $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle$, we obtain the configuration C given as follows:

$$C = \langle \left\{ \begin{array}{l} x_1 : f_1(a) \triangleq a, \\ x_2 : f_2(c,a) \triangleq f_2(c, f_1(b)), \\ x_3 : a \triangleq a \end{array} \right\} \mid \emptyset \mid \underbrace{\{x \mapsto f_3(x_1,x_2,x_3)\}}_{\sigma} \rangle$$

After that we apply (Sol) in the constraint with index variable x_1, solving it to the set of solved constraints. By applying (Dec) in the constraint with index variable x_2, we obtain the constraints with index variables y_1 and y_2 given below. Therefore, it follows that $C \overset{2}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} C_1$, where:

$$C_1 = \langle \left\{ \begin{array}{l} y_1 : c \triangleq c, \\ y_2 : a \triangleq f_1(b), \\ x_3 : a \triangleq a \end{array} \right\} \mid \{x_1 : f_1(a) \triangleq a\} \mid \underbrace{\sigma\{x_2 \mapsto f_2(y_1,y_2)\}}_{\sigma_1} \rangle$$

Then applying (Sol) in the constraint with index y_2, solving it to the set of solved constraints, we obtain that $C_1 \Longrightarrow_{(Sol)} C_2$, given as follows:

$$C_2 = \langle \left\{ \begin{matrix} y_1 : c \triangleq c, \\ x_2 : a \triangleq a \end{matrix} \right\} \mid \left\{ \begin{matrix} x_1 : f_1(a) \triangleq a, \\ y_2 : a \triangleq f_1(b) \end{matrix} \right\} \mid \sigma_1 \rangle$$

Therefore, $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} C_2$ and the set of solved constraints of C_2 is of the form that the Lemma 2.4 claims that should exist in the derivation tree of $\texttt{AUnif}_\emptyset(s,t)$.

The Lemma 2.5 presented next was not stated in [1]. However, it is necessary prove it in order to easy the proofs of the lemmas used to prove the main theorem of this section, Theorem 2.3, which states the correctness of $\texttt{AUnif}_\emptyset$ and was originally presented in [1].

Lemma 2.5. Given terms $s,t \in T(\mathcal{X}, \Sigma_\emptyset)$. If there exists a derivation

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle \{y : u \triangleq v\} \cup P \mid S \mid \sigma \rangle$$

then there exists a position $p \in \texttt{pos}(s) \cap \texttt{pos}(t)$ such that

$$(x\sigma|_p) = y, \quad s|_p = (x\sigma)|_p\{y \mapsto u\} = u, \quad t|_p = (x\sigma)|_p\{y \mapsto v\} = v.$$

Proof. The complete proof can be found in Appendix A, Lemma A.5. □

Given $\mathcal{A}\langle s,t \rangle$ the Lemma 2.5 relates the structures of s and t with each output term $x\sigma$ obtained by an application of a rule of Figure 2.2.

Since Lemma 5 of [1] has two items and their proofs are huge, we decided to split this result in two, to ease readability. Then Lemma 2.6 and Lemma 2.7 given as follows are versions of Lemma 5 in [1].

The next lemma says that given $\mathcal{A}\langle s,t \rangle$, each step of application of $\texttt{AUnif}_\emptyset$ gives a generalizer of s and t.

Lemma 2.6. Let $s,t \in T(\mathcal{X}, \Sigma_\emptyset)$. If $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \mid S \mid \sigma \rangle$ then $x\sigma \in \texttt{gen}(s,t)$.

Proof. By induction on the length n of the reduction $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{n}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \mid S \mid \sigma \rangle$.

We will present one interesting case of the inductive step; the basic case which $n = 0$ and the other cases of its inductive step can be found in Appendix A, Lemma A.6.

Inductive Step: Supposing that

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{n}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P' \mid S' \mid \sigma' \rangle \Longrightarrow_{(R)} \langle P \mid S \mid \sigma \rangle$$

We want to show that $x\sigma \in \texttt{gen}(s,t)$. Notice that $\sigma = \sigma'\delta$ for some δ that will obtained in the last step. The proof proceeds by analysing the rule (R) applied in this last step.

Case *(R) = (Dec)*:

Then the reduction is of the form

$$\begin{aligned}\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P_1 \cup \{y : f(\overline{u_m}) \triangleq f(\overline{v_m})\} \mid S \mid \sigma'\rangle \\ &\Longrightarrow_{(Dec)} \langle P_1 \cup \{y_1 : u_1 \triangleq v_1, \ldots, y_m : u_m \triangleq v_m\}\} \mid S \mid \sigma'\{z \mapsto f(\overline{y_m})\}\rangle,\end{aligned}$$

where,

$$\begin{aligned}&P' = P_1 \cup \{y : f(\overline{u_m}) \triangleq f(\overline{v_m}))\}, P = P_1 \cup \{y_1 : u_1 \triangleq v_1, \ldots, y_m : u_m \triangleq v_m\}, \\ &S = S' \text{ and } \sigma = \sigma'\{y \mapsto f(\overline{y_m})\}.\end{aligned}$$

Let $\delta = \{y \mapsto f(\overline{y_m})\}$. Then $\sigma = \sigma'\delta$.

The definition of (Dec) in Figure 2.2 yields that $y_1, \ldots, y_m$ are fresh variables. Hence, $y_1, \ldots, y_m \notin \texttt{dom}(\sigma')$. Besides, Lemma 2.2 yields that y is in the range of variables of σ', i.e, $y \in \texttt{vran}(\sigma')$, and Lemma 2.1 yields that y does not appears in any constraint of $P_1 \cup S$. Then $y \notin \texttt{dom}(\sigma')$. By Lemma 2.5, there is a position $p \in \texttt{pos}(s) \cap \texttt{pos}(t)$ such that $s|_p = f(\overline{u_m})$, $t|_p = f(\overline{v_m})$ and $x\sigma'|_p = y$. It follows that $x\sigma|_p = f(\overline{y_m})$.

By the induction hypotheses it follows that $x\sigma' \in \texttt{gen}(s,t)$, hence there exists a pair of substitutions $\overline{\theta} = (\theta_1, \theta_2)$ such that $x\theta_1 = s$ and $x\theta_2 = t$. Then,

$$x\sigma'\theta_1|_p = f(\overline{s_m}), \quad x\sigma'\theta_2|_p = f(\overline{t_m}).$$

Since $x\sigma'|_p = y$, it follows that $y\theta_1 = f(\overline{s_m})$ and $y\theta_2 = f(\overline{t_m})$.

Notice that $y \in \texttt{dom}(\sigma)$ and $\texttt{root}(y\sigma) = \texttt{root}(y\sigma\theta_1) = \texttt{root}(y\sigma\theta_2)$. Since $y_1, \ldots, y_2$ are fresh variables we can suppose without lost of generality that $y_1, \ldots, y_m \notin \texttt{dom}(\theta_1) \cup \texttt{dom}(\theta_2)$, then constructing the substitutions τ_1 and τ_2 as follows:

$$\tau_1 = \begin{Bmatrix} x \mapsto x\theta_1, y \neq x \in \texttt{dom}(\theta) \\ y_i \mapsto u_i \end{Bmatrix} \text{ and } \tau_2 = \begin{Bmatrix} x \mapsto x\theta_2, y \neq x \in \texttt{dom}(\theta) \\ y_i \mapsto v_i \end{Bmatrix}$$

with $i = 1, \ldots, m$, then the pair of substitutions $\overline{\tau} = (\tau_1, \tau_2)$ is such that

$$x\sigma\overline{\tau} = x\sigma'\overline{\theta} = (s,t).$$

Therefore, $x\sigma \in \mathtt{gen}(s,t)$.

$\square$

The next lemma uses the notion of equivalence "$\equiv$" between terms given by the instantiation ordering $>$ (cf. Definition 1.17). From Remark 1.2 it follows that $s \equiv t$ iff s is equal to t with exception for variable renaming.

Lemma 2.7. Let $s,t \in T(\mathcal{X},\Sigma_\emptyset)$. If $u \in \mathtt{gen}(s,t)$ then there exists a derivation

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P \mid S \mid \sigma\rangle$$

such that $u \equiv x\sigma$

Proof. By induction on the structure of u. We will present just one interesting case of the inductive step; the basic step of the induction which u is a variable and the other cases of the inductive step can be found in Appendix A, Lemma A.7.

Inductive Step: Let $u = f(u_1,\ldots,u_n)$, for an n-ary free function symbol f. Suppose that the result follows for generalizers u_i of s_i and t_i with structure simpler than u.

Since $u \in \mathtt{gen}(s,t)$ it follows that there exists a pair of substitutions $\overline{\theta} = (\theta_1,\theta_2)$ such that $u\overline{\theta} = (s,t)$. Then both s and t have as root symbol the n-ary function symbol f. Which follows that $s = f(s_1,\ldots,s_n)$ and $t = f(t_1,\ldots,t_n)$ for some s_i, $t_i \in T(\mathcal{X},\Sigma_\emptyset)$, with $i = 1,\ldots,n$. Therefore, the initial configuration is

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle = \langle \{x : f(\overline{s_n}) \triangleq f(\overline{t_n})\} \mid \emptyset \mid id\rangle.$$

We want to show that exists a derivation

$$\langle \{x : f(\overline{s_n}) \triangleq f(\overline{t_n})\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P \mid S \mid \sigma\rangle \tag{1}$$

such that $x\sigma \equiv u = f(u_1,\ldots,u_n)$. We will show the existence of this derivation constructing it.

Notice, (Dec) apply in the initial configuration. Therefore, the first step of the derivation (1) is of the form:

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \Longrightarrow \langle \{x_1 : s_1 \triangleq t_1,\ldots,x_n : s_n \triangleq t_n\} \mid \emptyset \mid \underbrace{\{x \mapsto f(x_1,\ldots,x_n)\}}_{\sigma_0}\rangle \tag{2}$$

To obtain the next steps it is necessary use the induction hypotheses. First, let's prove the claim bellow.

Claim: $u_i \in \texttt{gen}(s_i, t_i)$, for all $1 \leq i \leq n$.

In fact, is enough note that

$$u\theta_1 = f(u_1\theta_1, \ldots, u_n\theta_1) = f(s_1, \ldots, s_n),$$

$$u\theta_2 = f(u_1\theta_2, \ldots, u_n\theta_2) = f(t_1, \ldots, t_n).$$

Hence, $u_i\theta_1 = s_i$ and $u_i\theta_2 = t_i$. Then, u_i is a generalizer of s_i and t_i for every $1 \leq i \leq n$.

Therefore, the induction hypotheses implies that for each $i = 1, \ldots, n$ there exists a derivation R_i composed of the rules $\texttt{AUnif}_{\emptyset}$ given in Figure 2.2 such that

$$\langle x_i : s_i \triangleq t_i \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{R_i} \langle C_i \mid S_i \mid \sigma_i \rangle \tag{3}$$

such that $u_i \equiv x_i\sigma_i$.

The idea of this proof is after applies (Dec) in the initial configuration, as is shown in (2), and after sequentially apply each of the derivations chains given in (3), in order to obtain a configuration $\langle P \mid S \mid \sigma \rangle$ with $\sigma = \sigma_0\sigma_1 \ldots \sigma_n$, such that

$$\begin{aligned} x\sigma &= x\sigma_0 \ldots \sigma_n \\ &= (x\sigma_0)\sigma_1 \ldots \sigma_n \\ &= f(x_1, \ldots, x_n)\sigma_1 \ldots \sigma_n \\ &= f(x_1\sigma_1, \ldots, x_n\sigma_n) \equiv f(u_1, \ldots, u_n) \end{aligned}$$

However, it is necessary to ensure that this combination of derivations will be done in a way that each substitution σ_k obtained from the derivation R_k dos not change the value of the term $x_j\sigma_j \equiv u_j$ obtained by the derivation R_j, i.e, it is necessary to ensure that $f(x_1, \ldots, x_n)\sigma_1 \ldots \sigma_n = f(x_1\sigma_1, \ldots, x_n\sigma_n)$. Notice that $\texttt{dom}(\sigma_i) = \{x_i\}$ and by Lemma 2.2 it follows that $\texttt{vran}(\sigma_i) \subseteq \mathcal{V}(x\sigma_i)$. Then, it is enough to compare the variables of each u_j and u_k for every $k \neq j$. This analyse of variables is also important to ensure that the rules of Figure 2.2 were applied correctly as combine the derivations of (3), because just joint these rules without some adaptations could generates a application of (Sol) in a constraint in which (Rec) should apply instead. There are two cases of variables to analyse, studied in the following.

Case 1: If $\mathcal{V}(u_j) \cap \mathcal{V}(u_k) = \emptyset$ for every $j \neq k$.

Since $\{x_k : s_j \triangleq t_j\}$ and $\{x_k : s_k \triangleq t_k\}$ were obtained by an application of (Dec) in $\langle \{x : f(\overline{s_n}) \triangleq f(\overline{t_n})\} \mid \emptyset \mid id\rangle$, it follows that $x_j \neq x_k$.By Lemma 2.2 it follows that

$$\mathtt{vran}(\sigma_j) \subseteq \mathcal{V}(x_j\sigma_j) \cup \{x_j\}, \quad \mathtt{vran}(\sigma_k) \subseteq \mathcal{V}(x_k\sigma_k) \cup \{x_k\}.$$

Hence, $\mathtt{vran}(\sigma_j) \cap \mathtt{vran}(\sigma_k) = \emptyset$. It follows that $(x_k\sigma_k)\sigma_j \equiv u_k\sigma_j = u_k$ and $(x_j\sigma_j)\sigma_k \equiv u_j\sigma_k = u_j$.

Then, the result follows as recursively applies each derivation R_i in (3), in they respective constraint $\{x_i : s_i \triangleq t_i\}$, for all $i = 1, \dots, n$ as is shown bellow:

$$\begin{aligned}
\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle \{x_1 : s_1 \triangleq t_1\} \cup P_1 \mid \emptyset \mid \sigma_0\rangle \\
&\overset{*}{\Longrightarrow}_{(R_1)} \langle \{x_2 : s_2 \triangleq t_2\} \cup P_2 \cup C_1 \mid S_1 \mid \sigma_0\sigma_1\rangle \\
&\overset{*}{\Longrightarrow}_{(R_2)} \\
&\Longrightarrow \dots \\
&\overset{*}{\Longrightarrow}_{(R_i)} \langle P_i \cup C_1 \cup \dots \cup C_i \mid S_1 \cup \dots S_i \mid \sigma_0 \dots \sigma_i\rangle \\
&\overset{*}{\Longrightarrow} \dots \\
&\overset{*}{\Longrightarrow}_{(R_{n-1})} \langle \{x_n : s_n \triangleq t_n\} \cup C_1 \cup \dots \cup C_{n-1} \mid S_1 \cup \dots \cup S_{n-1} \mid \sigma_0 \dots \sigma_{n-1}\rangle \\
&\overset{*}{\Longrightarrow}_{(R_n)} \langle \bigcup_{i=1}^{n} C_i \mid \bigcup_{i=1}^{n} S_i \mid \sigma_0 \dots \sigma_n \rangle
\end{aligned}$$

where $P_i = \{x_1 : s_1 \triangleq t_1, \dots, x_n : s_n \triangleq t_n\} / (\bigcup_{i=1}^{n} \{x_i : s_i \triangleq t_i\})$. Then, the result follows as taking $P = \bigcup_{i=1}^{n} C_i$, $S = \bigcup_{i=1}^{n} S_i$ and $\sigma = \sigma_0 \dots \sigma_n$.

□

Let $\mathcal{A}\langle s,t\rangle$ be an AUP. It is easy to see that each step n of $\mathtt{AUnif}_\emptyset$ compute a substitution σ_n that is the composition of σ_{n-1}, the substitution computed in the previous $(n-1)$-step, with another substitution δ, i.e, $\sigma_n = \sigma_{n-1}\delta$. By Lemma 2.1 and Lemma 2.2 it follows that this composition of substitutions is such that $x\sigma_{n-1} \leq x\sigma_n$. Then, it is natural to conclude that the normal form of $\mathtt{AUnif}_\emptyset$ ($\langle \emptyset \mid S \mid \sigma\rangle$), must return a least general generalizer of s and t. The next theorem will formalize this intuitive idea.

Theorem 2.3 (Correctness of $\mathtt{AUnif}_\emptyset$ [1].)**.** Let $s,t \in T(\mathcal{X}, \Sigma_\emptyset)$. Then, $u \in \mathtt{lgg}(s,t)$ if, and only if, there exists a derivation $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \Longrightarrow^*_{\mathtt{AUnif}_\emptyset} \langle \emptyset \mid S \mid \sigma\rangle$ and $u \equiv x\sigma$.

Proof. Consider the normalizing derivation

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \Longrightarrow^*_{\texttt{AUnif}_\emptyset} \langle\emptyset \mid S \mid \sigma\rangle$$

Then Lemma 2.6 yields that $x\sigma \in \texttt{gen}(s,t)$ up to variable renaming. Suppose by contradiction that $x\sigma$ is not a least general generalizer of s and t, i.e, $x\sigma \notin \texttt{lgg}(s,t)$. Then, $x\sigma$ is more general than a term that is a least general generalizer of s and t, i.e, there exists a substitution δ, that is not a variable renaming such that $(x\sigma)\delta' \in \texttt{lgg}(s,t)$.

Lemma 2.2 yields that $\texttt{vran}(\sigma) \subseteq \mathcal{V}(x\sigma)$. Notice that if there exists a variable $w \in \texttt{dom}(\delta)$ such that $w \notin \mathcal{V}(x\sigma)$, $w\delta$ actually does not matter because we are applying δ in $x\sigma$. Therefore, suppose that $\texttt{dom}(\delta) \subseteq \mathcal{V}(x\sigma)$. Since δ is not a variable renaming, there are two possibilities on how δ acts in $x\sigma$.

Case 1: There are distinct variables $y, y' \in \mathcal{V}(x\sigma)$ and a variable z such that $y\delta = y'\delta = z$:

Suppose $x\sigma|_p = y$, $x\sigma|_{p'} = y'$ then there are constraints in S such that $y, y' \in \texttt{Index}(S)$ and Lemma 2.4 yields that p and p' are two conflicting positions. Since

$$(x\sigma)\delta|_p = z = (x\sigma)\delta|_{p'}$$

and $(x\sigma)\delta \in \texttt{lgg}(s,t)$ it follows that $s|_p = s|_{p'}$ and $t|_p = t|_{p'}$. Then the constraints in S with indexes y, y' represents the same Anti-Unification Problem $\mathcal{A}\langle s|_p, t|_p\rangle$. Since $\{y : s|_p \triangleq t|_p, y' : s|_{p'} \triangleq t|_{p'}\} \in S$ it means that each one of these constrains that are a step of $\texttt{AUnif}_\emptyset$ were (Sol) applied in it. What is a contradiction of definition of (Sol) rule in Figure 2.2, which yields that $\texttt{AUnif}_\emptyset$ was applied erroneously because (Rec) should have being applied in one of these constraints instead of (Sol). Therefore, there are no distinct variables $y, y' \in \mathcal{V}(x\sigma)$ such that $y\delta = y'\delta$.

Case 2: There is a variable $y \in \mathcal{V}(x\sigma)$ and a non variable term v such that $y\delta = v$:

Then, there exist a position p such that $x\sigma|_p = y$ and constraint C in S with $\texttt{Index}(C) = y$, therefore Lemma 2.4 implies that $(s|_p, t|_p)$ is a conflict pair of s and t. Since $(x\sigma)\delta \in \texttt{lgg}(s,t)$ it follows that $(x\sigma)\delta|_p = v$ which implies that

$$\texttt{root}(s|_p) = \texttt{root}(t|_p) = \texttt{root}(v)$$

what is a contradiction because the Definition 2.4 says that every conflict pair (s,t) must be such that $\texttt{root}(s) \neq \texttt{root}(t)$.

□

Now, we will show that given $\mathcal{A}\langle s,t\rangle$ a Anti-Unification Problem, the final configuration $\langle\emptyset \mid S \mid \sigma\rangle$ gives a map to how send the least general generalizer $x\sigma$ into s and t. First, it is necessary to do a remark.

Remark 2.4. Notice that Lemma 2.3 says that for every conflict pair of subterms (u,v) there exist a reduction such that $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$. In [1] there is a claim that says that this result follows by an sequence of applications of (Dec). It is easy to see that this claim is true, since the only rule in Figure 2.2 that create constraints is (Dec) and for each conflict pair of subterms (u,v) of s and t there exists a position $p \in \texttt{pos}(s) \cup \texttt{pos}(t)$ such that for every $p' < p$, $\texttt{root}(s|_{p'}) = \texttt{root}(t|_{p'})$. Then, there exists a derivation such that

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \cup \{y : u \triangleq v\} \mid \emptyset \mid \sigma\rangle.$$

Lemma 2.8. Let $\mathcal{A}_\emptyset\langle s,t\rangle$ be an AUP. If there exists a derivation of the form

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \Longrightarrow^*_{\texttt{AUnif}_\emptyset} \langle\emptyset \mid S \mid \sigma\rangle$$

with $S = \{y_1 : u_1 \triangleq v_1; y_2 : u_2 \triangleq v_2, \ldots, y_n : u_n \triangleq v_n\}$, then the pair of substitution (θ_1, θ_2) given for$\theta_1 = \{y_1 \mapsto u_1, \ldots, y_n \mapsto u_n\}$ and $\theta_2 = \{y_1 \mapsto v_1, \ldots, y_n \mapsto v_n\}$ is such that

$$(x\sigma)\theta_1 = s \text{ and } (x\sigma)\theta_2 = t.$$

Proof. The proof proceeds by analyse of the conflict pairs of s and t.

Case 1: s and t do not have any conflict pair:

Then by Lemma 2.4, $S = \emptyset$ and it follows that $\overline{\theta} = (\theta_1, \theta_2) = (id, id)$.

Notice that $s = t$. In fact, if there exists a $p \in \texttt{pos}(s)$ such that $s|_p \neq t|_p$ then would exist a conflict position q of s and t such that $q \leq p$, which contradicts the hypotheses of this case.

Thus, by Theorem 2.3 it follows that $x\sigma = s = t$. Therefore, $(x\sigma)\overline{\theta} = (s,t)$.

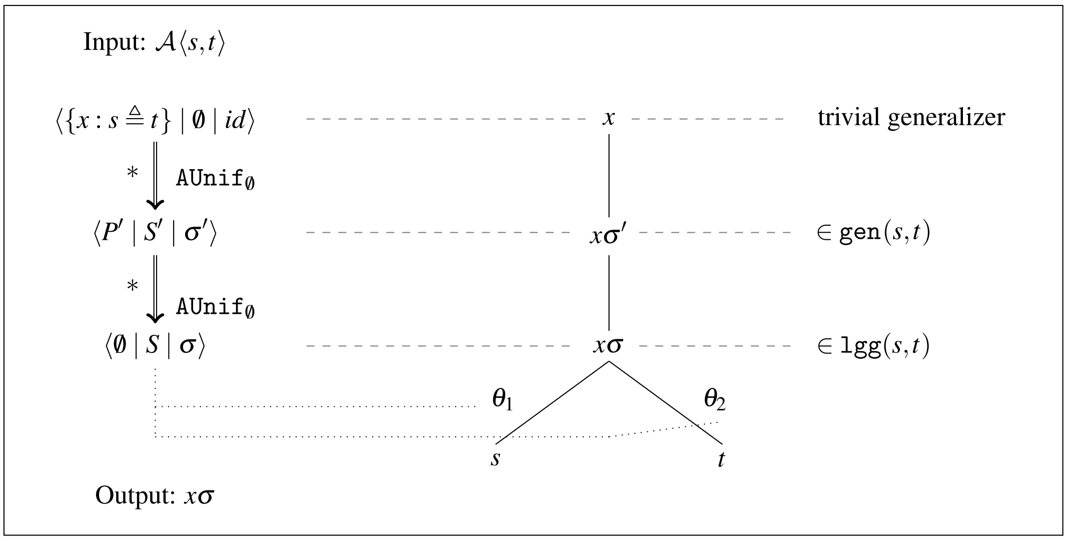
Case 2: s and t have conflict positions.

This case is analysed in Appendix A, Lemma A.8.

□

Corollary 2.1. Given a derivation $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \Longrightarrow^*_{\texttt{AUnif}_\emptyset} \langle\emptyset \mid S \mid \sigma\rangle$ then $x\sigma \in \texttt{lgg}(s,t)$ and exists a pair of substitutions (θ_1, θ_2), give by S, such that $(x\sigma)\theta_1 \equiv s$ and $(x\sigma)\theta_2 \equiv t$.

Proof. Straightforward from Lemma 2.8 and Theorem 2.3. □


Input: $\mathcal{A}\langle s,t\rangle$
$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$
x
trivial generalizer
$*$
$\mathtt{AUnif}_{\emptyset}$
$\langle P' \mid S' \mid \sigma'\rangle$
$x\sigma'$
$\in \mathtt{gen}(s,t)$
$*$
$\mathtt{AUnif}_{\emptyset}$
$\langle\emptyset \mid S \mid \sigma\rangle$
$x\sigma$
$\in \mathtt{lgg}(s,t)$
θ_1
θ_2
s
t
Output: $x\sigma$

Given an Anti-Unification Problem $\mathcal{A}\langle s,t\rangle$, $\mathtt{AUnif}_{\emptyset}$ output a term r that is the least general generalizer of s and t and also output a pair of substitutions $\overline{\theta}$ such that $\overline{\theta}r = (s,t)$, as was shown in the figure in above. Therefore, $\mathtt{AUnif}_{\emptyset}$ solve the Anti-Unification Problem.

Moreover, Theorem 2.3 says that u is a solution of the Anti-Unification Problem $\mathcal{A}\langle s,t\rangle$ iff and only if there is a normal form $\langle\emptyset \mid S \mid \sigma\rangle$ given by $\mathtt{AUnif}_{\emptyset}$ such that the term $x\sigma$ is equal to u except by variable renaming. Since Theorem 2.2 yields that $\mathtt{AUnif}_{\emptyset}$ is confluent it follows that the solution of $\mathcal{A}\langle s,t\rangle$ is unique except by variable renaming.

Chapter 3

Commutative Anti-Unification Problem

In this chapter we will show that in the case of Commutative Anti-Unification (for short AUP$_C$), if we use $\texttt{AUnif}_\emptyset$ it may output a generalizer that is not a least general one. This happens because $\texttt{AUnif}_\emptyset$ rules does not treat the properties of this theory properly. Then, it is necessary to adapt the previous inference rules and build a new algorithm called $\texttt{AUnif}_C$ that is able to handle commutativity. This adaptation takes place as adding a new rule *(C-Dec)* called Commutative-Decompose and keeping the syntactic inference rules of $\texttt{AUnif}_\emptyset$ with some restrictions as is shown in Figure 3.1. Then, we will study the Commutative Anti-Unification closely and verify the main properties of $\texttt{AUnif}_C$: termination (Theorem 3.1), soundness (3.2) and completeness (Theorem 3.3).

3.1 Motivating Examples

In this section we will illustrate the Anti-Unification Problem when commutative function symbols are considered, i.e, the $\mathcal{A}_C\langle s,t\rangle$ problem for $s,t \in T(\mathcal{X},\Sigma_{\emptyset\cup C})$, introduced in subsection 1.2.1. Also, we will show why $\texttt{AUnif}_\emptyset$ is not enough to solve $\mathcal{A}_C\langle s,t\rangle$. The purpose of this subsection is to make the reader more familiar with the problem.

Example 3.1 (Applying $\texttt{AUnif}_\emptyset$ to solve $\mathcal{A}_C\langle s,t\rangle$)**.** Let $g \in \Sigma_C$, then $s = g(a,b)$ and $t = g(b,c)$ are terms in $T(\mathcal{X},\Sigma_{\emptyset\cup C})$. If we try to solve the AUP$_C$ $\mathcal{A}_C\langle s,t\rangle$ using the $\texttt{AUnif}_\emptyset$ simplification rules, the following derivation is possible

$$\begin{array}{c}
\langle \{x : g(a,b) \triangleq g(b,c)\} \mid \emptyset \mid id\rangle \\
\text{\scriptsize\textit{(C-Dec)}}\Downarrow \\
\langle \{x_1 : a \triangleq b, x_2 : b \triangleq c\} \mid \emptyset \mid \underbrace{\{x \mapsto g(x_1,x_2)\}}_{\sigma}\rangle \\
\text{\scriptsize\textit{(C-Dec)}}\Downarrow \\
\langle \emptyset \mid \{x_1 : a \triangleq b, x_2 : b \triangleq c\} \mid \sigma\rangle.
\end{array}$$

Thus, $\texttt{AUnif}_\emptyset$ returns $x\sigma = g(x_1,x_2)$ as generalizer for s and t. It is easy to see that $g(x_1,x_2) \in \texttt{gen}_A(s,t)$, but note that it is not the least general generalizer modulo C of s and t. In fact, $g(b,x_2) \in \texttt{gen}_C(s,t)$ is also a generalizer modulo C of s and t, i.e, $g(b,x_2) \in \texttt{gen}_C(s,t)$. Moreover, it is less general modulo C than $g(x_1,x_2)$, i.e. $g(x_1,x_2) \leq_C g(b,x_2)$, as is it nicely illustrated in the figure below:

$$\begin{tikzcd}
 & x \arrow[d] \arrow[rr, dotted, dash] & & \text{Trivial solution} \\
 & \boxed{g(x_1,x_2)} \arrow[d] \arrow[rr, dotted, dash] & & \texttt{AUnif}_\emptyset\text{'s output} \\
 & g(b,x_2) \arrow[dl] \arrow[dr] \arrow[rr, dotted, dash] & & \texttt{lgg}_C(s,t) \\
g(a,b) & & g(b,c)) &
\end{tikzcd}$$

Then, the $\texttt{AUnif}_\emptyset$ still outputs generalizers, but not necessarily least general ones. To solve AUP$_C$, it is necessary to modify $\texttt{AUnif}_\emptyset$ algorithm by adding new rules and adapt some of its current rules. The next section will focus in this procedure.

Example 3.2 (The set $\texttt{lgg}_C(s,t)$ is not unitary)**.** Let $g \in \Sigma_C$, then $s = g(g(a,b),b)$ and $t = g(a,g(b,a))$ are terms in $T(\mathcal{X},\Sigma_{\emptyset\cup C})$. It follows that

$$r_1 = g(g(y_1,x_2),x_2) \in \texttt{gen}_C(s,t) \text{ and } r_2 = g(g(a,b),x_2) \in \texttt{gen}_C(s,t).$$

Notice that $r_2 \not\leq_C r_1$. In fact, if $r_2 \leq_C r_1$ then there would exist a substitution θ such that $r_2\theta =_C r_1$. However, $g(g(a,b)),x_2)\theta \neq_C g(g(y_1,x_2),x_2)$ for any substitution θ, because substitutions do not change constant function symbols.

Similarly, $r_1 \not\leq_C r_2$. In fact, if $r_1 \leq_C r_2$ then there would exist θ' such that $r_1\theta' =_C r_2$, i.e., $g(g(y_1,x_2),x_2)\theta' =_C g(g(a,b),x_2)$. Then, we would have

$$g(g(y_1,\theta',x_2\theta'),x_2\theta') =_C g(g(a,b),x_2),$$

thus this substitution θ' must send x_2 into a or b, but if θ' is a substitution such that

- $x_2\theta' = a$, then $\theta' r_1 = g(g(y_1,a),a) \neq_C r_2$, or
- $x_2\theta' = b$, then $\theta' r_1 = g(g(y_1,b,),b) \neq_C r_2$.

Then r_1 and r_2 are incomparable by $\equiv_C$. Hence, r_1 and r_2 are both maximal elements of $\texttt{gen}_C(s,t)$. In conclusion, the set of least general generalizations of s and t modulo commutativity is not unitary.

3.2 Solving the Commutative Anti-Unification Problem

In this section, we present the approach taken by Alpuente et. al. [1] for solving the Commutative Anti-Unification Problem (AUP$_C$). The proposal is to adapt the syntactic algorithm Anti-Unification `AUnif`$_\emptyset$ to handle the commutative property, obtaining a C-anti-unification algorithm called `AUnif`$_C$, which rules are shown in Figure 3.1 below.

(Dec) **: Decompose**
$(f : n \in \Sigma_\emptyset \cup \mathcal{X})$

$$\langle P \cup \{x : f(\overline{s_n}) \triangleq f(\overline{t_n})\} \mid S \mid \sigma' \rangle \Longrightarrow \left\langle P \cup \begin{Bmatrix} x_1 : s_1 \triangleq t_1, \\ \vdots \\ x_n : s_n \triangleq t_n \end{Bmatrix} \mid S \mid \sigma\{x \mapsto f(\overline{x_n})\} \right\rangle$$

where $x_1, \ldots, x_n$ are fresh variables.

(Sol)**: Solve:**
If $\texttt{root}(s) \neq \texttt{root}(t)$ and there is no constraint $\{y : s \triangleq t\} \in S$

$$\langle P \cup \{x : s \triangleq t\} \mid S \mid \sigma \rangle \Longrightarrow \langle P \mid S \cup \{x : s \triangleq t\} \mid \sigma \rangle$$

(Rec)**: Recover:**
If $\texttt{root}(s) \neq \texttt{root}(t)$

$$\langle P \cup \{x : s \triangleq t\} \mid S \cup \{y : s \triangleq t\} \mid \sigma \rangle \Longrightarrow \langle P \mid S \cup \{y : s \triangleq t\} \mid \sigma\{x \mapsto y\} \rangle$$

(C-Dec) **: Commutative Decompose**
$(g \in \Sigma_C)$

$$\begin{tikzcd}
 & \langle \{x : g(s_1, s_2) \triangleq g(t_1, t_2)\} \mid S \mid \sigma \rangle \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\
\left\langle \begin{Bmatrix} x_1 : s_1 \triangleq t_1 \\ x_2 : s_2 \triangleq t_2 \end{Bmatrix} \mid S \mid \sigma\{x \mapsto g(x_1, x_2)\} \right\rangle & & \left\langle \begin{Bmatrix} x_1 : s_1 \triangleq t_2 \\ x_2 : s_2 \triangleq t_1 \end{Bmatrix} \mid S \mid \sigma\{x \mapsto g(x_1, x_2)\} \right\rangle
\end{tikzcd}$$

where x_1, x_2 are fresh variables.

Fig. 3.1 `AUnif`$_C$ Simplification Rules.

The rule (Dec) is the same as $\texttt{AUnif}_{\emptyset}$ just now restricted to free function symbols $f \in \Sigma_{\emptyset}$ and variables. The opaque rules (in gray) are those that have not changed. In the following, we described the rule Commutative-Decompose.

(C-Dec) In the case $\{x : g(s_1,s_2) \triangleq g(t_1,t_2)\}$ is a constraint, where $g \in \Sigma_C$. We can commute the arguments of g in two different ways; thus, this rule branches.

For instance, we can compare s_1 with t_1 and s_2 with t_2, obtaining two new constraints $\{x_1 : s_1 \triangleq t_2, x_2 : s_2 \triangleq t_2\}$. But g is a commutative function symbol, thus we could have chosen commute: comparing s_1 with t_2 and s_2 with t_1, obtaining two new constraints $\{x_1 : s_1 \triangleq t_2, x_2 : s_2 \triangleq t_1\}$. For each choice, we delete the constraint $\{x : g(s_1,s_2) \triangleq g(t_1,t_2)\}$ and replace it with the two different constraints obtained; both cases create a new substitution $\sigma = \{x \mapsto g(x_1,x_2)\}$, where x_1,x_2 are the corresponding index variables of each new constraint created.

For $s = g(s_1,s_2)$ and $t = g(t_1,t_2)$, notice that x is a generalizer of s and t, but $x\sigma = g(x_1,x_2)$ is a C-generalizer of s and t that preserves more structure of both terms than x, i.e., $x < g(x_1,x_2)$. Therefore, the application of *(C-Dec)* rule gives a more specific C-generalizer of s and t.

Example 3.3 (How to apply $\texttt{AUnif}_C$). Let $s = g(a,b)$ and $t = g(b,c)$ be $T(\mathcal{X},\Sigma_{\emptyset\cup C})$ terms where $g \in \Sigma_C$. Then, $\texttt{AUnif}_C(s,t)$ derivation tree is of the form given below:

$$\begin{tikzcd}
 & \langle\{x : g(a,b) \triangleq g(b,c)\} \mid \emptyset \mid id\rangle \arrow[dl, Rightarrow, "\textit{(C-Dec)}"'] \arrow[dr, Rightarrow, "\textit{(C-Dec)}"] & \\
\langle\{x_1 : a \triangleq b, x_2 : b \triangleq c\} \mid \emptyset \mid \underbrace{\{x \mapsto g(x_1,x_2)\}}_{\sigma}\rangle \arrow[d, Rightarrow, "\texttt{AUnif}_C"', "*"] & & \langle\{x_1 : a \triangleq c, x_2 : b \triangleq b\} \mid \emptyset \mid \sigma\rangle \arrow[d, Rightarrow, "\texttt{AUnif}_C"', "*"] \\
\langle\emptyset \mid \{x_1 : a \triangleq b, x_2 : b \triangleq c\} \mid \sigma\rangle & & \langle\emptyset \mid \{x_1 : a \triangleq b\} \mid \underbrace{\sigma\{x_2 \mapsto b\}}_{\sigma_1}\rangle
\end{tikzcd}$$

Therefore $\texttt{AUnif}_C$ output $x\sigma = g(x_1,x_2)$ and $x\sigma_1 = g(x_1,b)$ and $g(x_1,x_2) <_C g(x_1,b)$, hence only $g(x_1,b) \in \texttt{lgg}_C(s,t)$.

The next example shows that $\texttt{AUnif}_C$ is not confluent, differently of the syntactic anti-unification algorithm $\texttt{AUnif}_{\emptyset}$. However, $\texttt{lgg}_C(s,t)$ is not unitary, $\texttt{AUnif}_C$ being non-confluent is convenient since the algorithm $\texttt{AUnif}_C$ when applied to a problem $\mathcal{A}_C\langle s,t\rangle$ returns normal forms that will generate a complete set of generalizers of s and t; consequently, a complete set of least general generalizers for s and t modulo C

Example 3.4 (AUnif$_C$ is not confluent)**.** Given $s = g(g(a,b),b)$ and $t = g(a,g(b,a))$, with $g \in \Sigma_C$, it follows that:

$$\begin{tikzcd}
{} & \langle\{x : g(g(a,b),b) \triangleq g(a,g(b,a))\} \mid \emptyset \mid id\rangle \arrow[dl, Rightarrow, "(C{-}Dec)"'] \arrow[dr, Rightarrow, "(C{-}Dec)"] & {} \\
\langle\left\{\begin{matrix} x_1 : g(a,b) \triangleq a, \\ x_2 : b \triangleq g(b,a) \end{matrix}\right\} \mid \emptyset \underbrace{\{x \mapsto g(x_1,x_2)\}}_{\sigma_1}\rangle \arrow[d, Rightarrow, "\mathtt{AUnif}_C"'] & {} & \langle\left\{\begin{matrix} x_1 : g(a,b) \triangleq g(b,a), \\ x_2 : b \triangleq a \end{matrix}\right\} \mid \emptyset \underbrace{\{x \mapsto g(x_1,x_2)\}}_{\sigma_1}\rangle \arrow[d, Rightarrow, "(Sol)"'] \\
\langle\emptyset \mid \left\{\begin{matrix} x_1 : g(a,b) \triangleq a, \\ x_2 : b \triangleq g(b,a) \end{matrix}\right\} \mid \sigma_1\rangle & {} & \langle\{x_1 : g(a,b) \triangleq g(b,a)\} \mid \underbrace{\{x_2 : b \triangleq a\}}_{S} \mid \sigma_1\rangle \arrow[dl, Rightarrow, "(C\text{-}Dec)"'] \arrow[d, Rightarrow, "(C\text{-}Dec)"] \\
{} & \langle\left\{\begin{matrix} y_1 : a \triangleq b, \\ y_2 : b \triangleq a \end{matrix}\right\} \mid S \mid \underbrace{\sigma_1\{x_1 \mapsto g(y_1,y_2)\}}_{\sigma'_2}\rangle \arrow[d, Rightarrow, "(Sol)"'] & \langle\left\{\begin{matrix} y_1 : a \triangleq a, \\ y_2 : b \triangleq b \end{matrix}\right\} \mid S \mid \sigma_1\rangle \arrow[d, Rightarrow, "(Dec)"] \\
{} & \langle\{y_2 : b \triangleq a\} \mid S \cup \{y_1 : a \triangleq b\} \mid \sigma'_2\rangle \arrow[d, Rightarrow, "(Rec)"'] & \langle\{y_2 : b \triangleq b\} \mid S \mid \underbrace{\sigma_1\{y_1 \mapsto a\}}_{\sigma'_3}\rangle \arrow[d, Rightarrow, "(Dec)"'] \\
{} & \langle\emptyset \mid S \cup \{y_1 : a \triangleq b\} \mid \underbrace{\sigma'_2\{y_2 \mapsto x_2\}}_{\sigma_2}\rangle & \langle\emptyset \mid S \mid \underbrace{\sigma'_3\{y_2 \mapsto b\}}_{\sigma_3}\rangle
\end{tikzcd}$$

Therefore AUnif$_C$ output $x\sigma_1 = g(x_1,x_2)$, $x\sigma_2 = g(g(y_1,x_2),x_2)$ and $x\sigma_3 = g(g(a,b),x_2)$. By definition, $x\sigma_1 <_C x\sigma_2$ and $x\sigma_1 <_C x\sigma_3$. Moreover, $x\sigma_2$ and $x\sigma_3$ are incomparable by the instantiation ordering, as it was shown in Example 3.2. Therefore, $x\sigma_2, x\sigma_3 \in \mathtt{lgg}(s,t)$.

Furthermore, the set of unsolved constraints of each final configuration gives a respective mapping from $x\sigma_1$, $x\sigma_2$ and $x\sigma_3$ to s and t. More precisely,

1. The final configuration $\langle\emptyset \mid \{x_1 : g(a,b) \triangleq a, x_2 : b \triangleq g(b,a)\} \mid \sigma_1\rangle$ is such that the left side of the solved constraints gives $\lambda_1 = \{x_1 \mapsto g(a,b), x_2 \mapsto b\}$ and the right side gives $\lambda_2 = \{x_1 \mapsto a, x_2 \mapsto g(b,a)\}$. Notice that $(x\sigma_1)(\lambda_1,\lambda_2) = g(x_1,x_2)(\lambda_1,\lambda_2) =_C (s,t)$.

2. The final configuration $\langle\emptyset \mid \{x_2 : b \triangleq a, y_1 : a \triangleq b\} \mid \sigma_2\rangle$ is such that the left side of the solved constraints gives the substitution $\theta_1 = \{x_2 \mapsto b, y_1 \mapsto a\}$, while the right side gives $\theta_2 = \{x_2 \mapsto a, y_1 \mapsto b\}$. Notice that $(x\sigma_2)(\theta_1,\theta_2) = g(g(y_1,x_2),x_2)(\theta_1,\theta_2) =_C (s,t)$.

3. The final configuration $\langle\emptyset \mid \{x_2 : b \triangleq a\} \mid \sigma_3\rangle$ is such that the left side of the solved constraints gives $\theta'_1 = \{x_2 \mapsto b\}$ and the right side of gives $\theta'_2 = \{x_2 \mapsto a\}$. Notice that $(x\sigma)(\theta'_1,\theta'_2) = g(g(a,b),x_2)(\theta'_1\theta'_2) =_C (s,t)$.

The figure bellow illustrate the mapping presented above for the least general generalizers, modulo C, of s and t.

$$\begin{tikzcd}
g(g(y_1,x_2),x_2) \arrow[rrr, dash, dotted, "\not\equiv_C" description] \arrow[dr, "\theta_1"'] \arrow[drr, "\theta_2"] & & & g(g(a,b),x_2) \arrow[dll, "\theta'_1"'] \arrow[dl, "\theta'_2"] \\
 & g(g(a,b),b) & g(a,g(b,a)) &
\end{tikzcd}$$

Commutative pair and commutative conflict pair of subterms.

The next definitions use the notions of prefix and index of a position, as given in Definition 1.6. Let $\mathcal{A}_C\langle s,t\rangle$ be a AUP$_C$, the definition of commutative pair of subterms that will be presented in the following is important to identify common structures in the term threes of s and t.

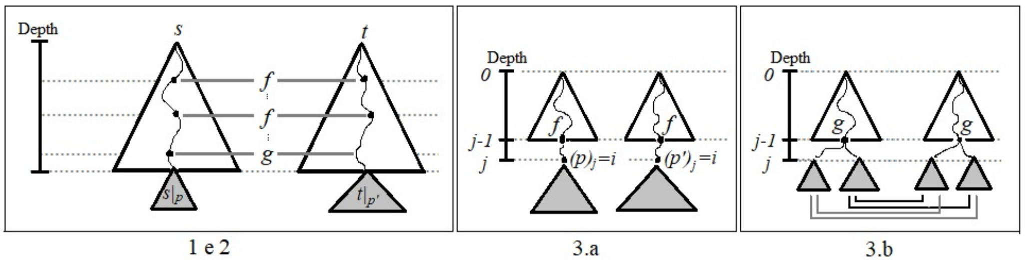


Fig. 3.2 Commutative pair of subterms.

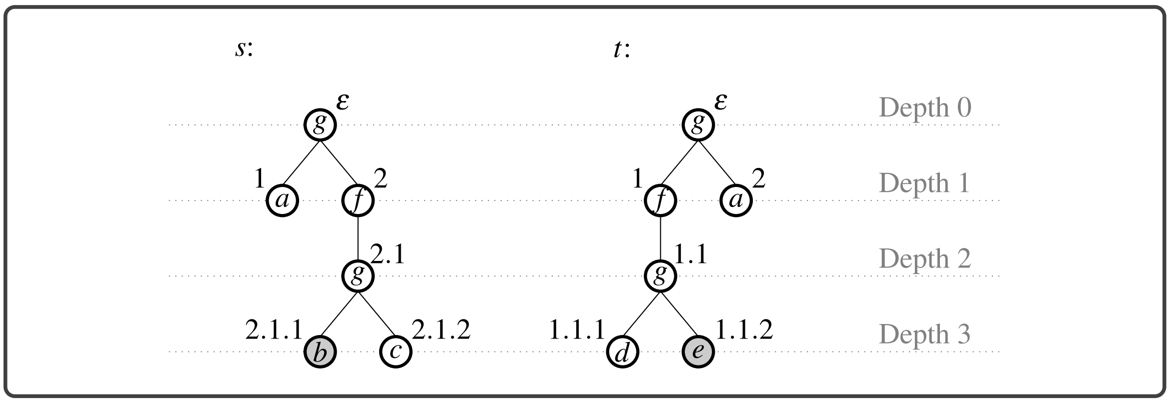
Definition 3.1 (Commutative pair of subterms)**.** Given terms s, $t \in T(\mathcal{X},\Sigma_{\emptyset\cup C})$, the pair (u,v) of terms is called a **commutative pair of subterms** of s and t if and only if there are positions $p \in \texttt{pos}(s)$ and $p' \in \texttt{pos}(t)$ such that the following hold;

1. $s|_p = u, t|_{p'} = v$ and $\texttt{depth}(p) = \texttt{depth}(p')$;
2. for each $0 \leq i < \texttt{depth}(p)$, s and t have the same i prefix $\texttt{root}(s|_{(p)^i}) = \texttt{root}(t|_{(p')^i})$; and
3. for each $0 < j \leq \texttt{depth}(p)$:
   - if $\texttt{root}(s|_{(p)^{j-1}}) = f \in \Sigma_\emptyset$ then $(p)_j = (p')_j$, i.e. if the root of the subterm of s at prefix position $(p)^{j-1}$ is a free function symbol $f \in \Sigma$, then p and p' have the same index at depth j;

- $\texttt{root}(s|_{(p)^{j-1}}) = g \in \Sigma_C$ then $(p)_j = (p')_j$ or $(p)_j = ((p)_j \ \texttt{mod}\ 2)+1$, i.e. if the the root of the subterm at the prefix position $(p)^{j-1}$ is a commutative function symbol $g \in \Sigma_C$, then p and p' have the same index at depth j or p at depth j is equal to the index of p' at depth of j modulo 2 plus 1.

This conditions are nicely illustrated in Figure 3.2.

Example 3.5. Let $f : 1 \in \Sigma_\emptyset$ and $g \in \Sigma_C$. Then $s = g(a, f(g(b,c)))$ and $t = g(f(g(d,e)),a)$ are terms in $T(\mathcal{X}, \Sigma_{\emptyset\cup C})$. The pair (b,e) is a commutative pair or subterms of s and t, as is illustrated bellow.



In fact, the positions $2.1.1 \in \texttt{pos}(s)$ and $1.1.2 \in \texttt{pos}(t)$ are such that

1. $s|_{2.1.1} = b$, $t|_{1.1.2} = e$ and $\texttt{depth}(2.1.1) = \texttt{depth}(1.1.2) = 3$. Therefore, the first condition of Definition 3.1 holds.

2. For each $i = 0,1,2$ it follows, respectively, that

$$\texttt{root}(s|_{(2.1.1)^0}) = \texttt{root}(s) = g = \texttt{root}(t) = \texttt{root}(t|_{(1.1.2)^0}),$$
$$\texttt{root}(s|_{(2.1.1)^1}) = \texttt{root}(f(g(b,c))) = f = \texttt{root}(f(g(d,e))) = \texttt{root}(t|_{(1.1.2)^1}),$$
$$\texttt{root}(s|_{(2.1.1)^2}) = \texttt{root}(g(b,c)) = g = \texttt{root}(g(d,e)) = \texttt{root}(t|_{(1.1.2)^2}).$$

   Therefore, the second condition of Definition 3.1 holds.

3. For each $j = 1,2,3$ the root symbol each of $s|_{(2.1.1)^{j-1}}$ are, respectively,

$$\texttt{root}(s|_{(2.1.1)^{1-1}}) = g, \quad \texttt{root}(s|_{(2.1.1)^{2-1}}) = f, \quad \texttt{root}(s|_{(2.1.1)^{3-1}}) = g.$$

   The index $1,2,3$ of the positions $p = 2.1.1$ and $p' = 1.1.2$ are such that

$$(2.1.1)_1 = 2 = (1 \ \texttt{mod}\ 2)+1 = ((1.1.2)_1 \ \texttt{mod}\ 2)+1,$$

$$(2.1.1)_2 = 1 = (1.1.2)_2,$$
$$(2.1.1)_3 = 1 = (2 \texttt{ mod } 2)) + 1 = ((1.1.2)_3 \texttt{ mod } 2) + 1,$$

Therefore, the Definition 3.1 is satisfied for each index. Thus, (b,e) is a commutative pair of subterms of s and t.

Definition 3.2. Given terms $s,t \in T(\mathcal{X},\Sigma_{\emptyset \cup C})$, the pair (u,v) is called a **commutative conflict pair** if, and only if, (u,v) is one commutative pair of subterms such that $\texttt{root}(u) \neq \texttt{root}(t)$.

Example 3.6 (Continuation of Example 3.5)**.** Given $f : 1 \in \Sigma_\emptyset, g \in \Sigma_C$, and taking terms $s = g(a, f(g(b,c)))$ and $t = g(f(g(d,e)),a)$. By Example 3.5, (b,e) is a commutative pair of subterms of s and t. Notice that $\texttt{root}(b) \neq \texttt{root}(e)$, then (b,e) is one commutative conflict pair of s and t.

3.3 Properties

In this section, we present the main properties of $\texttt{AUnif}_C$, such as termination (Theorem 3.1), soundness (Theorem 3.2) and completeness (Theorem 3.3). These results were initially proven in [1], here we present more detailed versions of their proofs.

Theorem 3.1 (Termination of $\texttt{AUnif}_C$ [1])**.** Let $\mathcal{A}_C\langle s,t\rangle$ be a AUP$_C$. Every derivation of $\texttt{AUnif}_C$ starting from $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ terminates in a final configuration of the form $\langle\emptyset \mid S \mid \sigma\rangle$.

Proof. To deal with the branches from the application of *(C-Dec)* rule we will assign a measure to each configuration in $\texttt{AUnif}_C$, denoted as follows: For a reduction step where the *(C-Dec)* rule was applied, we have the following shape:

$$\begin{tikzcd}
 & C = \langle P \mid S \mid \sigma\rangle \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\
C_1 = \langle P_1 \mid S_1 \mid \sigma_1\rangle & & \langle P_2 \mid S_2 \mid \sigma_2\rangle = C_2.
\end{tikzcd}$$

And for the others rules, the shape is $C = \langle P \mid S \mid \sigma\rangle \Longrightarrow_{(R)} \langle P_1 \mid S_1 \mid \sigma_1\rangle = C_1$.

To represent the branches obtained from *(C-Dec)* application, we will consider a more general form of configuration: $\{\{C\}\} \Longrightarrow_{\textit{(C-Dec)}} \{\{C_1, C_2\}\}$. And for the others rules, simply $\{\{C\}\} \Longrightarrow_{(R)} \{\{C_1\}\}$. Where $\{\{C\}\}$ denotes the multiset containing C.

Now, to prove that $\texttt{AUnif}_C$ terminates we will show that there is no infinite sequence of derivations obtained by application of derivation rules of Figure 3.1. In fact, for each step $M_1 \Longrightarrow_{(Rec)} M_2$, where M_1 and M_2 denotes the multisets of configurations, it holds that $M_1 >_{mul} M_2$, where $>_{mul}$ is the multiset ordering induced by the ordering defined in Definition 2.5. The proof proceed by analysis each rule of Figure 3.1. The cases for (Dec), (Sol) and (Red) where analysed during the proof of Theorem 2.1, therefore to complete the proof it remains to analyse the *(C-Dec)* case.

***(C-Dec)*:** It follows that $C = \langle\{y : g(s_1,s_2) \triangleq g(t_1,t_2)\} \cup P' \mid S \mid \sigma\rangle$ and

$$C_1 = \langle\begin{Bmatrix} x_1 : s_1 \triangleq t_1 \\ x_2 : s_2 \triangleq t_2 \end{Bmatrix} \mid S \mid \sigma\{y \mapsto g(y_1,y_2)\}\rangle,$$

$$C_2 = \langle\begin{Bmatrix} x_1 : s_1 \triangleq t_2 \\ x_2 : s_2 \triangleq t_1 \end{Bmatrix} \mid S \mid \sigma\{y \mapsto g(y_1,y_2)\}\rangle.$$

Thus, the reduction of the extended configuration is $\{\{C\}\} \Longrightarrow_{(C\text{-}Dec)} \{\{C_1,C_2\}\}$. To verify that $\{\{C\}\} >_{mul} \{\{C_1,C_2\}\}$, we need to show that $m(C) > m(C_1)$ and $m(C) > m(C_2)$. In fact,

$$\begin{aligned} m(C) &= |P'| + |g(s_1,s_2)| + |g(t_1,t_2)| \\ &= |P'| + |s_1| + |s_2| + |t_1| + |t_2| + 2 \\ &> |P'| + |s_1| + |s_2| + |t_1| + |t_2| = m(C_i), \end{aligned}$$

for $i = 1,2$ and the result follows.

□

Lemma 3.1 (cf. Lemmas 12 and 13 in [1]). Let $\mathcal{A}_C\langle s,t\rangle$ be a AUP$_C$. If

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P \mid S \mid \sigma\rangle,$$

then

1. if $P = P' \cup \{y : u \triangleq v\}$, y does not appear in any constraint in P or S,
2. $\texttt{Index}(S \cup P) \subseteq \texttt{vran}(\sigma) \cup \{x\}$, and $\texttt{vran}(\sigma) \subseteq \mathcal{V}(x\sigma)$

Proof. The complete proof is in Appendix B, Lemmas B.1 and B.2. □

Let $\mathcal{A}_C\langle s,t\rangle$ be a AUP$_C$ The next lemma establishes the relation between the commutative pair of subterms of s and t and the derivations of $\texttt{AUnif}_C$.

Lemma 3.2 (Lemma 14 in [1])**.** Let $\mathcal{A}_C\langle s,t\rangle$ be an AUP$_C$. There is a sequence of the form $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle\{y : u \triangleq v\} \cup P \mid S \mid \sigma\rangle$ if, and only if, (u,v) is a commutative pair of subterms of s and t.

Proof. The complete proof is in Appendix B, Lemma B.3 $\square$

The following lemma establishes the relation between the commutative conflict pairs of subterms of s and t and the derivations of $\mathtt{AUnif}_C$

Lemma 3.3 (Lemma 15 in [1])**.** Let $\mathcal{A}_C\langle s,t\rangle$ be a AUP$_C$. There is a derivation of the form $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P \mid \{u \triangleq v\} \cup S \mid \sigma\rangle$ if, and only if, (u,v) is a commutative conflict pair of s and t.

Proof. The complet proof is in Appendix B, Lemma B.4. $\square$

The next lemma is used to prove the correctness of $\mathtt{AUnif}_C$. Given $\mathcal{A}_C\langle s,t\rangle$ an AUP$_C$, it establishes the relation between the positions of s and t and the positions of each term $x\sigma$ given by a configuration $\langle P \mid S \mid \sigma\rangle$ obtained from the applications of $\mathtt{AUnif}_C$ rules in a derivation starting from a configuration $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$.

Lemma 3.4. Let $\mathcal{A}_C\langle s,t\rangle$ be an AUP$_C$. If there exists a derivation of the form

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle\{y : u \triangleq v\} \cup P \mid S \mid \sigma\rangle$$

then there exist positions $p \in \mathtt{pos}(s)$, and $p' \in \mathtt{pos}(t)$ such that

$$(x\sigma)|_p = y, \quad s|_p = (x\sigma)|_p\{y \mapsto u\} = u \quad \text{and} \quad t|_{p'} = (x\sigma)|_p\{y \mapsto v\} = v,$$

Proof. The complete proof can be found in Appendix B, Lemma B.5 $\square$

Theorem 3.2 (Soundness of $\mathtt{AUnif}_C$ [1])**.** Let $\mathcal{A}_C\langle s,t\rangle$ be an AUP$_C$. If there exists a derivation such that $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P \mid S \mid \sigma\rangle$, then $x\sigma \in \mathtt{gen}_C(s,t)$.

Proof. By induction on the length n of the reduction $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P \mid S \mid \sigma\rangle$.

Base Case: If $n = 0$, then $\langle P \mid S \mid \sigma\rangle = \langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ and $x\,id = x \in \mathtt{gen}_C(s,t)$ and x is the trivial generalizer of s and t.

Inductive Step: Suppose the result holds for derivations of length $n-1$. We will show that the result follows for derivations of length n, i.e, derivation of the form,

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P' \mid S' \mid \sigma'\rangle \Longrightarrow_{(R)} \langle P \mid S \mid \sigma\rangle$$

That is, $x\sigma \in \mathtt{gen}_C(s,t)$, where that $\sigma = \sigma'\delta$, for some δ.

The proof proceeds by analysis the rule (R) applied in the least step. The cases in which (R) is (Dec), (Sol) or (Rec) the prove were already verified in the prove of Lemma 2.6. It remains to verify the case which (R) is *(C-Dec)*. There is,

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P' \mid S' \mid \sigma'\rangle \Longrightarrow_{\textit{(C-Dec)}} \langle P \mid S \mid \sigma\rangle$$

The analysis depends on how *(C-Dec)* was applied in the last step.

1. Suppose the reduction is of the form

$$\begin{aligned}\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\overset{n-1}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle \{y : g(u_1,u_2) \triangleq g(v_1,v_2)\} \cup P' \mid S' \mid \sigma'\rangle \\ &\Longrightarrow_{\textit{(C-Dec)}} \langle \{y_1 : u_1 \triangleq v_2, y_2 : u_2 \triangleq v_1\} \mid S' \mid \underbrace{\sigma'\delta}_{\sigma}\rangle.\end{aligned}$$

   where $\delta = \{y \mapsto g(y_1,y_2)\}$, with y_1, y_2 as fresh variables.

   By Lemma 3.4 in the $n-1$ step of the reduction it follows that there are positions $p \in \mathtt{pos}(s)$ and $p' \in \mathtt{pos}(t)$ such that $(x\sigma')|_p = y$, $s|_p = (x\sigma')|_p\{y \mapsto g(u_1,u_2)\}$ and $t|_{p'} = (x\sigma')\{y \mapsto g(v_1,v_2)\}$. Furthermore, Lemma 3.2 yields that $(g(u_1,u_2), g(v_1,v_2))$ is a commutative pair of subterms. Then,

$$(x\sigma) = (x\sigma')\delta = g(y_1,y_2).$$

   The induction hypothesis yields that $x\sigma' \in \mathtt{gen}_C(s,t)$. Then, there exists a pair of substitutions $\overline{\theta} = (\theta_1, \theta_2)$ such that $(x\sigma')\overline{\theta} = ((x\sigma')\theta_1, (x\sigma')\theta_2) \equiv_C (s,t)$.

   In particular, $y\theta_1' = g(u_1,v_1)$. Then, taking θ_1' as θ_1 with domain restricted to $(\mathcal{X} - \{y\})$, it follows that the substitution θ can be unfolded as $\theta_1 = \{y \mapsto g(u_1,u_2)\}\theta_1'$. Thus,

$$\begin{aligned}(x\sigma')\theta_1 &= (x\sigma')\{y \mapsto g(u_1,u_2)\}\theta_1' \\ &= (x\sigma')\{y \mapsto g(y_1,y_2)\}\{y_1 \mapsto u_1, y_2 \mapsto u_2\}\theta_1' \\ &= (x\sigma)\delta\{y_1 \mapsto u_1, y_2 \mapsto u_2\}\theta_1',\end{aligned}$$

   similarly, $(x\sigma)\theta_2 \equiv_C x\sigma\delta\{y_1 \mapsto v_1, y_2 \mapsto v_2\}\theta_2'$.

   Taking the pair of substitutions $\overline{\tau} = (\tau_1, \tau_2)$ given as follows:

$$\tau_1 = \left\{\begin{matrix} x \mapsto x\theta_1', \text{ if } x \in \mathtt{dom}(\theta_1'), \\ y_1 \mapsto u_1, \\ y_2 \mapsto u_2 \end{matrix}\right\}, \quad \tau_2 = \left\{\begin{matrix} x \mapsto x\theta_2', \text{ if } x \in \mathtt{dom}(\theta_2'), \\ y_1 \mapsto v_2, \\ y_2 \mapsto v_1 \end{matrix}\right\}$$

Notice that from the freshness of y_1, y_2 it follows that $y_1, y_2 \notin \mathtt{dom}(\theta_1') \cup \mathtt{dom}(\theta_2')$ and $y_1, y_2 \notin \mathcal{V}(\theta_1') \cup \mathcal{V}(\theta_2)$. Hence,

$$\overline{\tau} = (\tau_1, \tau_2) = (\{y_1 \mapsto u_1, y_2 \mapsto u_2\}\theta_1', \{y_1 \mapsto v_1, y_2 \mapsto v_2\}\theta_2').$$

Thus, $(x\sigma)\overline{\tau} = (x\sigma'\delta)\overline{\tau} = (x\sigma')\delta\overline{\tau} = (x\sigma')\overline{\theta} = (s,t)$, and it follows that $x\sigma \in \mathtt{gen}_C(s,t)$.

2. This case is analogous to the previous.

□

Notice that Examples 3.3 and 3.4 both have final configurations that gives generalizers which are not least general ones. Therefore, this result cannot imply that the final configuration $\langle \emptyset \mid S \mid \sigma \rangle$ of $\mathtt{AUnif}_C$, will be such that $x\sigma \in \mathtt{lgg}_C(s,t)$.

The next result (Lemma 3.5) is used to prove Completeness of $\mathtt{AUnif}_C$.

Lemma 3.5 (Lemma 16 in [1]). Given terms $s, t \in T(\mathcal{X}, \Sigma_{\emptyset\cup C})$, if $u \in \mathtt{gen}_C(s,t)$ then there is a derivation $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P \mid S \mid \sigma \rangle$ such that $u \equiv_C x\sigma$.

Proof. The complete proof can be found in Appendix B, Lemma B.6. □

The next theorem estates that given $\mathtt{AUnif}_C$ a AUP$_C$, then every C-generalizer of s and t can be obtained modulo $\equiv_C$ by a derivation starting by $\mathtt{AUnif}_C(s,t)$.

Theorem 3.3 (Completeness of $\mathtt{AUnif}_C$ [1]). Let $s, t \in T(\mathcal{X}, \Sigma_{\emptyset\cup C})$. If $r \in \mathtt{lgg}_C(s,t)$, then there exists a derivation $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle \emptyset \mid S \mid \sigma \rangle$ such that $r \equiv_C x\sigma$.

Proof. If $r \in \mathtt{lgg}_C(s,t)$ it is clear that $r \in \mathtt{gen}_C(s,t)$, and then Lemma 3.5 yields that there exists a derivation $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P \mid S \mid \sigma \rangle$ such that $x\sigma \equiv_C r$.

Suppose by contradiction that $P = \{y : u \triangleq v\} \cup P'$, i.e, P is not the empty set. Then we have to analyse the following cases.

Case 1: If $\mathtt{root}(u) \neq \mathtt{root}(v)$.

There are two cases, depending on S.

1. If does not exist z such that $\{z : u \triangleq v\} \in S$.

   Then it follows that $\langle P' \cup \{y : u \triangleq v\} \cup \mid S \mid \sigma\rangle \Longrightarrow_{(Sol)} \langle P' \mid S \cup \{y : u \triangleq v\} \mid \sigma\rangle$. Then σ is not modified and $x\sigma$ stills the same.

2. If exists z such that $\{z : u \triangleq v\} \in S$.

   Thus, $\langle P' \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle \Longrightarrow_{(Rec)} \langle P' \mid S \mid \sigma\{y \mapsto z\}\rangle$. And it is easy to see that $r' = x\sigma\sigma'$ is less general than $r = x\sigma$, i.e $r <_C r'$, which is a contradiction since r is in $\mathtt{lgg}_C(s,t)$.

Case 2: If $\mathtt{root}(u) = \mathtt{root}(v) = f \in \Sigma_\emptyset$.

Thus, $u = f(\overline{u_n})$ and $v = f(\overline{v_n})$. It follows that,

$$\langle P' \cup \{y : f(\overline{u_n}) \triangleq f(\overline{v_n})\} \mid S \mid \sigma\rangle \Longrightarrow_{(Dec)} \langle P'' \mid S' \mid \sigma\sigma'\rangle$$

where $P'' = P_1 \cup \{y_1 : u_1 \triangleq v_1, \ldots, y_n : u_n \triangleq v_n\}$ and $\sigma' = \{y \mapsto f(\overline{y_n})\}$. By the same argument used above, $r = x\sigma <_C x\sigma\sigma' = r'$, we obtain a contradiction.

Case 3: If $\mathtt{root}(u) = \mathtt{root}(v) = g \in \Sigma_C$.

Thus, $u = g(u_1, u_2)$ and $v = g(v_1, v_2)$, hence

$$\langle P' \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle \Longrightarrow_{(C\text{-}Dec)} \langle P'' \mid S \mid \sigma'\rangle$$

where $P'' = P' \cup \{y_1 : u_1 \triangleq v_1, y_2 : u_2 \triangleq v_2\}$ or $P'' = P' \cup \{y_1 : u_1 \triangleq v_2, y_2 : u_2 \triangleq v_1\}$, depending on how *(C-Dec)* was applied. In both cases $\sigma' = \sigma\{y \mapsto g(y_1, y_2)\}$ and $r = x\sigma <_C x\sigma\sigma' = r$, and the result follows.

Therefore, the only possible case is **Case 1**. If there where several constraints in P' we should repeat the process and obtain $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle\emptyset \mid S \mid \sigma\rangle$ satisfying the theorem. □

A note on the finiteness of $\mathtt{lgg}_C(s,t)$.

Notice that given an AUP$_C$, $\mathcal{A}_C\langle s,t\rangle$, each step of $\mathtt{AUnif}_C(s,t)$ occurs by an application of a rule of Figure 3.1 in a (non)-solved constraint. Also by Remark 2.2, just one rule can be applied in each step. Therefore, it follows that given a derivation of the form $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P \mid S \mid \sigma\rangle$,

- if (Dec), (Sol) or (Rec) applies in $\langle P \mid S \mid \sigma\rangle$ then this configuration will have just one direct successor,

- if *(C-Dec)* applies in $\langle P \mid S \mid \sigma \rangle$ then it will have exactly two direct successors.

Therefore, $\texttt{AUnif}_C$ is finitely branching. Since the Theorem 3.1 yields that $\texttt{AUnif}_C$ always terminates, then by Lemma 1.5 it follows that $\texttt{AUnif}_C(s,t)$ is globally finite, i.e, it contains a finite number of direct successors.

Thus, the initial configuration of $\texttt{AUnif}_C$ will have a finite number of successors which are the normal forms w.r.t. $\Longrightarrow_{\texttt{AUnif}_C}$. By Theorem 3.2 these solutions will be enough to build a complete and minimal set of C-least general generalizers, $\texttt{lgg}_C(s,t)$. For s and t this construction is not done immediately by the C-anti-unification algorithm $\texttt{AUnif}_C$ because it may return repeated generalizers and also generalizers that are not least general ones.

Chapter 4

Associative Anti-Unification Problem

In this chapter we present a study of the the case of Associative Anti-Unification Problem. Similarly as Chapter 3, it is necessary to adapt the rules for $\texttt{AUnif}_\emptyset$, this time for handle the associativity. This adaptation takes place as adding two rules to $\texttt{AUnif}_\emptyset$ for Associative-Decompose *(A-Dec)*, that are presented in Figure 4.1 and restrings the usual (*Dec*) as was done in Figure 3.1 of the previous chapter.

We will study the main properties of $\texttt{AUnif}_A$, that are termination (Theorem 4.1), soundness (Theorem 4.2) and completeness (Theorem 4.3). Nonetheless, the proof of Theorem 3.3 presented in this chapter is different from that proposed by [1], since we had to fix an imprecision in the proof present in the paper that will be discussed later on.

In this chapter, the notions and notations established in Section 1.1.3 of Chapter 1 will be necessary.

It is important to say that we will consider only the case which there exists a unique associative function symbol, i.e., the case which Σ_A is unitary.

4.1 Motivating examples

As for commutativity, the algorithm to $\texttt{AUnif}_\emptyset$ that solves the Syntactic Anti-Unification Problem is not able to handle terms which are built with associative function symbols, i.e, terms in $T(\mathcal{X}, \Sigma_{\emptyset \cup A})$.

Example 4.1 (Applying $\texttt{AUnif}_\emptyset$ for solve $\mathcal{A}_A\langle s,t\rangle$.)**.** Let $h \in \Sigma_A$. Taking terms in $T(\mathcal{X}, \Sigma_{\emptyset \cup A})$ given as follows: $s = h(h(a,a), f(b,c))$ and $t = h(f(b,c), h(a,b))$ and trying to solve the anti-unification problem $\mathcal{A}_A\langle s,t\rangle$ with $\texttt{AUnif}_\emptyset$ it gives

$$\langle \{x : h(h(a,a),f(b,c)) \triangleq h(f(b,c),h(a,b))\} \mid \emptyset \mid id\rangle$$
$$\Downarrow (Dec)$$
$$\langle \{x_1 : h(a,a) \triangleq f(b,c), x_2 : f(b,c) \triangleq h(a,b)\} \mid \emptyset \mid \underbrace{x \mapsto h(x_1,x_2)}_{\sigma}\rangle$$
$$* \Downarrow (Sol)$$
$$\langle \emptyset \mid \{x_1 : h(a,a) \triangleq f(b,c), x_2 : f(b,c) \triangleq h(a,b)\} \mid \sigma\rangle$$

then $\texttt{AUnif}_\emptyset$ outputs $x\sigma = h(x_1,x_2)$ as the least general generalizer of s and t. But, s and t have associative function symbols, i.e. $s,t \in T(\mathcal{X},\Sigma_A)$ which implies that $h(x_1,h(a,y)) \in \texttt{gen}_A(s,t)$. Since $h(x_1,h(a,y)) <_E h(x_1,x_2)$, it follows $x\sigma \notin \texttt{lgg}_A(s,t)$. As is illustrated below

$$\begin{tikzcd}
 & x \arrow[d] \arrow[rr, dotted, dash] & & \text{Trivial solution} \\
 & \boxed{h(x_1,x_2)} \arrow[d] \arrow[rr, dotted, dash] & & \texttt{AUnif}_\emptyset\text{'s output} \\
 & h(x_1,h(a,y)) \arrow[dl] \arrow[dr] \arrow[rr, dotted, dash] & & \texttt{lgg}_A(s,t) \\
h(h(a,a),f(b,c)) & & h(f(b,c),h(a,b)) &
\end{tikzcd}$$

Example 4.2 (The set of $\texttt{lgg}_A$ is not unitary.). Let $h \in \Sigma_A$, then $s = h(h(a,b),h(b,c))$ and $t = h(h(b,b),c)$ are terms in $T(\mathcal{X},\Sigma_{\emptyset\cup A})$. It follows that

$$r_1 = h(x_1,h(x_1,x_4)) \text{ and } r_2 = h(x_1,h(b,c))) \in \texttt{gen}_A$$

Notice that $r_2 \not\leq_A r_1$. In fact, if $r_2 \leq_A r_1$ then there would exist a substitution λ such that $r_2\lambda = r_1$. However, $r_2\lambda = h(x_1\lambda,h(b,c)) \not\equiv_A h(x_1,h(x_1,x_4)) = r_1$ for any λ because substitutions do not change constant function symbols.

Similarly, $r_1 \not\leq_A r_2$. In fact, if $r_1 \leq_A r_2$ then would exist a substitution λ' such that $r_1\lambda' = h(x_1\lambda',h(x_1\lambda',x_4)\lambda')) \equiv_A h(x_1,h(b,c))$. Then, λ' must send x_1 into b, but if a substitution λ' is such that $x_1\lambda' = b$, then $r_1\lambda' = h(b,h(b,x_4\lambda')) \not\equiv_A r_2$.

Then, r_1 and r_2 are incomparable by $\equiv_A$. Hence r_1 and r_2 are both maximal elements of $\texttt{gen}_A$, and the set of least general generalizations modulo A is not unitary.

4.2 Solving the Associative Anti-Unification Problem

This section is based on the approach proposed by Alpuente et. al. in [1] for solving an Associative Anti-Unification Problem (for short AUP$_A$). The idea consists in adapting the syntactic algorithm $\texttt{AUnif}_{\emptyset}$ to handle the associative axiom, adding Associative-Decompose rules *(A-Dec)*, creating a new rule-based algorithm, $\texttt{AUnif}_A$ for solving the AUP$_A$.

The associative property allows us to analyse terms in a more practical way, discarding repeated applications of associative functions.This is done through a process called *flattening* as follows:

Definition 4.1 (Flattened term)**.** Given h an associative function symbol with $n \leq 2$ arguments, flattened terms are canonical forms w.r.t. the set of rules given by the following rule schema

$$h(x_1,\dots,h(t_1,\dots,t_m),\dots,x_n) \longrightarrow h(x_1,\dots,t_1,\dots,t_m,\dots,x_n)$$

For instance, $h(h(t_1,t_2),t_3)$ can be flattened to the term $h(t_1,t_2,t_3)$. Notice that

$$h(t_1,h(t_2,t_3)) \equiv_A h(h(t_1,t_2),t_3)$$

is also flattened to the same term.

In the following we will assume that h is an associative function symbol.

Remark 4.1. Note that given a flattened term $h(a,b,c,d,e)$ it does not mean the arity of h is 5. Since h is one associative function symbol its arity is 2, does not matter how many arguments this function symbol has in this flattened form the arity of h is still the same. The notation $h(a,b,c,d,e)$ represents all the following terms:

$$h(h(a,b),h(c,d)),h(a,h(h(b,c),d)),h(a,h(b,h(c,d))),$$

$$h(h(h(a,b),c),d),h(h(a,h(b,c)),d),$$

which are equivalent modulo associativity.

Given $\mathcal{A}_A\langle s,t\rangle$, we will use flatted versions of terms s and t in the initial configuration given as input to $\texttt{AUnif}_A$.

The rules for $\texttt{AUnif}_A$ are in Figure 4.1. The "old" rules inherited from $\texttt{AUnif}_{\emptyset}$ are opaque (in gray) and we emphasize the Associative-Decompose rules *(A-Dec)* which contains a rule for left-associativity *(A-Left)* and a rule for right associativity *(A-Right)*.

Basically, $\texttt{AUnif}_A$ adapts the rules in $\texttt{AUnif}_A$ and adds a new rule to handle with associative function symbols, namely Associative-Decompose and represented by *(A-Dec)*.

(Dec) **: Decompose**
$(f \in \Sigma_{\emptyset}^{n} \cup \mathcal{X})$

$$\langle P \cup \{x : f(\overline{s_n}) \triangleq f(\overline{t_n})\} \mid S \mid \sigma' \rangle \Longrightarrow \left\langle P \cup \begin{Bmatrix} x_1 : s_1 \triangleq t_1, \\ \vdots \\ x_n : s_n \triangleq t_n \end{Bmatrix} \mid S \mid \sigma\{x \mapsto f(\overline{x_n})\} \right\rangle$$

where $x_1, \ldots, x_n$ are fresh variables.

(Sol)**: Solve:**
If $\texttt{root}(s) \neq \texttt{root}(t)$ and there is no constraint $\{y : s \triangleq t\} \in S$

$$\langle P \cup \{x : s \triangleq t\} \mid S \mid \sigma \rangle \Longrightarrow \langle P \mid S \cup \{x : s \triangleq t\} \mid \sigma \rangle$$

(Rec)**: Recover:**
If $\texttt{root}(s) \neq \texttt{root}(t)$

$$\langle P \cup \{x : s \triangleq t\} \mid S \cup \{y : s \triangleq t\} \mid \sigma \rangle \Longrightarrow \langle P \mid S \cup \{y : s \triangleq t\} \mid \sigma\{x \mapsto y\} \rangle$$

(A-Dec): Associative-Decompose Rules

(A-Left): If $h \in \Sigma_A$, then

$$\langle P \cup \{x : h(\overline{s_n}) \triangleq h(\overline{t_m})\} \mid S \mid \sigma \rangle \Longrightarrow \langle P \cup \begin{Bmatrix} x_1 : h(s_1, \ldots, s_k) \triangleq t_1 \\ x_2 : h(s_{k+1}, \ldots, s_n) \triangleq h(t_2, \ldots, t_m) \end{Bmatrix} \mid S \mid \sigma\sigma' \rangle$$

with $\sigma' = \{x \mapsto h(x_1, x_2)\}$ and $1 \leq k \leq n-1$, where x_1, x_2 are fresh variables.

(A-Right): If $h \in \Sigma_A$, then

$$\langle P \cup \{x : h(\overline{s_n}) \triangleq h(\overline{t_m})\} \mid S \mid \sigma \rangle \Longrightarrow \langle P \cup \begin{Bmatrix} x_1 : s_1 \triangleq h(t_1, \ldots, t_k) \\ x_2 : h(s_2, \ldots, s_n) \triangleq h(t_{k+1}, \ldots, t_m) \end{Bmatrix} \mid S \mid \sigma\sigma' \rangle$$

with $\sigma' = \{x \mapsto h(x_1, x_2)\}$ and $1 < k \leq m-1$, where x_1, x_2 are fresh variables.

Fig. 4.1 $\texttt{AUnif}_A$ simplification rules

Now, we will describe the Associative Decompose rules *(A-Dec)*.

(A-Left) In the case $\{x : h(s_1,s_2,\dots,s_n) \triangleq h(t_1,t_2,\dots,t_m)\}$ is a constraint, where $h \in \Sigma_A$ and m and n are positive integers not necessarily equal, we associate the arguments of h to the left in several ways, until we cover all the possibilities, thus this rule branches. In fact, we could compare s_1 with t_1 and $h(s_2,\dots,s_n)$ with $h(t_2,\dots,t_m)$, obtaining two new constraints $\{x_1 : s_1 \triangleq t_1, x_2 : h(s_2,\dots,s_n) \triangleq h(t_2,\dots,t_m)\}$, and a new substitution $\sigma = \{x \mapsto h(x_1,x_2)\}$. But we could have chosen to associate differently, as $h(s_1,s_2)$ with t_1 and $h(s_3,\dots,s_n)$ with $h(t_2,\dots,t_m)$, or $h(s_1,s_2,s_3)$ with t_1 and $h(s_4,\dots,s_n)$ with $h(t_2,\dots,t_m)$, and so on. For each choice, we generate two different constraints and the substitution $\sigma = \{x \mapsto h(x_1,x_2)\}$, where the variable x_1 and x_2 are the index variables of the constraints created.

(A-Right) This rule is similar to the rule *(A-Left)*, but now arguments of h are associated to the right. For instance, we can compare s_1 with $h(t_1,t_2)$ and $h(s_2,\dots,s_n)$ with $h(t_3,\dots,t_m)$ obtaining new constraints $\{x_1 : s_1 \triangleq h(t_1,t_2), x_2 : h(s_2,\dots,s_n) \triangleq h(t_3,\dots,t_m)\}$, and a new substitution $\sigma = \{x \mapsto h(x_1,x_2)\}$. But we could have chosen to associate differently, as s_1 with $h(t_1,t_2,t_3)$ and $h(s_2,\dots,s_n)$ with $h(t_4,\dots,t_m)$, and so on.

Notice that in rules *(A-Left)* and *(A-Right)*, for $k = 1$, $h(t_1)$ or $h(s_1)$ denote t_1 and s_1 respectively.

Example 4.3 (How to apply *(A-Dec)*)**.** Given flattened terms $h(s_1,s_2,s_3)$ and $h(t_1,t_2,t_3,t_4)$. We will apply $\texttt{AUnif}_A$ in the initial configuration $\langle\{x : h(s_1,s_2,s_3) \triangleq h(t_1,t_2,t_3,t_4)\} \mid \emptyset \mid id\rangle$ that will produce configurations $C_1 = \langle P_1 \mid \emptyset \mid \{x \mapsto h(x_1,x_2)\}\rangle$, $C_2 = \langle P_2 \mid \emptyset \mid \{x \mapsto h(x_1,x_2)\}\rangle$, $C_3 = \langle P_3 \mid \emptyset \mid \{x \mapsto h(x_1,x_2)\}\rangle$ and $C_4 = \langle P_4 \mid \emptyset \mid \{x \mapsto h(x_1,x_2)\}\rangle$that differ only in their set of unsolved-constraints. Thus, we will show only the set of unsolved constraint of each configuration.

$$\begin{tikzcd}
 & & \langle\{x : h(s_1,s_2,s_3) \triangleq h(t_1,t_2,t_3,t_4)\} \mid \emptyset \mid id\rangle \arrow[dll, Rightarrow, "\textit{(A-Left)}"'] \arrow[dl, Rightarrow, "\textit{(A-Left)}"] \arrow[d, Rightarrow, "\textit{(A-Right)}"] \arrow[dr, Rightarrow, "\textit{(A-Right)}"] & \\
\underbrace{\left\{\begin{matrix} x_1 : s_1 \triangleq t_1, \\ x_2 : h(s_2,s_3) \triangleq h(t_2,t_3) \end{matrix}\right\}}_{P_1} & \underbrace{\left\{\begin{matrix} x_1 : h(s_1,s_2) \triangleq t_1, \\ x_2 : s_3 \triangleq h(t_2,t_3) \end{matrix}\right\}}_{P_2} & \underbrace{\left\{\begin{matrix} x_1 : s_1 \triangleq h(t_1,t_2), \\ x_2 : h(s_2,s_3) \triangleq h(t_3,t_4) \end{matrix}\right\}}_{P_3} & \underbrace{\left\{\begin{matrix} x_1 : s_1 \triangleq h(t_1,t_2,t_3), \\ x_2 : h(s_2,s_3) \triangleq t_4 \end{matrix}\right\}}_{P_4}
\end{tikzcd}$$

Notice that in the example above, we had applied *(A-A)* in just one constraint of the initial configuration, creating four different configurations. It is easy to see `AUnif`$_A$ reduction tree grows very fast, since each application of *(A-A)* add new branches to it.

In the following, to easy readability in this section we will only show convenient reductions related to *(A-Dec)*, omitting the others. This convenient branches are that which normal forms are not give repeated generalizers.

Remark 4.2. The original definition of *(A-Right)* in [1] does not restrict to $k > 1$. However, if $k = 1$, the application of both *(A-Left)* and *(Right-Dec)* rules will create repeated configurations:

$$\begin{tikzcd}
 & \langle \{x : h(\overline{s_n}) \triangleq h(\overline{t_m})\} \mid S \mid \sigma \rangle \arrow[dl, Rightarrow, "\textit{(A-Left)}_{k=1}"'] \arrow[dr, Rightarrow, "\textit{(A-Right)}_{k=1}"] & \\
\langle \begin{Bmatrix} y_1 : s_1 \triangleq t_1 \\ y_2 : h(s_2 \dots, s_n) \triangleq h(t_2, \dots, t_m) \end{Bmatrix} \mid S \mid \sigma' \rangle & & \langle \begin{Bmatrix} y_1 : s_1 \triangleq t_1 \\ y_2 : h(s_2, \dots, s_n) \triangleq h(t_2, \dots, t_m) \end{Bmatrix} \mid S \mid \sigma' \rangle
\end{tikzcd}$$

To avoid this we will estrict *(A-Right)* to $1 < k \leq m-1$, as have written in Figure 4.1.

Example 4.4 (Applying `AUnif`$_A$.)**.** Given $h \in \Sigma_A$, taking terms $s = h(a, h(b, b))$ and $t = h(h(a, c), c)$, we will apply `AUnif`$_A$ rules to resolve $\mathcal{A}_A$. First, it is necessary to put s and t in their respective flattened forms: $h(a, b, b)$ and $h(a, c, c)$. Second, we apply `AUnif`$_A$ rules in the initial constraint $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle$. Hence in the first step of `AUnif`$_A$ application of *(A-Dec)* rules makes the reduction branches to the configurations $C_1 = \langle P_1 \mid \emptyset \mid \{x \mapsto h(x_1, x_2)\} \rangle$, $C_2 = \langle P_2 \mid \emptyset \mid \{x \mapsto h(x_1, x_2)\} \rangle$ and $C_3 = \langle P_3 \mid \emptyset \mid \{x \mapsto h(x_1, x_2)\} \rangle$ that differ only in their set of unsolved-constraints, see figure below.

$$\begin{tikzcd}
 & \langle x : h(a, b, b) \triangleq h(a, c, c) \mid \emptyset \mid id \rangle \arrow[dl, Rightarrow, "\textit{(A-Left)}"'] \arrow[d, Rightarrow, "\textit{(A-Left)}"] \arrow[dr, Rightarrow, "\textit{(A-Right)}"] & \\
\underbrace{\begin{Bmatrix} x_1 : a \triangleq a \\ x_2 : h(b, b) \triangleq h(c, c) \end{Bmatrix}}_{P_1} & \underbrace{\begin{Bmatrix} x_1 : h(a, b) \triangleq a \\ x_2 : b \triangleq h(c, c) \end{Bmatrix}}_{P_2} & \underbrace{\begin{Bmatrix} x_1 : a \triangleq h(a, c) \\ x_2 : h(b, b) \triangleq c \end{Bmatrix}}_{P_3}
\end{tikzcd}$$

If we continue the derivation, we obtain the following derivation tree:

$$\begin{tikzcd}
 & \langle x : h(a,b,b) \triangleq h(a,c,c) \mid \emptyset \mid id\rangle \arrow[dl, Rightarrow, "\textit{(A-Left)}"'] \arrow[d, Rightarrow, "\textit{(A-Right)}"] \arrow[dr, Rightarrow, "\textit{(A-Right)}"] & \\
C_1 \arrow[d, Rightarrow, "\mathtt{AUnif}_A"', "*"] & C_2 \arrow[d, Rightarrow, "\mathtt{AUnif}_A"', "*"] & C_3 \arrow[d, Rightarrow, "\mathtt{AUnif}_A"', "*"] \\
\langle \emptyset \mid \{y : b \triangleq c\} \mid \sigma_1\rangle & \langle \emptyset \mid P_2 \mid \sigma_2\rangle & \langle \emptyset \mid P_3 \mid \sigma_2\rangle
\end{tikzcd}$$

Where $\sigma_1 = \{x \mapsto h(a,h(y,y))\}$ and $\sigma_2 = \{x \mapsto h(x_1,x_2)\}$. Hence, $\mathtt{AUnif}_A(s,t)$ output two different substitutions σ_1 and σ_2 that produce two different terms, $x\sigma_1 = h(a,h(y,y))$ and $x\sigma_2 = h(x_1,x_2)$. It is easy to see that $h(a,h(y,y))$ is less general than $h(x_1,x_2)$. Therefore we can conclude that

1. $\mathtt{AUnif}_A$ can output terms that are not least general generalizers,
2. $\Longrightarrow_{\mathtt{AUnif}_A}$ is not confluent because $C_1 \ {}_{\mathtt{AUnif}_A}\!\Longleftarrow \langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \Longrightarrow_{\mathtt{AUnif}_A} C_2$ and there is no configuration C such that $C_1 \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} C \ {}_{\mathtt{AUnif}_A}\!\overset{*}{\Longleftarrow} C_2$.

For the next results it is necessary to recall the notions and notations established in Definition 1.6 such as $(p)^k$ for the prefix of p at depth k and $(p)_k$ for the index of p at depth k.

Associative pair and associative conflict pair of subterms.

In the following we will introduce some notions in order to formally define associative pair of subterms and associative conflict pair of subterms. The goal of associative pairs is to capture exactly the pair of subterms (u,v) of s and t which should being considering to solve the AUP$_A$ for s and t, $\mathcal{A}_A\langle s,t\rangle$. While the goal of the definition of conflict associative pair of subterms is identify when the procedure of comparing these subterms must stop by identifying the point where the term tree of s and t diverges.

For instance, let $f_1 : 2, f_2 : 1 \in \Sigma_\emptyset$ and $h \in \Sigma_A$. Taking the flattened terms

$$s = f_1(h(a,b,c), f_2(x)) \text{ and } t = f_1(h(d,e), f_2(a))$$

and considering the AUP$_A$ $\mathcal{A}_A\langle s,t\rangle$. The rules of $\mathtt{AUnif}_A$ yields the following derivation:

$$\begin{tikzcd}
\langle\{y : f_1(h(a,b,c), f_2(x)) \triangleq f_1(h(d,e), f_2(a))\} \mid \emptyset \mid id\rangle \arrow[d, Rightarrow, "(Dec)"'] \\
\langle\{y_1 : h(a,b,c) \triangleq h(b,c), y_2 : f_2(x) \triangleq f_2(a)\} \mid \emptyset \mid \{y \mapsto f_1(y_1,y_2)\}\rangle.
\end{tikzcd}$$

That splits $\mathcal{A}_A\langle s,t\rangle$ in two new problems: $\mathcal{A}_A\langle h(a,b,c),h(d,e)\rangle$ and $\mathcal{A}_A(f_2(x),f_2(a))$. However, that are two possible ways to express $\mathcal{A}_A\langle h(a,b,c),h(d,e)\rangle$ in unflattened terms, that are $\mathcal{A}_A(h(h(a,b),c),h(d,e))$ and $\mathcal{A}_A(h(a,h(b,c)),h(d,e))$.

Notice that for any possibility to organize the associative function symbols of $h(a,b,c)$ and $h(d,e)$, there is no way to compare a with e, or $h(a,b)$ with e. Thus, (a,e) and $(h(a,b),e)$ should not being associative pair of subterms of s and t. Similarly, (b,c), (b,d), (d,c) and $(h(b,c),d)$ should not being associative pairs of subterms either.

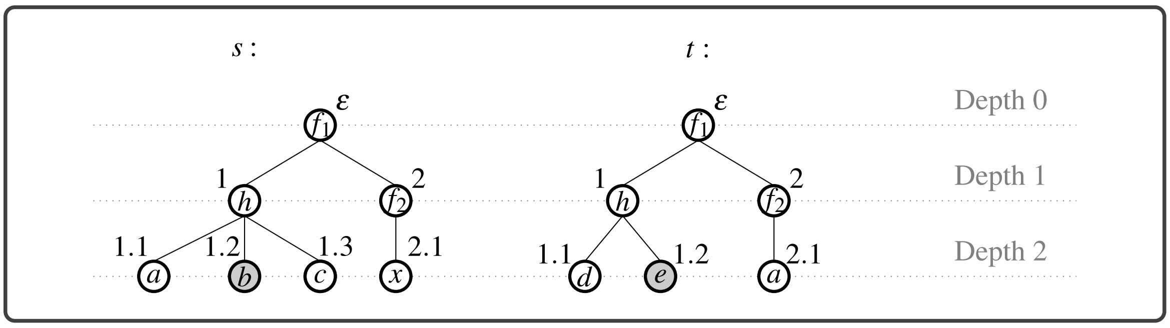
Therefore, the definition of associative pair of subterms must work to delete these kind of pair of subterms of s and t that are not necessary to analyse to solve the main problem $\mathcal{A}_A\langle s,t\rangle$.

To define associative pair of subterms Alpuente et. al. [1] uses an auxiliary definition, that is stated in the box below. However, this definition has a lack of conditions that ends up naming some pairs (u,v) as associative pairs subterms of s and t when they should not being considered to solve $\mathcal{A}_A\langle s,t\rangle$. In the following, we will discuss this lack of conditions and propose a new auxiliary definition that avoids this mistake.

Definition 4.2 (Associative pair of positions proposed in [1].)**.** Given flattened versions of terms $s,t \in T(\mathcal{X},\Sigma_{\emptyset\cup A})$, and positions $p \in \texttt{pos}(s)$, and $p' \in \texttt{pos}(t)$, the pair (p,p') of positions is called associative pair of positions of s and t iff

1. $\texttt{depth}(p) = \texttt{depth}(p')$,
2. for each $0 \leq i < \texttt{depth}(p), \texttt{root}(s|_{(p)^i}) = \texttt{root}(t|_{(p')^i})$, and
3. for each $0 < j \leq \texttt{depth}(p)$:
   a. if $\texttt{root}(s|_{(p)^{j-1}}) = f \in \Sigma_\emptyset$ then $(p)_j = (p')_j$; i.e, if the root of the subterm of s at prefix position $(p)^{j-1}$ is a free function symbol $f \in \Sigma_\emptyset$, then p and p' have the same index at depth j.
   b. if $\texttt{root}(s|_{(p)^{j-1}}) = h \in \Sigma_A$, then no restriction on $(p)_j$ and $(p')_j$; i.e, if the root of the subterm of s at prefix position $(p)^{j-1}$ is an associative function symbol $h \in \Sigma_A$, then there is no condition over the indexes of p and p' at depth p

Example 4.5. Taking the terms $s = f_1(h(a,b,c),f_2(x))$ and $t = f_1(h(d,e),f_2(a))$, with respective positions illustrated bellow. It follows that the Definition 4.2 yields that $(p,p') = (1.2,1.2)$ is an associative pair of positions.



In fact,

1. $\mathtt{depth}(p) = 2 = \mathtt{depth}(p')$, then the condition 1 holds.
2. Notice that $(p)^0 = (p')^0 = \varepsilon$ and $\mathtt{root}(s|_\varepsilon) = f_1 = \mathtt{root}(t|_\varepsilon)$.

   Also, $(p)^1 = 1 = (p')^1 = 1$ and $\mathtt{root}(s|_1) = h = \mathtt{root}(t|_1)$. Thus, the condition 2 holds.
3. For $i = 1$, we have that $(p)^{1-1} = \varepsilon$. Since $\mathtt{root}(s|_\varepsilon) = f \in \Sigma_\emptyset$, we have to verify the condition on the index "$(p)_i$", with $i = 1$, (condition 3.a. of Definition 4.5)

$$(p)_1 = (1.2)_1 = 2 = (p')_1.$$

   Similarly for $i = 2$, it follows $(p)^{i-1} = 1$. Since $\mathtt{root}(s|_1) = h \in \Sigma_A$, there is no condition for the indexes of p and p' at depth 2.

Therefore, $(p, p') = (1.2, 1.2)$ is an associative pair of positions of s and t.

Definition 4.3 (Associative Pair of Subterms)**.** Let $s,t \in T(\mathcal{X}, \Sigma_{\emptyset\cup A})$ be terms in flattened form. The pair of terms (u, v) is called an **associative pair of subterms** of s and t iff

1. **(Regular Subterms)** For each pair of positions $p \in \mathtt{pos}(s)$ and $p' \in \mathtt{pos}(t)$ such that $s|_p = u, t|_{p'} = v$ and (p, p') is an associative pair of positions of s and t. Or:
2. **(Associative pair of subterms)** There are positions $p \in \mathtt{pos}(s)$, and $p' \in \mathtt{pos}(t)$ such that the following conditions are satisfied:

   a. (p, p') is an associative pair of positions of s and t;

   b. $u = h(u_1, \ldots, u_{n_u})$ and $v = h(v_1, \ldots, v_{n_v})$, with $n_u, n_v \geq 1$, $h \in \Sigma_A$;

   c. $s|_p$ and $t|_{p'}$ are of the form

$$s|_p = h(s_1, \ldots, s_{k_1}, u_1, \ldots, u_{n_u}, s_{k_2}, \ldots, s_{n_p})$$

$$t|_{p'} = h(t_1,\dots,t_{k'_1},v_1,\dots,v_{n_v},t_{k'_2},\dots,t_{n_{p'}}),$$

with $n_p, n_{p'} \geq 2$, and

- $\{s_1,\dots,s_{k_1}\} = \emptyset$ (no arguments before u_1) iff $\{t_1,\dots,t_{k'_1}\} = \emptyset$ (no arguments before v_1),
- $\{s_{k_2},\dots,s_n\} = \emptyset$ (no arguments after u_{n_u}) iff $\{t_{k'_2,\dots,t_m}\}$ (no arguments after v_{n_v}).

The following example will show how Definition 4.2 trigger a mistake in Definition 4.3.

Example 4.6 (Cont. Example 4.5)**.** Taking $s = f_1(h(a,b,c), f_2(x))$ and $t = f_1(h(d,e), f_2(a))$, since $(s|_{1.2}, t|_{1.2}) = (b,e)$ and the Definition 4.2 yields that $(1.2, 1.2)$ is an associative pair of subterms, it follows that (b,e) is an associative pair of positions.

However, as previously discussed, this pair of subterms should not being consider to solve the AUP$_A$ $\mathcal{A}_A\langle s,t\rangle$, then the conclusion of Example 4.6 is wrong.

It happens because the Definition 4.2 does not put any condition for the indexes of a pair of position deeper than a position headed with an associative function symbol $h \in \Sigma_A$.

Then, the Condition 1. of Definition 4.3 "if (p,p) is an associative pair of positions, then $(s|_p, t|_{p'})$ is an associative pair of subterms" implies that some associative pair of subterms are associative whereas they should not be. It is interesting since the Condition 2.c of Definition 4.3 tries to exactly avoid this kind of mistake, but it ends up failing. Therefore, it is necessary to pay more attention in the positions under positions whose buttons are associative function symbols.

In following we will presenting the definition of associative pair of positions that will be used in this dissertation. First, we state the notion of the number of immediate arguments of an associative function symbol.

Definition 4.4. Let $h \in \Sigma_A$, then given a flattened term $s = h(s_1,\dots,s_n)$ the number **of immediate arguments of** s is denoted by $\mathtt{arg}(s) = n$.

For instance, for $s = h(a,b,c)$ and $t = h(d,e)$, we have $\mathtt{arg}(s) = 3$ and $\mathtt{arg}(t) = 2$.

Definition 4.5 (Associative pair of positions)**.** Given flattened versions of terms $s,t \in T(\mathcal{X}, \Sigma_{\emptyset\cup A})$, and positions $p \in \mathtt{pos}(s)$, and $p' \in \mathtt{pos}(t)$, the pair (p,p') of positions is called **associative pair of positions of** s **and** t iff the following conditions are satisfied

1. $\mathtt{depth}(p) = \mathtt{depth}(p')$; and
2. for each $0 \leq i < \mathtt{depth}(p)$, $\mathtt{root}(s|_{(p)^i}) = \mathtt{root}(t|_{(p')^i})$, i.e, $s|_p$ and $t|_{p'}$ have the same root as prefix i; and

3. for each $0 < j \leq \mathtt{depth}(p)$:

   a. if $\mathtt{root}(s|_{(p)^{j-1}}) = f \in \Sigma_\emptyset$ then $(p)_j = (p')_j$, i.e, if the root of the subterm of s at prefix position $(p)^{j-1}$ is a free function symbol $f \in \Sigma_\emptyset$, then p and p' have the same index at depth j,

   b. if $\mathtt{root}(s|_{(p)^{j-1}}) = h \in \Sigma_A$, then

      – $(p)_j = 1$ iff $(p')_j = 1$, i.e, the subterm $s|_{(p)^j}$ is the first argument of $s|_{(p)^{j-1}}$ iff $t|_{(p)^j}$ is the fist argument of $t|_{(p)^{j-1}}$.
      – $(p)_j = \mathtt{arg}(s|_{(p)^{j-1}})$ iff $(p')_j = \mathtt{arg}(t|_{(p')^{j-1}})$, i.e, the subterm $s|_{(p)^j}$ is the last argument of $s|_{(p)^{j-1}}$ iff $t|_{(p')^j}$ is the last argument of $t|_{(p')^{j-1}}$.
      – $1 < (p)_j < \mathtt{arg}(s|_{(p)^{j-1}})$ iff $1 < (p')_j < \mathtt{arg}(t|_{(p')^{j-1}})$, i.e., the position $p \in \mathtt{pos}(s)$ has an intermediary index (not the first or the last) at depth j iff the position $p' \in \mathtt{pos}(t)$ have an intermediary index at depth j too.

The condition 3 is illustrated in Figure 4.2.

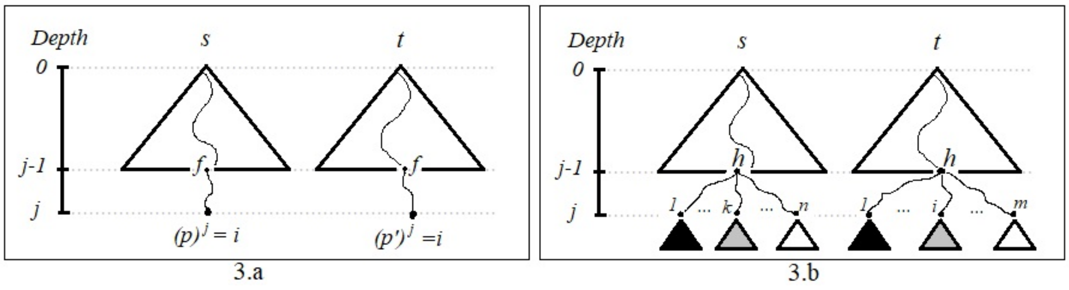


Fig. 4.2 Conditions 3.a (at left) and 3.b. (at right) of Definition 4.5, where $n = \mathtt{arg}(s|_{(p)^{j-1}})$ and $m = \mathtt{arg}(t|_{(p')^{j-1}})$.

Example 4.7. Taking terms $s = f_1(h(a,b,c), f_2(a))$ and $t = f_1(h(a,e), f_2(y))$, we will show that (b,e) is not an associative pair of subterms.

1. First, notice that the $(p,p') = (1.2, 1.2)$ is not an associative pair of positions.

   In fact, for $i = 1$ it follows that $\mathtt{root}(s|_{(p)^{2-1}}) = h \in \Sigma_A$, then by the condition 2.b of Definition 4.5 it is necessary to analyse the indexes of the 1.2 at depth 2. Since $\mathtt{arg}(s|_{(p)^{2-1}}) = \mathtt{arg}(h(a,b,c)) = 3$ and $\mathtt{arg}(t|_{(p')^{2-1}}) = \mathtt{arg}(h(c,d)) = 2$ it follows that $(p)_2 = 2 < \mathtt{arg}(s|_{(p)^{2-1}})$ and $(p')_2 = 2 = \mathtt{arg}(t|_{(p')^{2-1}})$. Thus, $(p,p') = (1.2, 1.2)$ is not an associative pair of positions of s and t and since $(s|_p, t|_{p'}) = (b,e)$, it follows that (b,e) does not satisfies the condition 1. of Definition 4.3.

2. Thus, to show that (b,e) is not an associative pair of positions it remains to show that it does not satisfies the Condition 2 of Definition 4.3. We have that s and t are such that $s|_\varepsilon = h(a,b,c)$, $t|_\varepsilon = h(d,e)$ thus there exists the argument c after b in $s|_\varepsilon$, while there is no argument after e in $t|_\varepsilon$. Thus, (b,e) does not satisfies the Condition 2 of Definition 4.3.

Therefore, (b,e) is not an associative pair of subterms of s and t.

In the next example we will still studying the terms $s = h(a,b,c)$ and $t = h(b,c)$, showing their associative pair of subterms.

Example 4.8 (Cont. of Example 4.7)**.** Consider the terms $s = f_1(h(a,b,c), f_2(x))$ and $t = f_1(h(d,e), f_2(a))$.

For $(q,q') = (1.1, 1.1)$, the analysis of the conditions 1, 2 and 3.a of Definition 4.5 is analogously as the previously did in Example 4.7. We have that $\texttt{root}(s|_{2-1}) = h \in \Sigma_A$, then it is necessary to analyse the index of q and q' at depth 2 as required by the condition 3.c of Definition 4.5; since $(q)_2 = 1 = (q')_2$, (q,q') satisfies it. Thus $(1.1, 1.1)$ is an associative pair of positions of s and t.

Similarly, $(\varepsilon,\varepsilon)$, $(1,1)$, $(2,2)$, $(1.3,1.2)$ and $(2.1,2.1)$ are associative pairs of positions of s and t too.

The regular associative pair of subterms of s and t are given by the associative pair of positions, as is shown in the table below.

Associative pair of positions	Associative pair of subterms
(ε)	(s,t)
$(1,1)$	$(h(a,b,c), h(d,e))$
$(2,2)$	$(f_2(x), f_2(a))$
$(1.1,1.1)$	(a,d)
$(1.3,1.2)$	(c,e)
$(2.1,2.1)$	(x,a)

The associative pair $(h(a,b,c), h(d,e))$ is such that $h(a,b,c)$ is flattened and it will be necessary to move it back to the unflattened form in order to obtain the pairs of subterms of s and t that should being compared to solve $\mathcal{A}_A\langle h(a,b,c), h(d,e)\rangle$ depending on the ways that $h(a,b,c)$ becomes unflattened. We have already seen that there are multiple ways of doing that, and the Condition 2. of Definition 4.3 capture these ways by establishing how to group the arguments of flattened associative pairs. Thus, by this condition, it follows that $(h(a,b),d)$ and $(h(b,c),e)$ are also associative pairs of subterms of s and t.

On other hand, the Condition 2.c of Definition 4.3 implies that $(h(a,b),e)$ and $(h(b,c),d)$ are not associative pair of subterms.

Definition 4.6 (Associative Conflict Pair)**.** Given terms $s,t \in T(\mathcal{X},\Sigma_{\emptyset\cup A})$, the pair (u,v) is called an **associative conflict pair of** s **and** t if and only if $\mathtt{root}(u) \neq \mathtt{root}(v)$ and (u,v) is an associative pair of subterms.

Example 4.9 (Cont. of Example 4.8)**.** We have that (a,d) is one associative pair of $f_1(h(a,b,c),f_2(x))$ and $f_1(h(d,e),f_1(a))$, and since $\mathtt{root}(a) \neq \mathtt{root}(e)$ it follows (a,d) is one associative conflict pair.

If u is a subterm of s and v is a subterm of t, the (u,v) being an associative pair of subterms of s and t means that to solve the AUP$_A$ for s and t it is necessary solve the AUP$_A$ for u and v, and so on, going deeper in the comparisons of s and t terms trees, until reach in a associative conflict pair (u',v') which only has trivial generalizations. For more details, see Lemma 4.2, 4.3, 4.4 and Lemma 4.5.

Definition 4.7. Given one configuration $\langle P \mid S \mid \sigma \rangle$ with $P = P' \cup \{x : s \triangleq t\}$, where s and t are flattened terms we define the following measure: $m(\langle P \mid S \mid \sigma \rangle) = (n_1,n_2)$, where

- $n_1(P)$ is the number of non-associative symbol occurrences in P.
- $n_2(P)$ is the number of arguments under the associative function symbol in the root position of a constraint, i.e:
  - if both $\mathtt{root}(s)$ and $\mathtt{root}(t)$ are non-associative function symbols then $n_2(P) = n_2(P') + 0$,
  - if $s = h(s_1,\dots,s_n)$ where h is one associative function symbol and $\mathtt{root}(t) = f \in \Sigma_\emptyset$ then $n_2(P) = n_2(P') + n$ (similarly if $t = h(t_1,\dots,t_n)$ and $\mathtt{root}(s) = f \in \Sigma_\emptyset$),
  - if $s = h_1(s_1,\dots,s_n)$ and $t = h_2(t_1,\dots,t_m)$ with $h_1,h_2 \in \Sigma_A$ then $n_2(P) = n_2(P') + n + m$

4.3 Properties

In this section we present the main properties of the $\mathtt{AUnif}_A$ algorithm, such as termination (Theorem 4.1), soundness (Theorem 4.2) and completeness (Theorem 4.3). This results were initially proven in [1], here we present more detailed versions of the proofs adapting them to handle with the new definitions and also fixing an inaccuracy in the proof of completeness.

Theorem 4.1 (Termination of $\texttt{AUnif}_A$ [1]). Let $\mathcal{A}_A\langle s,t\rangle$ be AUP_A. Every derivation of $\texttt{AUnif}_A$ starting from $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ terminates in a final configuration of the form $\langle\emptyset \mid S \mid \sigma\rangle$.

Proof. For a reduction step in which *(A-Dec)* rules were applied, we have the following shape

$$\begin{tikzcd}
 & & C = \langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \arrow[dll, Rightarrow] \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] \arrow[drr, Rightarrow] & & \\
\underbrace{\langle P_1 \mid S_1 \mid \sigma_1\rangle}_{C_1} & \underbrace{\langle P_2 \mid S_2 \mid \sigma_2\rangle}_{C_2} & \cdots & \underbrace{\langle P_{l-1} \mid S_{l-1} \mid \sigma_{l-1}\rangle}_{C_{l-1}} & \underbrace{\langle P_l \mid S_l \mid \sigma_l.\rangle}_{C_l}
\end{tikzcd}$$

And, for the other rules, the shape is $C = \langle P \mid S \mid \sigma\rangle \Longrightarrow_{(R)} \langle P_1 \mid S_1 \mid \sigma_1\rangle = C_1$. To deal with the branches obtained from *(A-Dec)* rules application, we will consider a more general form of configuration $\{\!\{C\}\!\} \Longrightarrow_{\textit{(A-Dec)}} \{\!\{C_1,\ldots,C_l\}\!\}$, where $\{\!\{\}\!\}$ denotes a multiset. And for other the rules, simply $\{\!\{C\}\!\} \Longrightarrow_{(R)} \{\!\{C_1\}\!\}$.

Now, to prove that $\texttt{AUnif}_A$ terminates we will show that for each step $M_1 \Longrightarrow_{(R)} M_2$, where M_1 and M_2 denotes the the mulstiset of configurations presented above, it holds that $M_1 >_{mul} M_2$, where $>_{mul}$ is the multiset order induced by the lexicographic product $>_{\mathbb{N}} \times >_{\mathbb{N}}$ of the standard order $>$ (over naturals) on pairs $(n_1(P), n_2(P)) \in \mathbb{N}\times\mathbb{N}$ from $m(\langle P \mid S \mid \sigma\rangle)$ as given in Definition 4.7.

The proof proceed for analysis of each inference rule (R) of Figure 4.1.

Case *(R) = (Dec)*:

The reduction $\{\!\{C\}\!\} \Longrightarrow_{(Dec)} \{\!\{C_1\}\!\}$ is such that

$$\begin{aligned}
C &= \langle P' \cup \{y : f(\overline{s_n}) \triangleq f(\overline{t_n})\} \mid S \mid \sigma'\rangle,\\
C_1 &= \langle P' \cup \{y_1 : s_1 \triangleq t_1, \ldots, s_n \triangleq t_n\} \mid S_1 \mid \sigma_1\rangle.
\end{aligned}$$

where $P = P' \cup \{y : f(\overline{s_n}) \triangleq f(\overline{t_n})\}$ and $P_1 = P' \cup \{y_1 : s_1 \triangleq t_1, \ldots, s_n \triangleq t_n\}$. Then, $n_1(P) = n_1(P') + r + 2$ and $n_1(P_1) = n_1(P') + r$, where r is the number of non-associative functions symbols in $s_1,\ldots,s_n$, $t_1,\ldots,t_n$. Hence $n_1(P) > n_1(P_1)$, which implies that $m(C) >_{lex} m(C_1)$. Therefore, $M_1 = \{\!\{C\}\!\} >_{mul} \{\!\{C_1\}\!\} = M_2$.

Case *(R) = (Sol)*:

Then, $\{\!\{C\}\!\} = \{\!\{\langle P_2 \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle\}\!\} \Longrightarrow_{(R)} \{\!\{\langle P_2 \mid S \cup \{y : u \triangleq v\} \mid \sigma\rangle\}\!\} = \{\!\{C_1\}\!\}$ where $P_1 = P_2 \cup \{y : u \triangleq v\}$. By Definition 4.7, it follows that $n_1(P_1) = n_1(P_2) + r$, where r is the number of occurrences of non-associative function symbols in u and v. Thus, $n_1(P_1) = n_1(P_2 \cup \{y : u \triangleq v\}) > n_1(P_2)$. Therefore, $m(C) >_{lex} m(C_1)$ yields that $M_1 = \{\{C\}\} >_{mul} \{\{C_1\}\} = M_2$.

Case *(R)* = (Rec):

Then, $\{\{C\}\} = \{\{\langle P_2 \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle\}\} \Longrightarrow_{(R)} \{\{\langle P_2 \mid S \mid \sigma_1\rangle\}\} = \{\{C_1\}\}$ where the constraint $\{y : u \triangleq v\}$ is in S, and $P_1 = P_2 \cup \{y : u \triangleq v\}$. By Definition 4.7, it follows that $n_1(P_1) > n_1(P_2)$ by the same argument used in the case above.

Case *(R)* = *(A-Left)*:

Then,

$$C \Longrightarrow_{(A-Left)} \langle P' \cup \left\{ \begin{matrix} x_1 : h(s_1, \ldots, s_k) \triangleq t_1, \\ x_2 : h(s_{k+1}, \ldots, s_n) \triangleq h(t_2, \ldots, t_m) \end{matrix} \right\} \mid S_k \mid \sigma_k\rangle = C_k$$

with $1 \leq k \leq n-1$. Therefore, the reduction branches and $M_1 = \{\{C\}\}$ and $M_2 = \{\{C_1, \ldots . C_{n-1}\}\}$

Let

$$P = P' \cup \{x : h(s_1, \ldots, s_n) \triangleq h(t_1, \ldots, t_m)\}, \text{ and}$$
$$P_k = P' \cup \{x_1 : h(s_1, \ldots, t_k) \triangleq s_k, x_2 : h(s_{k+1}, \ldots, s_n) \triangleq h(t_2, \ldots, t_m)\}.$$

Thus by Definition 4.7 it follows that $n_1(P) = n_1(P') + r = n_1(P_k)$, where r is the number of occurrences of non-associative function symbols in $s_1, \ldots, s_n$, $t_1, \ldots, t_m$.

Then, is necessary to show that $n_2(P) > n_2(P_1)$ to obtain $m(C) >_{lex} m(C_k)$. By Definition 4.7, we have that $n_2(P) = n_2(P') + n + m$. On the other hand, the value of $n_2(P_k)$ depends on k,

- if $k = 1$ then $n_2(P_1) = n_2(P') + (n-1) + (m-1) < n_2(P)$,
- if $1 < k < n-1$ then $n_2(P_k) = n_2(P') + n + (m-1) < n_2(P)$,
- if $k = n-1$ then $n_2(P_{n-1}) = n_2(P') + (n-1) + (m-1) < n_2(P)$.

Therefore, $m(C) >_{lex} m(C_k)$ for every $1 \leq k \leq n-1$, which implies that $M_1 >_{mul} M_2$.

Case *(R)* = *(A-Right)*:

This case is analogous as the case above.

□

Lemma 4.1 (Lemmas 17 and 18 in [1].)**.** Let $\mathcal{A}_A\langle s,t\rangle$ be an AUP$_A$. If

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P \mid S \mid \sigma\rangle$$

then

1. if $P = P' \cup \{y : u \triangleq v\}$, then y does not appear in any constraint in P or S,
2. $\texttt{Index}(P\cup S) \subseteq \texttt{vran}(\sigma)\cup\{x\}$ and $\texttt{vran}(\sigma) = \mathcal{V}(x\sigma)$.

Proof. The complete proof is in Appendix C, Lemmas C.1 and C.2 □

The next results: Lemma 4.2, Lemma 4.3 and Lemma 4.4 are used to adapt the Lemma 19 of [1].

Remark 4.3 (Counter example of Lemma 19 of [1].). In [1], the Lemma 19 says: "Given flattened terms t and t' such that every symbol in t and t' is either free or associative, and a fresh variable x, then there is a sequence

$$\langle\{x : t \triangleq t'\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$$

such that there is no variable z such that $\{z : u \triangleq v\} \in S$ if and only if (u,v) is an associative pair of subterms of t and t'."
Notice that for $t = h(a,b,c,d)$ and $t' = h(a',b',c',d')$, it follows $(h(a,b),h(a',b'))$ is an associative pair of positions of t and t', but using the rules in Figure 4.1 there is no sequence starting from $\langle\{y : t \triangleq t'\} \mid \emptyset \mid id\rangle$ such that the configuration

$$\langle P \cup \{y : h(a,b) \triangleq h(a',b')\} \mid S \mid \sigma\rangle$$

appears. This contradicts this lemma. For more details about this example, see Section C.2.1 where we give a full derivation for $\texttt{AUnif}_A(s,t)$.
In [1], this claim is used to prove completeness. Since it does not hold, it trigger mistakes in the proof of completeness presented there. To prove the completeness of $\texttt{AUnif}_A$, we propose a correction where Lemma 19 of [1] is replaced for Lemmas 4.2, 4.3 and 4.4.

The next Lemma established that for every regular pair of subterms of s and t, i.e, for every associative pair of subterms (u,v) according with the Condition 1 of Definition 4.3, there exists a sequence $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$.

Lemma 4.2. Let $\mathcal{A}_A\langle s,t\rangle$ a AUP$_A$. If (p,p') is an associative pair of positions of s and t, then there exists a derivation of the form $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle\{y : u \triangleq v\} \mid S \mid \sigma\rangle$ with $(s|_p, t|_{p'}) = (u,v)$.

Proof. Notice that since (p,p') is an associative pair of subterms then $\texttt{depth}(p) = \texttt{depth}(p')$ by definition. The proof follows by induction over the depth d of p.

Base Case: If $\texttt{depth}(p) = 0$, then $p = p' = \varepsilon$, $(u,v) = (s,t)$ and the result follows trivially for the initial configuration $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$.

Inductive Step: If $\texttt{depth}(p) = d+1$, then p and p' are immediately under some positions $q \in \texttt{pos}(s)$ and $q' \in \texttt{pos}(t)$, such that $p = q.i$ and $p' = q'.j$, for some natural numbers i and j. Notice that (p,p') being an associative pair of positions of s and t yields that (q,q') is an associative pair of positions too. On the other hand, $\texttt{depth}(q) = d$. And, the induction hypothesis implies that there exists a derivation such that

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P' \cup \{z : s|_q \triangleq t|_{q'}\} \mid S' \mid \sigma'\rangle. \tag{1}$$

The proof follows by analysis $\texttt{root}(s|_q)$ and $\texttt{root}(t|_{q'})$. Notice that (p,p') being an associative pair of positions implies that $\texttt{root}(s|_q) = \texttt{root}(t|_{q'})$.

Case 1: $\texttt{root}(s|_q) = \texttt{root}(t|_{q'}) = f \in \Sigma_\emptyset$.

The proof is analogously to the proof of Lemma 2.3.

Case 2: $\texttt{root}(s|_q) = \texttt{root}(t|_{q'}) = h \in \Sigma_h$.

Then, $s|_q = h(\overline{u_n})$ and $t|_{q'} = h(\overline{v_m})$. It follows that (1) is of the form

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P_1 \cup \{z : h(\overline{u_n}) \triangleq h(\overline{v_m})\} \mid S' \mid \sigma'\rangle. \tag{2}$$

Since, $p = q.i$ and $p' = q'.j$ it follows that $u = u_i$ and $v = v_j$, i.e, u is the i-th argument of $h(\overline{u_m})$ and v is the j-th argument of $h(\overline{v_m})$. We want to show that exists a derivation of the form

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle,$$

For that we need to apply conveniently *(A-Dec)*-rules in (2). Notice that the definition of index of a position yields that $(p)_{d+1} = i$ and $(p')_{d+1} = j$. Thus, the proof proceeds by analysing of the index of p at depth $d+1$.

1. $(p)_{d+1} = 1$.
Then, by Definition 4.5 it follows that $(p')_{d+1} = 1$. Hence, $u = u_1$ and $v = v_1$ and the result follows by an application of *(A-Left)* in (2), as is showed below

$$\begin{array}{c}
\langle P_1 \cup \{z : h(\overline{u_n}) \triangleq h(\overline{u_m})\} \mid S' \mid \sigma' \rangle \\
\Big\Downarrow{\scriptstyle ((A\text{-}Left))} \\
\langle P_1 \cup \left\{ \begin{array}{l} y_1 : u_1 \triangleq t_1, \\ y_2 : h(u_2, \dots, u_n) \triangleq h(v_2, \dots, v_m) \end{array} \right\} \mid S \mid \sigma \rangle
\end{array}$$

with $P = P_1 \cup \{y_2 : h(u_2, \dots, u_n) \triangleq h(v_2, \dots, v_n)\}$, $S = S'$, $\sigma = \sigma'\{z \mapsto g(y_1, y_2)\}$ and $y = y_1$.

2. $1 < (p)_{d+1} < n$.
Thus, by Definition 4.5 it follows that $1 < (p')_{d+1} < m$. Hence, applying *(A-Dec)* rules as follows

$$\begin{array}{c}
\langle P_1 \cup \{z : h(\overline{u_n}) \triangleq \overline{v_m}\} \mid S' \mid \sigma' \rangle \\
\Big\Downarrow{\scriptstyle (A\text{-}Left)} \\
\langle P_1 \cup \left\{ \begin{array}{l} z_1 : h(u_1, \dots, u_{i-2}) \triangleq v_1, \\ z_2 : h(u_{i-1}, u_i, \dots, u_n) \triangleq h(v_2, \dots, v_m) \end{array} \right\} \mid S' \mid \underbrace{\sigma'\{z \mapsto h(z_1, z_2)\}}_{\sigma''} \rangle \\
\Big\Downarrow{\scriptstyle (A\text{-}Right)} \\
\langle P_2 \cup \left\{ \begin{array}{l} z_3 : u_{i-1} \triangleq h(v_2, \dots, v_{j-1}), \\ z_4 : h(u_i, \dots, u_n) \triangleq h(v_j, \dots, v_m) \end{array} \right\} \mid S' \mid \underbrace{\sigma''\{z_2 \mapsto h(z_1, z_4)\}}_{\sigma'''} \rangle \\
\Big\Downarrow{\scriptstyle (A\text{-}Left)} \\
\langle P_3 \cup \left\{ \begin{array}{l} w_1 : u_i \triangleq v_j, \\ w_2 : h(s_{i+1}, \dots, s_n) \triangleq h(t_{j+1}, \dots, t_m) \end{array} \right\} \mid S' \mid \sigma'''\{z_4 \mapsto h(w_1, w_2)\} \rangle
\end{array}$$

where

$$P_2 = P_1 \cup \{z_1 : h(u_1, \dots, u_{i-2}) \triangleq v_1\} \text{ and } P_3 = P_2 \cup \{z_3 : u_{i-1} \triangleq h(v_2, \dots, v_{j-1})\},$$

then result follows by taking $P = P_3 \cup \{w_2 : h(s_{i+1}, \dots, s_n) \triangleq h(t_{j+1}, \dots, t_m)\}$, $S = S'$, $\sigma = \sigma'''\{z_4 \mapsto h(w_1, w_2)\}$ and $y = w_1$.

3. $(p)_{d+1} = n$.
Notice that $n = \mathtt{arg}(h(\overline{u_n})) = \mathtt{arg}(s|_{(p)^d})$, then the Definition 4.5 implies that $(p')_{d+1} = m = \mathtt{arg}(h(\overline{v_m})) = \mathtt{arg}(t|_{(p')^d})$. Thus, $u = u_n$ and $v = v_m$ and

the result follows by two applications of *(A-Dec)* rules in (2) as is showed below

$$\begin{array}{c}
\langle P_1 \cup \{z : h(\overline{u_n}) \triangleq h(\overline{v_m})\} \mid S' \mid \sigma' \rangle \\
\Big\Downarrow{\scriptstyle \textit{(A-Left)}} \\
\langle P_1 \cup \left\{ \begin{array}{l} z_1 : h(u_1,\dots,u_{n-2}) \triangleq v_1, \\ z_2 : h(u_{n-1},u_n) \triangleq h(v_2,\dots,v_m) \end{array} \right\} \mid S' \mid \underbrace{\sigma' \{z \mapsto h(z_1,z_2)\}}_{\sigma''} \rangle \\
\Big\Downarrow{\scriptstyle \textit{(A-Right)}} \\
\langle P_1 \cup \left\{ \begin{array}{l} z_1 : h(u_1,\dots,u_{n-2}) \triangleq v_1, \\ w_1 : u_{n-1} \triangleq h(v_2,\dots,v_{m-1}) \\ w_2 : u_n \triangleq v_m, \end{array} \right\} \mid S' \mid \sigma''\{z_2 \mapsto h(w_1,w_2)\} \rangle
\end{array}$$

with $P = P_1 \cup \{z_1 : h(u_1,\dots,u_{n-2}) \triangleq v_1, w_2 : u_{n-1} \triangleq h(v_2,\dots,v_{m-1})\}$, $S = S'$, $\sigma = \sigma''\{z_2 \mapsto h(w_1,w_2)\}$ and $y = w_2$.

□

Let $\mathcal{A}_A\langle s,t\rangle$, the following lemma establishes the relation between $\mathtt{AUnif}_A(s,t)$ and the associative pairs of positions given by the Condition 2 of Definition 4.3.

Lemma 4.3. Let $\mathcal{A}_A\langle s,t\rangle$ be an AUP$_A$ and (p,p') an associative pair of positions of s and t, such that

$$s|_p = h(s_1,\dots,s_k,u_1,\dots,u_n,s_{k+1}\dots,s_q),$$

$$t|_{p'} = h(t_1,\dots,s_{k'},v_1,\dots,v_m,t_{k'+1},\dots,s_{q'}).$$

If $(u,v) = (h(\overline{u_n}),h(\overline{v_m}))$ is an associative pair of subterms, then there exists derivations such that

1. $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \cup \{y : h(u_1,\dots,u_i) \triangleq v_1\} \mid S \mid \sigma\rangle$ with $1 \le i \le n-1$, and
2. $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \cup \{y : u_1 \triangleq h(v_1,\dots,v_j)\} \mid S \mid \sigma\rangle$ with $1 < j \le m-1$.

Proof. The Lemma 4.3 implies that there exists a sequence such that

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P' \cup \{z : s|_p \triangleq t|_{p'}\} \mid S \mid \sigma'\rangle$$

First we apply *(A-Dec)* rules as follows:

$$\langle P' \cup \{y : s|_p \triangleq t|_{p'}\} \mid S \mid \sigma' \rangle$$

$$\Big\Downarrow \text{\textit{(A-Right)}}$$

$$\langle P' \cup \left\{ \begin{array}{l} z_1 : s_1 \triangleq h(t_1,\dots,t_{k'-1}), \\ z_2 : h(s_2,\dots,s_q) \triangleq h(t_{k'},v_1,\dots,v_m,t_{k'+1},\dots,t_{q'}) \end{array} \right\} \mid S \mid \underbrace{\sigma'\{z \mapsto h(z_1,z_2)\}}_{\sigma''} \rangle$$

$$\Big\Downarrow \text{\textit{(A-Left)}}$$

$$C = \langle P' \cup \left\{ \begin{array}{l} z_1 : s_1 \triangleq h(t_1,\dots,t_{k'-1}), \\ z_3 : h(s_2,\dots,s_k) \triangleq t_{k'}, \\ z_4 : h(u_1,\dots,u_n,s_{k+1},\dots,s_q) \triangleq h(v_1,\dots,v_m,t_{k'+1},\dots t_{q'}) \end{array} \right\} \mid S \mid \sigma_1 \rangle$$

where $\sigma_1 = \sigma''\{z_2 \mapsto h(z_3,z_4)\}$. Taking $P_1 = \{z_1 : s_1 \triangleq h(t_1,\dots,t_{k-1}), z_3 : h(s_2,\dots,s_k) \triangleq t_{k'}\}$, it follows that

$$C = \langle P_1 \cup \{z_4 : h(u_1,\dots,u_n,s_{k+1},\dots,s_q) \triangleq h(v_1,\dots,v_m,t_{k'+1},\dots t_{q'})\} \mid S \mid \sigma_1 \rangle$$

1. by one application of *(A-Left)* in the configuration C, we obtain

$$C \Longrightarrow_{\text{\textit{(A-Left)}}} \langle P_1 \cup \left\{ \begin{array}{l} y_1 : h(u_1,\dots,u_i) \triangleq v_1, \\ y_2 : h(u_{i+1},\dots,u_n,s_{k+1},\dots,s_q) \triangleq h(v_2,\dots,v_m,t_{k'+1},\dots,t_{q'}) \end{array} \right\} \mid S \mid \sigma' \rangle$$

   with $1 \le i \le n-1$ depending on how *(A-Left)* was applied. Thus, the result follows when taking $P = P_1 \cup \{y_2 : h(u_{i+1},\dots,u_n,s_{k+1},\dots,s_q) \triangleq h(v_2,\dots,v_m,t_{k'+1},\dots,t_{q'})\}$, $\sigma = \sigma_1\{z_4 \mapsto h(y_1,y_2)\} = \sigma'$ and $y = y_1$.

2. by one application of *(A-Right)* rule in the configuration C, we obtain that

$$C \Longrightarrow_{\text{\textit{(A-Right)}}} \langle P_1 \cup \left\{ \begin{array}{l} y_1 : u_1 \triangleq h(v_1,\dots,v_i), \\ y_2 : h(u_2,\dots,u_n,s_{k+1},\dots,s_q) \triangleq h(v_{i+1},\dots,v_m,t_{k'+1},\dots,t_{q'}) \end{array} \right\} \mid S \mid \sigma' \rangle$$

   with $1 < j \le m-1$, depending on how *(A-Right)* was applied. Thus, the result follows when taking $P = P_1 \cup \{y_2 : h(u_2,\dots,u_n,s_{k+1},\dots,s_q) \triangleq h(v_{i+1},\dots,v_m,t_{k'+1},\dots,t_{q'})\}$, $\sigma = \sigma_1\{z_4 \mapsto h(y_1,y_2)\} = \sigma'$ and $y = y_1$.

$\square$

Let $\mathcal{A}_A\langle s,t\rangle$ be an AUP$_A$, the next lemma states that every constraint $\{y : u \triangleq v\}$ produced by one step of application of the rules in Figure 4.1 from $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ is such that (u,v) is an associative pair of subterms of s and t.

Lemma 4.4. Let $\mathcal{A}_A\langle s,t\rangle$ be an AUP$_A$. If there exists a sequence of the form

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$$

then (u,v) is an associative pair of subterms of s and t.

Proof. The proof is by induction on the length n of the reduction

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n}{\Longrightarrow}_{\texttt{AUnif}_A} \langle\{y : u \triangleq v\} \cup P \mid S \mid \sigma\rangle$$

Base Case: If $n = 0$ then $\langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle = \langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ and the result follows trivially.

Inductive Step: If $n > 0$, then the derivation unfolds as

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P' \mid S' \mid \sigma'\rangle \Longrightarrow_{(R)} \langle\{y : u \triangleq v\} \cup P \mid S \mid \sigma\rangle$$

The proof proceeds by analysis of each inference rule of `AUnif`$_A$, denoted by (R), used in n-th of the derivation. The cases which (R) is (Dec), (Sol) of (Rec) were already verified in the proof of Lemma 2.3. Thus, it remains verify the cases which (R) is *(A-Left)* or *(A-Right)*.

Case *(R)* = *(A-Left)*:

Thus, the n-th step of the reduction if as follows, for some $s(s_1,\dots,s_n)$ and $t(t_1,\dots,t_n)$.

$$\begin{array}{c}\langle P_1 \cup \{z : h(\overline{u_n}) \triangleq h(\overline{t_m})\} \mid S \mid \sigma'\rangle \\ \Big\Downarrow{\scriptstyle ((A\text{-}Left))} \\ \langle P_1 \cup \left\{\begin{array}{l} y_1 : h(u_1,\dots,u_k) \triangleq v_1 \\ y_2 : h(u_{k+1},\dots,u_n) \triangleq h(t_2,\dots,v_m)\end{array}\right\} \mid S \mid \sigma\rangle\end{array}$$

Case 1: if $y \neq y_1, y_2$, then $\{y : u \triangleq v\} \in P_1$ and the result follows by the induction hypothesis,

Case 2: if $y = y_1$, then $\{y : u \triangleq v\} = \{y_1 : h(s_1,\dots,s_k) \triangleq t_1\}$, with $1 \leq k \leq n-1$. By induction hypothesis in the configuration it follows that $(h(\overline{u_n}), h(\overline{v_m}))$ is an associative pair of subterms of s and t. There are two cases, depending on what condition this pair of subterms attend in Definition 4.3.

1. If $(h(\overline{u_n}), h(\overline{v_m}))$ is an associative pair of subterms according to Condition 1 of Definition 4.3, i.e, if there are positions $p \in \texttt{pos}(s)$ and $p \in \texttt{pos}(t)$ such that $s|_p = h(\overline{u_n})$ and $t|_{p'} = h(\overline{v_m})$. Then, $(u,v) = (h(u_1,\dots,u_k), v_1)$ is an associative pair of subterms of s and t by Condition 2 of Definition 4.3.
2. If (u,v) is an associative pair of subterms according to Condition 2 of Definition 4.3, i.e, if there exists positions $p \in \texttt{pos}(s)$ and $p' \in \texttt{pos}(t)$ such that
$$s|_p = h(s_1,\dots,s_l,u_1,\dots,u_n,s|_{l+1}\dots,s|_q)$$
$$t|_p = h(t_1,\dots,t|_{l'},v_1,\dots,v_m,t|_{l'+1},\dots,t|_{q'})$$
then, $(u,v) = (h(u_1,\dots,u_k), v_1)$ is an associative pair of subterms for every $1 \le k \le n$.

Case 3: If $y = y_2$, then $\{y : u \triangleq v\} = \{y_2 : h(u_{k+1},\dots,u_n) \triangleq h(v_2,\dots,v_m)\}$. Then it is necessary to show that $(h(u_{k+1},\dots,u_n)h(v_2,\dots,v_m))$. The argument is similar to the previous case.

Case *(R) = (A-Right)*:

The proof is analogous as the case above.

□

Let $\texttt{AUnif}_A$ be an AUP$_A$ $\mathcal{A}_A\langle s,t\rangle$. Notice that the *(A-Dec)* rules will produce substitutions σ such that $x\sigma$ will be in non-flatted form. However, each argument of the flatted subterm that should being compared to solve $\mathcal{A}_A\langle s,t\rangle$ is obtained from the derivation of $\texttt{AUnif}_A(s,t)$ as it was shown in Lemma 4.2. On the other hand Lemma 4.3 show how to obtain these non-flattened forms as we apply $\texttt{AUnif}_A$ rules in the initial configuration $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$.

Lemma 4.5 (Lemma 20 in [1])**.** Given flattened terms s,t of terms in $T(\mathcal{X},\Sigma_{\emptyset\cup A})$ then there exists a sequence $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P \mid \{y : u \triangleq v\} \cup S \mid \sigma\rangle$ if, and only if (u,v) is an associative conflict pair of s and t.

Proof. The complete proof can be found in Appendix C, Lemma C.3. □

As previous mentioned, *(A-Dec)* rules will produce substitutions σ such that $x\sigma$ will be in non-flattened form. In fact, let $\mathcal{A}_A\langle s,t\rangle$ be an AUP$_A$, as we apply the rules of Figure 4.1 in the initial configuration $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ we will obtain substitutions $\sigma_1,\dots,\sigma_n$ such

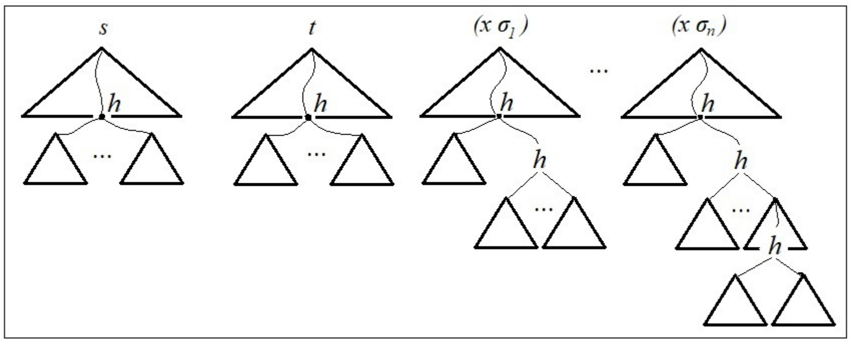


Fig. 4.3 $x\sigma_i$ are becoming less flattened as the value of i grows.

that the terms $(x\sigma_1),\ldots,(x\sigma_n)$ will become closer to the non-flattened form, as it is shown in Figure 4.3. The following lemma establishes the relation between the non flattened terms s and t and the terms $x\sigma$ given by some configuration $\langle P \mid S \mid \sigma\rangle$ obtained by $\mathtt{AUnif}_A(s,t)$. To prove it, we will use the notion of replacing a subterm of a term by another term in a given position p; such notion was established in Definition 1.4.

Lemma 4.6. Let $\mathcal{A}_A\langle s,t\rangle$ be an AUP$_A$ and s' and t' be terms such that s and t are their respective flattened forms. If there exists a derivation of the form

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \cup \{x : u \triangleq v\} \mid S \mid \sigma\rangle,$$

then there exist non-flattened terms $s'' \equiv_A s'$ and $t'' \equiv_A t'$ for which there exist positions $p \in \mathtt{pos}(s'')$ and $p' \in \mathtt{pos}(t'')$ such that

$$(x\sigma)|_p = y,\ s''|_p = (x\sigma)|_p\{y \mapsto u'\} = u' \text{ and } t''|_{p'} = (x\sigma)|_p\{y \mapsto v'\} = v'$$

where u' and v' are unflattened versions of u and v respectively.

Proof. By induction on the lenght n of the reduction

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle.$$

Induction Base: If $n = 0$ then $\langle x : s \triangleq t \mid \emptyset \mid id\rangle = \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$, $P = \emptyset$, $u = s$, $v = t$ and $\sigma = id$. Since s' and t' are the unflattened versions of s and t, thus the result follows by taking $p = p' = \varepsilon$, obtaining that $(xid)|_\varepsilon = x$, $s|_\varepsilon = x\{x \mapsto s\}$ and $t|_\varepsilon = x\{x \mapsto t\}$.

Inductive Step: Suppose that $n > 0$ and consider the reduction

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P' \mid S' \mid \sigma'\rangle \Longrightarrow_{(R)} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle.$$

We want to show that there exist non-flattened terms s'' and t'' such that $s'' \equiv_A s'$ and $t'' \equiv_A t'$, positions $p \in \mathtt{pos}(s'')$ and $p' \in \mathtt{pos}(t'')$ such that

$$(x\sigma)|_p = y,\ s''|_p = (x\sigma)|_p\{y \mapsto u'\} = u' \text{ and } t''|_{p'} = (x\sigma)|_p\{y \mapsto v'\} = v',$$

where u' and v' are unflattened versions of u and v respectively.

The proof follows by analysis of what rule (R) of Figure 4.1 was applied in the n-th step of the derivation. The cases where (R) is (Dec), (Rec) or (Sol) were already analysed during the proof of Lemma 2.5. Thus, it remains to analyzing the cases where (R) is one of the *(A-Dec)* rules, i.e, the cases where $(R) =$ *(A-Left)* or $(R) =$ *(A-Right)*.

Case *(R)=(A-Left)*:

Thus, the reduction is of the form

$$\begin{aligned}\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\overset{n-1}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P_1 \cup \{z : h(\overline{u_n}) \triangleq h(\overline{v_m})\} \mid S \mid \sigma'\rangle \\ &\Longrightarrow_{\textit{(A-Left)}} \langle P_1 \cup \left\{\begin{matrix} y_1 : h(u_1,\dots,u_k) \triangleq v_1 \\ y_2 : h(u_{k+1},\dots,u_n) \triangleq h(v_2,\dots,v_m)\end{matrix}\right\} \mid S \mid \sigma\rangle.\end{aligned}$$

Where

$$\begin{aligned}P' &= P_1 \cup \{z : h(\overline{u_n}) \triangleq h(\overline{v_m})\}, \\ P &= P_1 \cup \{y_1 : h(u_1,\dots,u_k) \triangleq v_1, y_2 : h(u_{k+1},\dots,u_n) \triangleq h(v_2,\dots,v_m)\}, \\ S &= S' \text{ and } \sigma = \sigma'\{z \mapsto h(y_1,y_2)\}.\end{aligned}$$

There are three cases to be analysed, depending on the form of y.

1. If $y \neq y_1, y_2$.
   Then, $\{y : u \triangleq v\} \in P_1$ and the result follows by induction hypothesis.
2. If $y = y_1$.
   Then, $\{y : u \triangleq v\} = \{y_1 : h(u_1,\dots,u_k) \triangleq v_1\}$. By induction hypothesis in the $(n-1)$-th step of the reduction, it follows that there exist terms s_1 and t_1 such that $s_1 \equiv_A s'$ and $t_1 \equiv_A t'$ and positions $q \in \mathtt{pos}(s_1)$ and $q' \in \mathtt{pos}(t_2)$ such that $(x\sigma)|_p = z$, $s_1|_q = u''$ is a unflattened version of $h(\overline{u_n})$, and $t_1|_{q'} = v''$ is a unflattened version of $h(\overline{v_m})$. Thus,

$$\begin{aligned}&u'' \equiv_A h(h(u_1,h(u_2,\dots h(u_{k-2},h(u_{k-1},u_k))\dots),h(u_{k+1},h(\dots,h(u_{n-1},u_n)\dots))), \\ &v'' \equiv_A h(v_1,h(v_2,h(\dots,h(v_{m-1},v_m))\dots)\end{aligned}$$

Taking $s'' = s_1[u'']_q$ and $t'' = t_1[v'']_{q'}$, it follows that $s'' \equiv_A s_1$ and $t'' \equiv_A t_1$. Therefore, $s' \equiv_A s''$ and $t' \equiv_A t''$. Furthermore, by taking

$$u' = u''|_1 = h(u_1, h(u_2, \dots h(u_{k-2}, h(u_{k-1}, u_k))\dots) = u_1, \text{ and}$$
$$v' = v''|_1 = v_1$$

we have that the positions $p = q.1 \in \mathtt{pos}(s'')$ and $q = q'.1 \in \mathtt{pos}(t'')$ are such that $s''|_p = u''|_1 = u'$ and $t''|_{p'} = v''|_1 = v'$ are unflattened versions of $u = h(\overline{u_n})$ and $v = v_1$ respectively.

3. if $y = y_2$.
Then, $\{y : u \triangleq v\} = \{y_2 : h(u_{k+1}, \dots, u_n) \triangleq h(v_2, \dots, v_m)\}$. The result by an analogously argument, just taking $p = q.2 \in \mathtt{pos}(s'')$ and $p' = q'.2 \in \mathtt{pos}(t'')$.

Case *(R) = (A-Right)*:

The proof is analogously as the previous case.

□

In the following we will show that $\mathtt{AUnif}_A$ is sound, i.e., that every step of a derivation $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \mid S \mid \sigma\rangle$ gives a term $x\sigma$ that is a generalizer of s and t.

Theorem 4.2 (Soundness of $\mathtt{AUnif}_A$ [1]). Let $\mathcal{A}_A\langle s,t\rangle$ be an AUP$_A$. If there exists a derivation of the form $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A}, \langle P \mid S \mid \sigma\rangle$ then $x\sigma \in \mathtt{gen}_A(s,t)$.

Proof. By induction on the length n of the reduction $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \mid S \mid \sigma\rangle$.

Base Case: If $n = 0$, then $\langle P \mid S \mid \sigma\rangle = \langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ and $xid = x \in \mathtt{gen}_A(s,t)$ and x is the trivial generalizer of s and t modulo A.

Inductive Step: Suppose the result holds for derivations of length $n-1$, i.e, derivations of the form, $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P' \mid S' \mid \sigma'\rangle \Longrightarrow \langle P \mid S \mid \sigma\rangle$. That is, $x\sigma' \in \mathtt{gen}_A(s,t)$, where $\sigma = \sigma'\delta$ for some δ.

The proof proceeds by analysis the rule (R) applied in the n-th step of the derivation. The cases in which (R) is (Dec), (Sol) or (Rec) were already verified in the proof of Lemma 2.6. It remains to verify the case in which (R) is *(A-Left)* or *(A-Right)*.

Case *(R) = (A-Left)*: Thus, the reduction is of the form:

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P_1 \cup \{y : h(\overline{u_n}) \triangleq h(\overline{v_m})\} \mid S' \mid \sigma'\rangle$$
$$\Longrightarrow_{\textit{(A-Left)}} \langle P_1 \cup \left\{\begin{matrix} y_1 : h(u_1, \dots, u_k) \triangleq v_1, \\ y_2 : h(u_{k+1}, \dots, u_n) \triangleq h(v_2, \dots, v_m) \end{matrix}\right\} \mid S \mid \sigma\rangle.$$

From now, take

$$
\begin{aligned}
P' &= P_1 \cup \{y : h(\overline{u_n}) \triangleq h(\overline{v_m})\}\\
P &= P_1 \cup \{y_1 : h(u_1,\dots,u_k) \triangleq v_1, y_2 : h(u_{k+1},\dots,u_n) \triangleq h(v_2,\dots,v_m)\},
\end{aligned}
$$

with $1 \leq k \leq n-1$. In any case, $S = S'$ and $\sigma = \sigma'\{y \mapsto h(y_1,y_2)\} = \sigma'\delta$.

We we will show that $x\sigma$ generalizers an unfllatened version of s and t, which will imply that $x\sigma \in \mathtt{gen}_A(s,t)$ by Remark 4.1.

By Lemma 4.6, it follows that are s'' and t'' that there are unflattened versions of s and t for such that there are positions $p \in \mathtt{pos}(s'')$, $p' \in \mathtt{pos}(t'')$ such that $(x\sigma)|_p = y$, $s''|_p = y\{y \mapsto u'\} = u'$ is an unflatenned version of u and $t''|_{p'} = y\{y \mapsto v'\} = v'$ is an unflattened version of v.

By the proof of Lemma 4.6, u' is such that $u'|_1 = u'_1$ is an unflateened version of $h(\overline{u_k})$ and $u'|_2 = u'_2$ is an unflattened version of $h(u_{k+1},\dots,u_n)$. Similarly, $v'|_1 = v'_1$ and $v'|_2 = v'_2$ are unflattened versions v_1 and $h(v_2,\dots,v_m)$ respectively.

By induction hypothesis it follows that $(x\sigma') \in \mathtt{gen}_A(s,t)$. Then, there exists a pair of substitutions $\overline{\theta} = (\theta_1,\theta_2)$ such that $(x\sigma')\theta_1 \equiv_A s''$ and $(x\sigma')\theta_2 \equiv_A t''$. Therefore, $y\theta_1 = h(u'_1,u'_2)$ and $y\theta_2 = h(v'_1,v'_2)$.

Thus, taking θ'_1 as the substitution θ_1 with domain restricted to $(\mathcal{X} - \{y\})$, it follows that the substitution θ_1 can be unfolded as

$$
\begin{aligned}
(x\sigma')\theta_1 &= (x\sigma')\{y \mapsto h(u'_1,u'_2)\}\theta_1\\
&= (x\sigma')\{y \mapsto h(y_1,y_2)\}\{y_1 \mapsto u'_1, y_2 \mapsto u'_2\}\theta'_1\\
&= (x\sigma)\delta\{y_1 \mapsto u'_1, y_2 \mapsto u'_2\}\theta'_1.
\end{aligned}
$$

Similarly, $(x\sigma')\theta_2 = (x\sigma)\delta\{y_1 \mapsto v'_1, y_2 \mapsto v'_2\}\theta'_2$, where θ'_2 is the substitution θ_2 with domain restricted to the domain restricted to $(\mathcal{X} - \{y\})$.

Taking the pair of substitutions $\overline{\tau} = (\tau_1,\tau_2)$ given as follows:

$$
\tau_1 = \left\{\begin{matrix} x \mapsto x\theta'_1, \text{ if } x \in \mathtt{dom}(\theta'_1),\\ y_1 \mapsto u'_1,\\ y_2 \mapsto u'_2 \end{matrix}\right\}, \quad \tau_2 = \left\{\begin{matrix} x \mapsto x\theta'_2, \text{ if } x \in \mathtt{dom}(\theta'_2),\\ y_1 \mapsto v'_2,\\ y_2 \mapsto v'_1 \end{matrix}\right\}
$$

Notice that for the freshness of y_1, y_2 it follows that $y_1, y_2 \notin \mathtt{dom}(\theta_1') \cup \mathtt{dom}(\theta_2')$ and $y_1, y_2 \notin \mathcal{V}(\theta_1') \cup \mathcal{V}(\theta_2')$. Hence,

$$\overline{\theta} = (\theta_1, \theta_2) = (\{y_1 \mapsto u_1', y_2 \mapsto u_2'\}\theta_1', \{y_1 \mapsto v_1', y_2 \mapsto v_2'\}\theta_2').$$

Thus, $(x\sigma)\overline{\tau} = (x\sigma')\delta\overline{\tau} = \overline{x\sigma\overline{\theta}} = (s'', t'')$. Then, $(x\sigma)$ is a generalizer of s'' and t'', that are terms which s and t are their flattened forms. Therefore, $x\sigma \in \mathtt{gen}_A(s,t)$.

□

The following lemma is an auxiliary result to show that $\mathtt{AUnif}_A$ is complete. This lemma relates each generalizer of s and t with an expression $x\sigma$ given by some derivation $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \mid S \mid \sigma\rangle$ of $\mathtt{AUnif}_A(s,t)$.

Lemma 4.7 (Lemma 21 in [1]). Let $\mathcal{A}_A\langle s,t\rangle$ be AUP$_A$. If $u \in \mathtt{gen}_A(s,t)$ then there exist a sequence $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \mid S \mid \sigma\rangle$ such that $u \equiv_A x\sigma$.

Proof. The complete proof can be found in Appendix C, Lemma C.4. □

Theorem 4.3 (Completeness of $\mathtt{AUnif}_A$ [1]). Let $\mathcal{A}_A\langle s,t\rangle$ a AUP$_A$. If $r \in \mathtt{lgg}_A(s,t)$ then there exists a derivation of the form $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \mid S \mid \sigma\rangle$ such that $r \equiv_A x\sigma$.

Proof. If $r \in \mathtt{lgg}_C(s,t)$ it is clear that $r \in \mathtt{gen}_C(s,t)$, and then Lemma 4.7 yields that there exists a derivation $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \mid S \mid \sigma'\rangle$ such that $r \equiv_A x\sigma$.

Suppose by contradiction that $P = P_1 \cup \{y : u \triangleq v\}$, i.e, $P \neq \emptyset$.

The cases where $\mathtt{root}(u) \neq \mathtt{root}(v)$ and $\mathtt{root}(u) = \mathtt{root}(v) = f \in \Sigma$ were already analysed in the prove of Theorem 3.3. Then, only the case which $\mathtt{root}(u) = \mathtt{root}(v) = h \in \Sigma_A$ remains to be analysed.

Case: $\mathtt{root}(u) = \mathtt{root}(v) = h$ is an associative function symbol remains to be analysed.

Thus, $u = h(s_1, \ldots, u_n)$ and $v = h(v_1, \ldots, v_m)$, hence

$$\begin{tikzcd}
 & \langle P' \cup \{y : h(\overline{u_n}) \triangleq h(\overline{v_m})\} \mid S \mid \sigma'\rangle \arrow[dl, Rightarrow, "(A\text{-}Left)"'] \arrow[dr, Rightarrow, "(A\text{-}Right)"] & \\
\langle\left\{\begin{array}{l} y_1 : h(u_1, \ldots, u_k) \triangleq v_1, \\ y_2 : h(u_{k+1}, \ldots, u_n) \triangleq h(v_2, \ldots, v_m) \end{array}\right\} \mid S \mid \sigma\rangle & & \langle\left\{\begin{array}{l} y_1 : u_1 \triangleq h(v_1, \ldots, v_{k'}), \\ y_2 : h(u_2, \ldots, u_n) \triangleq h(v_{k'+1}, \ldots, v_m) \end{array}\right\} \mid S \mid \sigma\rangle
\end{tikzcd}$$

with $1 \leq k < n$ and $1 < k' < m$. In every case, $\sigma = \{y \mapsto h(y_1, y_2)\}$ and $r = x\sigma' <_A x\sigma$, and the result follows.

□

A note on the finiteness of $\mathtt{lgg}_A(s,t)$.

Given an AUP$_A$, $\mathcal{A}_A\langle s,t\rangle$, each step of $\mathtt{AUnif}_A$ occurs by an application of a rule of Figure 4.1 in a unsolved constraint. Also, by Remark 2.2, just one rule can be applied in each step. Therefore, it follows that given a derivation of the form $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \mid S \mid \sigma\rangle$,

- if $(Dec), (Sol)$ or (Rec) applies in $\langle P \mid S \mid \sigma\rangle$ then this configuration will have just one direct successor;
- if the *(A-Dec)* applies in a constraint $\{y : h(\overline{u_n}) \triangleq h(\overline{v_m})\} \in P$, then *(A-Left)* will produce $n-1$ directly successors and *(A-Right)* $m-2$ ones. Thus, in this case $\langle P \mid S \mid \sigma\rangle$ will have $(n+m)-3$ directly successors;

Therefore, $\mathtt{AUnif}_A$ is finitely branching. Since the Theorem 4.1 yields that $\mathtt{AUnif}_A$ always terminates, then by Lemma 1.5 it follows that $\mathtt{AUnif}_A$ is globally finite, i.e, each configuration $\langle P \mid S \mid \sigma\rangle$ contains a finite number of successors.

Thus, the initial configuration of $\mathtt{AUnif}_A$ will have a finite number of successors which are the normal forms w.r.t $\Longrightarrow_{\mathtt{AUnif}_A}$. By Theorem 4.2 will be enough to build a finite, complete and minimal set of A-least general generalizers, $\mathtt{lgg}_A(s,t)$.

However, this construction is not immediately by the A-anti-unification algorithm $\mathtt{AUnif}_A$ because it may return normal forms that are repeated generalizers equal modulo associativity.

Chapter 5

Conclusion and Future Work

In this dissertation we have presented a detailed study of the Syntactic Anti-Unification Problem (AUP) and the Equational Anti-Unification Problems AUP_C and AUP_A.

First, we presented a rule-based algorithm for the Syntactic Anti-Unification Problem as in [1], called $\mathtt{AUnif}_\emptyset$, and verified its main properties, such as termination, confluence and correctness. From correctness we could explore the set of solutions of the problem, concluding that the AUP always have a solution and it is unique except for variable renaming.

Second, we have presented an extension of the $\mathtt{AUnif}_\emptyset$ to solve the AUP_C as in [1], obtaining the algorithm $\mathtt{AUnif}_C$, as well as detailed proofs of its main properties, such as, termination, soundness and completeness. From completeness we verified that AUP_C is finitary, i.e, it always have a finite and minimal set of solutions. However, the $\mathtt{AUnif}_C$ is only able to give complete set of C-generalizers but not minimal ones.

Third, similarly as C, we have presented an extension of the $\mathtt{AUnif}_\emptyset$ to solve the AUP_A, obtaining the algorithm $\mathtt{AUnif}_A$, we verified its main properties, such as, termination, soundness and completeness. In special, we have fixed inaccuracies on the proof of completeness in [1] by replacing the Lemma 19 in [1] for three new lemmas that were used to proved the property. Again, from the completeness we verified that AUP_A is finitary. However, as the commutative case, the $\mathtt{AUnif}_A$ is only able to give complete set of A-generalizers but not minimal ones.

In the following, we listed the two main lines of research that we plan to follow as future work.

1. Since $\mathtt{AUnif}_C$ and $\mathtt{AUnif}_A$ have a finite number of normal forms and we have not presented a bound of these number, we want to obtain measure that gives a maximal bound of the number of normal forms obtained by $\Longrightarrow_{\mathtt{AUnif}_C}$ and $\Longrightarrow_{\mathtt{AUnif}_A}$.

2. To extend the study of AUP$_C$ and AUP$_A$ in languages with binders such as in the Nominal framework. More precisely, our goal is to extend the algorithm proposed in [5] to Nominal Syntax when Associative or Commutative theories are involved, like we had extended $\mathtt{AUnif}_\emptyset$ to deal with AUP$_A$ and AUP$_C$ in this dissertation.

References

[1] Alpuente, M., Escobar, S., Espert, J., and Meseguer, J. (2014). A modular order-sorted equational generalization algorithm. *Inf. Comput.*, 235:98–136.

[2] Alpuente, M., Escobar, S., Meseguer, J., and Sapina, J. (2022). Order-sorted equational generalization algorithm revisited. *Annals of Mathematics and Artificial Intelligence*, 90(5):499–522.

[3] Baader, F. (1991). Unification, weak unification, upper bound, lower bound, and generalization problems. In Book, R. V., editor, *Rewriting Techniques and Applications, 4th International Conference, RTA-91, Como, Italy, April 10-12, 1991, Proceedings*, volume 488 of *Lecture Notes in Computer Science*, pages 86–97. Springer.

[4] Baader, F. and Nipkow, T. (1998). *Term rewriting and all that.* Cambridge University Press.

[5] Baumgartner, A., Kutsia, T., Levy, J., and Villaret, M. (2015). Nominal Anti-Unification. In Fernández, M., editor, *26th International Conference on Rewriting Techniques and Applications (RTA 2015)*, volume 36 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 57–73, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[6] Bulychev, P. E., Kostylev, E. V., and Zakharov, V. A. (2009). Anti-unification algorithms and their applications in program analysis. In *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*, pages 413–423. Springer.

[7] Cerna, D. M. and Kutsia, T. (2020). Unital anti-unification: Type and algorithms. In *5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020).* Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[8] Mehta, S., Bhagwan, R., Kumar, R., Bansal, C., Maddila, C., Ashok, B., Asthana, S., Bird, C., and Kumar, A. (2020). Rex: Preventing bugs and misconfiguration in large services using correlated change analysis. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 435–448.

[9] Muggleton, S. (1999). Inductive logic programming: issues, results and the challenge of learning language in logic. *Artificial Intelligence*, 114(1-2):283–296.

[10] Plotkin, G. D. (1970). A note on inductive generalization. *Machine intelligence*, 5:153–163.

[11] Popplestone, R. (1970). An experiment in automatic induction. *Machine Intelligence*, 5:203–215.

[12] Reynolds, J. C. (1970). Transformational systems and algebraic structure of atomic formulas. *Machine intelligence*, 5:135–151.

[13] Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, 12(1):23–41.

[14] Schmid, U. and Wysotzki, F. (2000). Applying inductive program synthesis to macro learning. In *AIPS*, pages 371–378.

Appendix A

Appendix of Chapter 2

A.1 Proofs of Section 2.2

Lemma A.1 (Uniqueness of generalization variables [1]). Let $\mathcal{A}\langle s,t\rangle$ be an AUP. If there is a $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$ then y does not appear in any constraint in P or S.

Proof. By induction on the length n of the derivation

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$$

Base case: If $n > 0$ then $P = \{x : s \triangleq t\}$ and $S = \emptyset$ and the conclusion follows trivially.

Inductive Step: If $n > 0$ then the derivation unfolds as

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P' \mid S \mid \sigma\rangle \Longrightarrow_{(R)} \langle P \mid S \mid \sigma\rangle$$

The proof follows by analyses of the inference rule (R) applied in the n-th step of the reduction.

Case *(R) = (Dec)*:

Then the derivation is of the form

$$\begin{aligned}\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P_1 \cup \{z : f(\overline{u_m}) \triangleq f(\overline{v_m})\} \mid S \mid \sigma'\rangle \\ &\Longrightarrow_{(Dec)} \langle P_1 \cup \{y_1 : u_1 \triangleq v_1, \ldots, y_n : u_n \triangleq v_n\} \mid S \mid \sigma'\{z \mapsto f(\overline{y_m})\}\rangle\end{aligned}$$

where,

$$P' = P_1 \cup \{z : f(\overline{u_m}) \triangleq f(\overline{v_m})\},\ P = P_1 \cup \{y_1 : u_1 \triangleq v_1, \ldots, y_n : u_n \triangleq v_n\},$$
$$S = S' \text{ and } \sigma = \sigma'\{z \mapsto f(\overline{y_m})\}$$

Notice that if $y \neq y_j$ for all $j = 1, \ldots, m$, then $\{y : u \triangleq v\} \in P_1$ and the result follows immediately by induction hypothesis. Otherwise, if $y = y_j$ for some $j = 1, \ldots, m$, then the Definition of Decompose rule hence that y is a fresh variable. Therefore, y does not appear in any constraint of $S \cup P_1$.

Case *(R)* = *(Sol)*:

Then $P' = P \cup \{y : u \triangleq v\}$, $S = S' \cup \{y : u \triangleq v\}$ and $\sigma = \sigma'$. Therefore, the result follows by the induction hypothesis.

Case *(R)* = *(Rec)*:

Then there is nothing to prove since $P' = P \cup \{y : u \triangleq v\}$ and $S = S_1 \cup \{z : u \triangleq v\} = S'$.

□

Lemma A.2 (Range of Substitutions [1]). Given terms $s, t \in T(\mathcal{X}, \Sigma_\emptyset)$ and a fresh variable x such that $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \mid S \mid \sigma \rangle$. Then, $\texttt{Index}(P \cup S) \subseteq \texttt{vran}(\sigma) \cup \{x\}$ and $\texttt{vran}(\sigma) \subseteq \mathcal{V}(x\sigma)$.

Proof. By induction over the length n of the reduction

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{n}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \mid S \mid \sigma \rangle.$$

Base Case: If $n = 0$ then $P = \{x : s \triangleq t\}$, $S = \emptyset$ and $\sigma = id$. Hence the definition of the set of index variable (Definition 2.2) and the definition of range of variables (Definition 1.7) implies that

$$\texttt{Index}(P) = \{x\} \text{ and } \texttt{vran}(\sigma) = \emptyset.$$

Then, it is easy to see that $\texttt{Index}(P) \subseteq \texttt{vran}(\sigma) \cup \{x\}$. Furthermore, for Definition 1.1 it follows that $\mathcal{V}(x\sigma) = \mathcal{V}(x) = \{x\}$, then $\texttt{vran}(\sigma) \subseteq \mathcal{V}(x\sigma)$.

Inductive Step: If $n > 0$ then we split the derivation as

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{n-1}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P' \mid S' \mid \sigma' \rangle \Longrightarrow_{(R)} \langle P \mid S \mid \sigma \rangle.$$

By induction hypothesis it follows that

$$\mathtt{Index}(P' \cup S') \subseteq \mathtt{vran}(\sigma') \cup \{x\} \text{ and } \mathtt{vran}(\sigma') \subseteq \mathcal{V}(x\sigma').$$

The proof follows by analysing each rule (R) of Figure 2.2 that could have been applied in the n-th step of the derivation:

Case *(R) = (Dec):*

Then,

$$P' = P_1 \cup \{y : f(\overline{s_m}) \triangleq f(\overline{t_m})\},\ P = P_1 \cup \{y_1 : s_1 \triangleq t_1, \ldots, y_m : s_m \triangleq t_m\},$$
$$S' = S \text{ and } \sigma'\{y \mapsto f(\overline{y_m})\} = \sigma.$$

Notice that the Lemma 2.1 implies that y does not appear in any constraint of $P_1 \cup S$. Then, $\{y\} \not\subseteq \mathtt{Index}(P_1 \cup S)$. Hence, by Definition 2.2 and Definition 1.7, it follows that

$$\mathtt{Index}(P \cup S) = \mathtt{Index}(P_1 \cup S) \cup \{\overline{y_m}\} \cup \{x\}. \tag{A.1}$$

Since $\mathtt{Index}(P') = \{y\} \cup \mathtt{Index}(P_1)$, by induction hypothesis it follows that $y \in \mathtt{vran}(x\sigma')$, but the composition of substitutions $\sigma = \sigma'\{y \mapsto f(\overline{y_m})\}$ and the freshness of each variable $y_1, \ldots, y_n$ implies that $y \notin \mathtt{vran}(x\sigma)$. Then,

$$\mathtt{vran}(\sigma') \subseteq \mathtt{vran}(\sigma) \cup \{y\}. \tag{A.2}$$

Now, notice that $\mathtt{Index}(P_1 \cup S) \subseteq \mathtt{vran}(\sigma') \cup \{x\}$ by induction hypothesis. And by Definition 1.7 it follows that $\{\overline{y_m}\} \in \mathtt{vran}(\sigma)$. Hence (A.1) and (A.2) implies that $\mathtt{Index}(P \cup S) \subseteq \mathtt{vran}(\sigma) \cup \{x\} \cup \{y\}$.

Further, since y does not appear in any constraint of P_1 and S, we obtain the first result required: $\mathtt{Index}(P \cup S) \subseteq \mathtt{vran}(\sigma) \cup \{x\}$. Besides, by Definition 1.1 and Lemma 2.1 it follows that $\mathtt{vran}(\sigma) \cup \{y\} = \mathtt{vran}(\sigma') \cup \{\overline{y_m}\}$.

Hence, the induction hypothesis implies that $\mathtt{vran}(\sigma) \cup \{y\} \subseteq \mathcal{V}(x\sigma') \cup \{\overline{y_m}\}$. Since $\sigma = \sigma'\{y \mapsto f(\overline{y_m})\}$, it is easy to see that $\mathcal{V}(x\sigma) = (\mathcal{V}(x\sigma') \cup \{\overline{y_m}\})/\{y\}$. Then we obtain that $\mathtt{vran}(\sigma) \subseteq \mathcal{V}(x\sigma)$.

Case *(R)= (Sol)*:

Then $P' = P \cup \{y : s_1 \triangleq t_1\}, S = S \cup S'$ and $\sigma' = \sigma$. It follows that $\mathtt{Index}(P \cup S) = \mathtt{Index}(P' \cup S')$ and $\mathtt{vran}(\sigma) = \mathtt{vran}(\sigma')$. Then the result follows directly from the induction hypothesis.

Case *(R) = (Rec)*:

Then, $P' = P \cup \{y : s_1 \triangleq s_2\}$, $S' = S_1 \cup \{z : s_1 \triangleq t_1\} = S$, and $\sigma'\{y \mapsto z\} = \sigma$. It follows that $\mathtt{Index}(P \cup S) \subseteq \mathtt{Index}(P' \cup S')$ and $\mathtt{vran}(\sigma) \subseteq \mathtt{vran}(\sigma')$. Then, $\mathtt{Index}(P \cup S) \cup \{x\} \subseteq \mathtt{vran}(\sigma)$, follows by induction hypothesis. Furthermore, the proof that $\mathtt{vran}(\sigma) = \mathcal{V}(x\sigma)$ is similar what was done in (Dec) case.

□

Lemma A.3 (Lemma 3 in [1]). Let $\mathcal{A}\langle s,t\rangle$ be an AUP. There is a derivation of the form $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle \{y : u \triangleq v\} \cup P \mid S \mid \sigma\rangle$ if, and only if, there exists a position $p \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$ such that $s|_p = u$, $t|_p = v$, and for all position $p' < p$, $\mathtt{root}(s|_{p'}) = \mathtt{root}(t|_{p'})$.

Proof.

$(\Longrightarrow)$ Suppose there is a sequence $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$. We want to prove that there exists a position $p \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$, such that $s|_p = u$, $t|_p = v$, and for all $p' < p$, $\mathtt{root}(s|_{p'}) = \mathtt{root}(t|_{p'})$. The proof is by induction in the length n of the reduction $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$.

Base Case: If $n = 0$ then $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle = \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$, i.e,

$$P = \emptyset, \quad \{y : u \triangleq v\} = \{x : s \triangleq t\}, \quad S = \emptyset, \quad \sigma = id.$$

Then, the result follows trivially by taken $p = \varepsilon$.

Inductive Step: If $n > 0$ the reduction unfolds as

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P' \mid S' \mid \sigma'\rangle \Longrightarrow_{(R)} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle.$$

The proof proceeds by analysing of each rule (R) of Figure 2.2 applied in the last step.

Case *(R) = (Dec)*:

Then, the reduction is of the form

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P_1 \cup \underbrace{\{w : f(\overline{u_m}) \triangleq f(\overline{v_m})\}}_{P_2'} \mid S \mid \sigma'\rangle$$

$$\Longrightarrow_{(Dec)} \langle P_1 \cup \underbrace{\{y_1 : u_1 \triangleq v_1, \ldots, y_m : u_m \triangleq v_m\} \mid S \mid \sigma\{w \mapsto f(\overline{y_m})\}.}_{P_2}\rangle$$

where, $P' = P_1 \cup P_2'$, $P = P_1 \cup P_2$, $S = S'$ and $\sigma = \sigma'\{w \mapsto f(\overline{y_m})\}$.

There are two cases to consider.

1. If $\{y : u \triangleq v\} \neq \{y_j : u_j \triangleq v_j\}$, for all $j = 1,\dots,m$, then $\{y_1 : u \triangleq v\} \in P_1$ and by induction hypothesis it follows that exists $p \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$ such that $s|_p = u$, $t|_p = v$ and for all $p' < p$, $\mathtt{root}(s|_{p'}) = \mathtt{root}(t|_{p'})$.
2. If $\{y : u \triangleq v\} = \{y_j : u_j \triangleq v_j\}$, for some $j = 1,\dots,n$. Then this constraint was created from an application of (Dec) in $\{w : f(\overline{u_m}) \triangleq f(v_m)\} \in P'$, which induction hypothesis yields that exist a position $q \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$ such that $\mathtt{root}(s|_q) = f(\overline{u_m})$ and $\mathtt{root}(t|_q) = f(\overline{v_m})$. Since u_j and v_j are respectively the j-th arguments of $f(\overline{u_m})$ and $f(\overline{v_m})$, taking $p = q.j$, it follows that

$$s|_p = s|_{q.j} = f(\overline{u_m})|_j = u_j = u \text{ and } t|_p = t|_{q.j} = f(\overline{v_m})|_j = v_j = v.$$

Furthermore, if $p' < p = q.j$, then $p' = q$ or $p' < q$. On one hand it follows that the induction hypothesis implies that for all $p' < q$, holds that $\mathtt{root}(s|_{p'}) = \mathtt{root}(t|_{p'})$. On the other hand, for $p' = q$, it is easy to see that $\mathtt{root}(s|_q) = \mathtt{root}(t|_q) = f$. Therefore, for every position $p' < p$, $\mathtt{root}(s|_{p'}) = \mathtt{root}(t|_{p'})$.

Case *(R) = (Sol)*:

Then, the reduction is of the form

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P \cup \underbrace{\{z : s_1 \triangleq s_2\}}_{P_1} \mid S' \mid \sigma\rangle \Longrightarrow_{(Sol)} \langle P \mid S \cup \underbrace{\{z : s_1 \triangleq s_2\}}_{P_1} \mid \sigma.\rangle$$

where, $P' = P \cup P_1$, $S = S' \cup P_1$ and $\sigma' = \sigma$. Then, $\{y : u \triangleq v\} \in P$ and by induction hypothesis it follows that exists $p \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$ such that $s|_p = u$, $t|_p = v$ and for all $p' < p$, $\mathtt{root}(s|_{p'}) = \mathtt{root}(t|_{p'})$.

Case *(R) = (Rec)*:

The argument is similar to that used in the Case $(R) = (Sol)$.

$(\Longleftarrow)$ Suppose that $p \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$ is such that $\mathtt{root}(s|_p) = u$, $\mathtt{root}(t|_p) = v$ and for all $p' < p$, $\mathtt{root}(s|_{p'}) = \mathtt{root}(t|_{p'})$. We want to prove that exists a derivation

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle.$$

The proof is by induction over the size p.

Base Case: If $p = \varepsilon$, then $s|_p = s$ and $t|_p = v$. Then, taking $S = \emptyset$, $P = \emptyset$, $\sigma = if$ and $\{y : u \triangleq v\} = \{x : s \triangleq t\}$ it follows that $\langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle = \langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ and the result follows trivially.

Inductive Step: If $p > 0$, then it follows that $p = q.j$ with j being a natural number. Since $p \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$, then $q \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$. Therefore there exist terms s_q and t_q such that $s|_q = s_q$ and $t|_q = t_q$. Notice that $s|_p = s|_{q.j} = u$ and $t|_p = t|_{q.j} = v$ are subterms of s_q and t_q, it follows that the root symbol of s_q and t_q must be both of functional type with arities bigger or equal to j. It is easy to see that $q < p$. Hence, the induction hypothesis can applies in q, which implies that $\mathtt{root}(s|_q) = \mathtt{root}(t|_q)$. Therefore, we can suppose without loss of generality that exist a function symbol $f : m \in \Sigma_\emptyset$ such that $s_q = f(\overline{u_m})$ and $t_q = f(\overline{v_m})$, with $u = u_j$ and $v = v_j$ for some $j \leq m$.

By induction hypothesis there exists a configuration C such that

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P_1 \cup \{z : f(\overline{u_m}) \triangleq f(\overline{v_m})\} \mid S \mid \sigma'\rangle$$

. Then, applying (Dec) in C it follows by the transitivity of $\overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset}$ that

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P_1 \cup \{y_1 : u_1 \triangleq v_1, \ldots, y_m : u_m \triangleq v_m\} \mid S \mid \sigma'\{z \mapsto f(\overline{y_m})\}\rangle.$$

Then, taking $P = P_1 \cup ((\bigcup_{i=1}^{m}\{y_i : u_i \triangleq v_i\})/\{y_j : u \triangleq v\})$, $\sigma = \sigma'\{z \mapsto f(\overline{y_m})\}$ and $y_j = j$ the result follows.

□

Lemma A.4 (Lemma 4 in [1]). Let $\mathcal{A}\langle s, t\rangle$ be an AUP. There is a derivation of the form $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P \mid \{y : u \triangleq v\} \cup S \mid \sigma\rangle$ if, and only if, there exists a conflict position p of s and t such that $s|_p = u$ and $t|_p = v$.

Proof.

($\Longrightarrow$) Suppose that $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P \mid \{y : u \triangleq v\} \cup S \mid \sigma\rangle$. Since $\{y : u \triangleq v\}$ is in the set of solved constraints S, then there is a step in this reduction such that rule (Sol) was applied. Then, it is possible unfold the derivation as

$$\begin{aligned} \langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle\{y : u \triangleq v\} \cup P_1 \mid S_1 \mid \sigma_1\rangle \\ &\Longrightarrow_{(Sol)} \langle P_1 \mid \{y : u \triangleq v\} \cup S_1 \mid \sigma_1\rangle \\ &\overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P \mid \{y : u \triangleq v\} \cup S \mid \sigma\rangle \end{aligned} \tag{1}$$

Notice that we can apply Lemma 2.3 in (1) which implies that there exists $p \in \texttt{pos}(s) \cup \texttt{pos}(t)$ such that $s|_p = u$ and $t|_p = v$, and for all position $q < p$, $\texttt{root}(s|_q) = \texttt{root}(p|_q)$. Since (Sol) was applied in $\{y : u \triangleq v\}$ then $\texttt{root}(u) \neq \texttt{root}(v)$. Therefore, p is a conflict position and (u, v) is a conflict pair.

($\Longleftarrow$) Suppose that $p \in \texttt{pos}(s) \cap \texttt{pos}(t)$ is a conflict position and $(u, v) = (s|_p, t|_p)$ is a conflict pair. We want to show that exists a reduction

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \mid S \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle = C_1.$$

Lemma 2.3 implies that exist a reduction such that

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P' \cup \{z : u \triangleq v\} \mid S' \mid \sigma'\rangle$$

There are two cases:

1. If there is no variable z such that $\{z : u \triangleq v\} \in S'$. Then, the result follow as apply (Sol) in the last constraint, as is shown bellow

$$\begin{aligned}\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P' \cup \{z : u \triangleq v\} \mid S' \mid \sigma'\rangle \\ &\Longrightarrow_{(Sol)} \langle P' \mid S' \cup \{z : u \triangleq v\} \mid \sigma\rangle,\end{aligned}$$

   and as take $y = z$, $P' = P$, $S = S' \cup \{y : u \triangleq v\}$ and $\sigma = \sigma'$.

2. If already exists a variable y such that $\{y : u \triangleq v\} \in S'$. Then the result follows immediately as take $S = S'$, $P = P'$ and $\sigma = \sigma'$.

□

Lemma A.5. Given terms $s, t \in T(\mathcal{X}, \Sigma_\emptyset)$. If there exists a derivation

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle \{y : u \triangleq v\} \cup P \mid S \mid \sigma\rangle$$

then there exists a position $p \in \texttt{pos}(s) \cap \texttt{pos}(t)$ such that

$$(x\sigma|_p) = y, \quad s|_p = (x\sigma)|_p\{y \mapsto u\} = u, \quad t|_p = (x\sigma)|_p\{y \mapsto v\} = v.$$

Proof. By induction on the length n of the reduction

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle \{y : u \triangleq v\} \cup P \mid S \mid \sigma\rangle$$

Induction base: If $n = 0$ then $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle = \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$. Taking $p = \varepsilon$, it follows that $(xid)|_\varepsilon = x$, $s|_\varepsilon = (xid)|_\varepsilon\{x \mapsto s\} = s$ and $t|_\varepsilon = (xid)|_\varepsilon\{x \mapsto t\} = t$.

Inductive step: Suppose that $n > 0$ and consider the reduction

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P' \mid S' \mid \sigma'\rangle \Longrightarrow_{(R)} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$$

The proof proceeds by analysing of each inference rule (R) of Figure 2.2 applied in the last n-th step.

Case *(R) = (Dec)*:

Then, the reduction is of the form

$$\begin{aligned}\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P_1 \cup \{z : f(\overline{u_m}) \triangleq \overline{v_m}\} \mid S \mid \sigma'\rangle \\ &\Longrightarrow_{(Dec)} \langle P_1 \cup \{y_1 : u_1 \triangleq v_1, \ldots, y_m : u_m \triangleq v_m\}\} \mid S \mid \sigma'\{z \mapsto f(\overline{u_m})\}.\rangle\end{aligned}$$

Where $P' = P_1 \cup \{z : f(\overline{u_m}) \triangleq \overline{v_m}\}$, $P = P_1 \cup \{y_1 : u_1 \triangleq v_1, \ldots, y_m : u_m \triangleq v_m\}$, $S = S'$ and $\sigma = \sigma'\{z \mapsto f(\overline{y_m})\}$.

There are two cases to consider.

1. If $y \neq y_j$ for all $j = 1, \ldots, m$ then $\{y : u \triangleq v\} \in P_1$ and the result follows by induction hypothesis.
2. If $y = y_j$ for some $j = 1, \ldots, m$, then $u = u_j$ and $v = v_j$, i.e., u is the j-th argument of $f(\overline{u_m})$ and v is the j-th argument of $f(\overline{v_m})$. Since

$$\{z : f(\overline{u_m}) \triangleq f(\overline{v_m})\} \in P',$$

the induction hypothesis applies to this constraints. Which implies that exist a position $q \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$ such that $s|_q = f(\overline{u_m})$, $t|_q = f(\overline{v_m})$. Therefore, taking $p = q.j$, it follows that

$$s|_p = s|_{q.j} = (s|_q)|_j \overset{I.H.}{=} f(\overline{u_m})|_j = u_j = u,$$

$$t|_p = t|_{q.j} = (t|_q)|_j \overset{I.H}{=} f(\overline{v_m})|_j = v_j = v.$$

The induction hypothesis implies that $x\sigma'|_q = z$, then $x\sigma' = x\sigma'[z]_q = z$. The freshness of z in $\langle P' \mid S' \mid \sigma'\rangle$ implies that the unique occurrence of z in $x\sigma'$ is in the position q. Then, as applying $x\sigma'\{z \mapsto f(\overline{y_m})\}$ the substitution $\{z \mapsto f(\overline{y_m})\}$ will changes the term $x\sigma'$ only in position q. It follows that

$x\sigma = x\sigma'[f(\overline{y_m})]_q$. Therefore,

$$\begin{aligned}(x\sigma)|_p &= (x\sigma)|_{q.j}\\ &= (x\sigma'[f(\overline{y_m})]_q)|_{q.j}\\ &= ((x\sigma'[f(\overline{y_m})]_q)|_q)|_j\\ &= f(\overline{y_m})|_j\\ &= y_j = y.\end{aligned}$$

Finally, we obtain that

$$(x\sigma)\{y \mapsto u\}|_p = y\{y \mapsto u\} = u = s|_p$$

$$(x\sigma)\{y \mapsto v\}|_p = y\{y \mapsto v\} = v = t|_p.$$

Case *(R) = (Sol)* or *(R) = (Rec)*:

Then $P \subset P'$ and since $\{u : u \triangleq v\} \in P$, the result follows immediately by induction hypothesis.

□

Lemma A.6. Let $s,t \in T(\mathcal{X},\Sigma_\emptyset)$. If $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \mid S \mid \sigma\rangle$ then $x\sigma \in \texttt{gen}(s,t)$.

Proof. By induction on the length n of the reduction $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \mid S \mid \sigma\rangle$.

Base Case: If $n = 0$ then $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle = \langle P \mid S \mid \sigma\rangle$ which gives $(x id) = x$, the trivial generalizer of s and t.

Inductive Step: Supposing that

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P' \mid S' \mid \sigma'\rangle \Longrightarrow_{(R)} \langle P \mid S \mid \sigma\rangle$$

We want to show that $x\sigma \in \texttt{gen}(s,t)$. Note that $\sigma = \sigma'\delta$ for some δ that will obtained in the last step. The proof proceed analysing the rule (R) applied in this least step.

Case *(R) = (Sol)*:

Then $\delta = id$ and the result follows by the induction hypothesis $(x\sigma') = (x\sigma')id \in \texttt{gen}(s,t)$.

Case *(R) = (Rec)*:

Then, $P' = P \cup \{y : u \triangleq v\}$, $S = S_1 \cup \{z : u \triangleq v\} = S'$ and $\sigma = \sigma'\{y \mapsto z\}$. It follows that exists a step $n - m$, for some $1 < m < n$ such that (Sol) was applied in a constraint $\{z : u \triangleq v\}$. Thus, calling $\delta = \{y \mapsto z\}$, the reduction is of the form:

$$\begin{aligned}\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle\{z : u \triangleq v\} \cup P'' \mid S'' \mid \sigma''\rangle = C_1\\ &\Longrightarrow_{(Sol)} \langle P'' \mid S'' \cup \{z : u \triangleq v\} \mid \sigma''\rangle = C_2\\ &\overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle\{y : u \triangleq v\} \cup P \mid \{z : u \triangleq v\} \cup S_1 \mid \sigma'\rangle = C_3\\ &\Longrightarrow_{(Rec)} \langle P \mid \{z : u \triangleq v\} \cup S_1 \mid \underbrace{\sigma'\delta}_{\sigma}\rangle = C_4\end{aligned}$$

Hence, the Lemma 2.5 in C_1 implies that exists a position $q \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$ such that $s|_q = u, t|_q = v$ and $x\sigma''|_q = z$. And the Lemma 2.3 in C_1 yields that for every position q' such that $q' < q$, then $\mathtt{root}(s|_{q'}) = \mathtt{root}(t|_{q'})$. Similarly, Lemma 2.5 in C_3 implies that exists a position $p \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$ such that $s|_p = u, t|_p = v$ and $x\sigma'|_p = y$. Furthermore, the Lemma 2.3 in C_3 implies that for every position p' such that $p' < p$, then $\mathtt{root}(s|_{p'}) = \mathtt{root}(t|_{p'})$. Hence, since $\mathtt{root}(s|_p) \neq \mathtt{root}(t|_p)$, it follows that p and q must to be parallel positions.

Notice that Lemma 2.1 implies that the unique constraint which z appear is the constraint $\{z : u \triangleq v\} \in S$. Since $\{z : u \triangleq v\}$ is solved in C_2 then it will be in the set of solved constraint for all configurations obtained from C_2. Since there is no rule in Figure 2.2 that can compute a substitution which will change the value of z, it follows that $z \notin \mathtt{dom}(\sigma')$ because the unique rule in Figure 2.2 that change the domain of the substitution computed is (Dec) and this rule never was applies in $\{z : u \triangleq v\}$. Therefore, $x\sigma|_q = z$.

Consequently, for every position $r \neq p$, $x\sigma'|_r = x\sigma|_r$. On the other hand $x\sigma'|_p = y$, while $x\sigma|_p = y\{y \mapsto z\} = y\delta = z$.

By the induction hypothesis in C_3 it follows that $x\sigma' \in \mathtt{gen}(s,t)$. Then, there is a pair of substitutions $\overline{\theta} = (\theta_1, \theta_2)$, such that $x\theta_1 = s$ and $x\theta_2 = t$. Hence,

$$x\sigma'\theta_1|_p = s|_p = u, \quad x\sigma'\theta_2|_p = t|_p = v,$$

$$x\sigma'\theta_1|_q = s|_q = u, \quad x\sigma'\theta_2|_q = t|_q = v.$$

It follows that θ_1 changes the subterms of $x\sigma'$ at position p and q, i.e., $\theta_1 y = u$ and $\theta_1 z = u$. Constructing a substitution θ_1' such that $w\theta_1' = w\theta$ for every variable $w \neq y, z$ and such that $y, z \notin \mathtt{dom}(\theta_1')$, i.e, θ' is equal to θ with y and z taken out of its domain.

$$\begin{aligned}\theta_1 &= \{y \mapsto u, z \mapsto u\}\theta_1' \\ &= \{y \mapsto u\}\{z \mapsto u\}\theta_1' \\ &= \delta\{z \mapsto u\}\theta_1'.\end{aligned}$$

By an analogous argument it follows that there exists a substitution θ_2' such that

$$\theta_2 = \delta\{z \mapsto v\}\theta_2'.$$

Then, taking the pair of substitutions $\overline{\theta'} = (\{z \mapsto u\}\theta_1', \{z \mapsto v\}\theta_2')$ it follows that

$$x\sigma\overline{\theta'} = x\sigma'\delta\overline{\theta'} = (s,t).$$

Therefore, $x\sigma \in \mathtt{gen}(s,t)$.

□

Lemma A.7. Let $s,t \in T(\mathcal{X},\Sigma_\emptyset)$. If $u \in \mathtt{gen}(s,t)$ then there exists derivation

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P \mid S \mid \sigma\rangle$$

such that $u \equiv x\sigma$

Proof. By induction on the structure of u.

Base Case: If $u \in \mathcal{X}$, i.e, if u is a variable. The result follows if ones takes

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle = \langle P \mid S \mid \sigma\rangle,$$

which implies that $\sigma = id$ and, consequently, $x\,id = x \equiv u$.

Inductive Step: If $u = f(u_1,\ldots,u_n)$, for an n-ary free function symbol f. Suppose that the result follows for generalizers u_i of s and t with structure simpler than u.

Since $u \in \mathtt{gen}(s,t)$ it follows there exists a pair of substitutions $\overline{\theta} = (\theta_1,\theta_2)$ such that $u\overline{\theta} = (s,t)$. Then both s and t have as root symbol the n-ary function symbol f. Which follows that $s = f(s_1,\ldots,s_n)$ and $t = f(t_1,\ldots,t_n)$ for some s_i, $t_i \in T(\mathcal{X},\Sigma_\emptyset)$, with $i = 1,\ldots,n$. Therefore, the initial configuration is

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle = \langle \{x : f(\overline{s_n}) \triangleq f(\overline{t_n})\} \mid \emptyset \mid id\rangle.$$

We want to show that exists a derivation

$$\langle\{x : f(\overline{s_n}) \triangleq f(\overline{t_n})\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P \mid S \mid \sigma\rangle \tag{1}$$

such that $x\sigma \equiv u = f(u_1,\ldots,u_n)$. We will show the existence of this derivation constructing it.

Notice that (Dec) apply in the initial configuration. Therefore, the first step of the derivation (1) is of the form:

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \Longrightarrow \langle\{x_1 : s_1 \triangleq t_1,\ldots,x_n : s_n \triangleq t_n\} \mid \emptyset \mid \underbrace{\{x \mapsto f(x_1,\ldots,x_n)\}}_{\sigma_0}\rangle \tag{2}$$

To obtain the next steps it is necessary to use the induction hypothesis. First, let's prove the claim bellow.

Claim: $u_i \in \mathtt{gen}(s_i,t_i)$, for all $1 \le i \le n$.

In fact, it is enough to notice that

$$u\theta_1 = f(u_1\theta_1,\ldots,u_n\theta_1) = f(s_1,\ldots,s_n),$$

$$u\theta_2 = f(u_1\theta_2,\ldots,u_n\theta_2) = f(t_1,\ldots,t_n).$$

Hence, $u_i\theta_1 = s_i$ and $u_i\theta_2 = t_i$. Then, u_i is a generalizer of s_i and t_i for every $1 \le i \le n$.

Therefore, the induction hypothesis implies that for each $i = 1,\ldots,n$ there exists a derivation R_i composed of the rules $\mathtt{AUnif}_\emptyset$ given in Figure 2.2 such that

$$\langle x_i : s_i \triangleq t_i \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{R_i} \langle C_i \mid S_i \mid \sigma_i\rangle \tag{3}$$

such that $u_i \equiv x_i\sigma_i$.

The idea of this proof is after apply (Dec) in the initial configuration, as is shown in (2), and after sequentially apply each of the derivations chains given in (3), in order to obtain a configuration $\langle P \mid S \mid \sigma\rangle$ with $\sigma = \sigma_0\sigma_1\ldots\sigma_n$, such that

$$\begin{aligned} x\sigma &= x\sigma_0 \dots \sigma_n \\ &= (x\sigma_0)\sigma_1 \dots \sigma_n \\ &= f(x_1,\dots,x_n)\sigma_1 \dots \sigma_n \\ &= f(x_1\sigma_1,\dots,x_n\sigma_n) \equiv f(u_1,\dots,u_n) \end{aligned}$$

However, it is necessary ensure that this combination of derivations will be done in a way that each substitution σ_k obtained from the derivation R_k dos not change the value of the term $x_j\sigma_j \equiv u_j$ obtained by the derivation R_j, i.e, it is necessary to ensure that $f(x_1,\dots,x_n)\sigma_1 \dots \sigma_n = f(x_1\sigma_1,\dots,x_n\sigma_n)$. Notice that $\mathtt{dom}(\sigma_i) = x_i$ and by Lemma 2.2 it follows that $\mathtt{vran}(\sigma_i) \subseteq \mathcal{V}(x\sigma_i)$. Then, it is enough compare the variables of each u_j and u_k for every $k \neq j$. This analyses of variables is also important to ensure that the rules of Figure 2.2 were applied correctly as combine the derivations of (3), because just join these rules could generates a application of (Sol) in a constraint which (Rec) should apply instead. There are two cases of variables analysing, studied in the following.

- **Case 1:** If $\mathcal{V}(u_j) \cap \mathcal{V}(u_k) = \emptyset$ for every $j \neq k$.

  This case was analysed in 2, Lemma 2.7.

- **Case 2:** If $\mathcal{V}(u_j) \cap \mathcal{V}(u_k) = y$, for some $j,k = 1,\dots,n$.

  It follows that the variable y is a subterm of u_j and also a subterm of u_k. Thus, there are positions $p \in \mathtt{pos}(u_j)$ and $q \in \mathtt{pos}(u_k)$ such that $y = u_j|_p = u|_{jp}$, and $y = u_k|_q = u|_{kq}$. Notice that, since u_i is a generalizer of s_j and t_j then y is a generalizer of a pair subterms of s_j and t_j. Similarly, since u_k is a generalizer of s_k and t_k, then y is a generalizer of a pair of subterms of s_k and t_k. More precisely, there exist terms v_1 and v_2 such that

$$s|_{jp} = t|_{jp} = v_1 \text{ and } s|_{kq} = t|_{kq} = v_2.$$

By Lemma 2.3 there exists constraints $\{z_j : v_1 \triangleq v_2\}$ in the set of unsolved constraint of some configuration obtained by the derivations R_j, and a non-solved constraint $\{z_k : v_1 \triangleq v_2\}$ in some configuration obtained by the derivation R_k. Therefore, we can unfold these derivations as follows

$$\langle x_l : s_l \triangleq t_l \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{R_l'} \langle C_l' \mid S_l'\{z_l : v_1 \triangleq v_2\} \mid \sigma_l' \rangle \overset{*}{\Longrightarrow}_{R_l''} \langle C_l \mid S_l \mid \sigma_l \rangle$$

with $l = i, j$. The proof follows by analysing of v_1 and v_2:

1. If $v_1 = v_2 = y$: Then the occurrence of y in $x\sigma_j$ and in $x\sigma_k$ was given by the respective application of (Dec) in $\{z_j : v_1 \triangleq v_2\}$ and in $\{z_k : v_1 \triangleq v_2\}$. Then, the result follows just collect all reductions R_i given by (3), with $i = 1, \ldots, n$, in the same way as we did in the case 1.
2. $\texttt{root}(v_1) \neq \texttt{root}(v_2)$: Then $z_j = z_k = y$. It follows that (v_1, v_2) is a conflict pair of subterms of s_j and t_j with respect to the conflict position $p \in \texttt{pos}(s_j) \cap \texttt{pos}(t_j)$. Then, we can split the reduction R_j as

$$\langle x_j : s_j \triangleq t_j \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{R'_j} \langle C'_j \mid S'_j\{y : v_1 \triangleq v_2\} \mid \sigma'_j \rangle \overset{*}{\Longrightarrow}_{R''_j} \langle C_j \mid S_j \mid \sigma_j \rangle$$

Similarly, (v_1, v_2) is a conflict pair of subterms of s_k and t_k with respective conflict position $q \in \texttt{pos}(u_k) \cap \texttt{pos}(v_k)$. Then we can split the reduction R_k as

$$\begin{aligned}\langle x_k : s_k \triangleq t_k \mid \emptyset \mid id \rangle &\overset{*}{\Longrightarrow}_{(R'_k)} \langle C'_k \cup \{y : v_1 \triangleq v_2\} \mid S'_k \mid \sigma'_k \rangle \\ &\Longrightarrow_{(Sol)} \langle C'_k \mid S'_k \cup \{y : v_1 \triangleq v_2\} \mid \sigma'_k \rangle \\ &\overset{*}{\Longrightarrow}_{(R''_k)} \langle C_k \mid S_k \mid \sigma_k . \rangle\end{aligned}$$

Then (Sol) was applied in the constraint $\{y : v_1 \triangleq v_2\}$ in some step of R_j and also applied in the same constraint in some step of R_k. Since there is no rule in Figure 2.2 that changes the set of solved constraints, it follows that $\{y : v_1 \triangleq v_2\} \in S_j \cap S_k$. Therefore, we cannot just join all the reductions of the derivations given by (3) as was did in the previous case. Otherwise, we would obtain two applications of (Sol) in constraints representing the same anti-unification problem $\mathcal{A}\langle v_1, v_2 \rangle$, what is not allowed by definition of (Sol) in Figure 2.2.
Then, it is necessary to do a variable renaming in all configurations of the derivation $\langle \{x_k : s_k \triangleq t_k\} \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{R'_k} \langle C'_k \cup \{y : v_1 \triangleq v_2\} \rangle$, substituting every occurrence of y by a fresh variable z. Suppose without loss of generality that $j < k$. The results follows as join all derivations in (3), just making the small change described in the following: Apply all the rules starting from R_1 until R'_k. To avoid the second application of (Sol) in the reduction obtained, we apply (Rec) instead of (Sol). Finally, we apply all the remaining rules $R''_k, R_{k+1}, \ldots, R_n$ in the reduction. Therefore the reduction is of the form

showed below:

$$\begin{tikzcd}
\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \arrow[d, Rightarrow, "*"', "(R_1),\dots,(R_i),\dots,(R'_k)"] & \\
\langle \{z : v_1 \triangleq v_2\} \cup P_k \cup C_1 \cup \dots C_{k-1} \mid S_1 \cup \dots \cup S'_k \mid \sigma_0 \dots \sigma_{k-1}\sigma'_k \rangle, \arrow[d, Rightarrow, "*"', "(Rec)"] & \{y : v_1 \triangleq v_2\} \in S_i \\
\langle P_k \cup C_1 \cup \dots \cup C_{k-1} \mid S_1 \dots S_k \mid \sigma_0 \dots \sigma_{k-1}\sigma'_k\{z \mapsto y\} \rangle \arrow[d, Rightarrow, "*"', "(R''_k),\dots,(R_n)"] & \\
\langle \bigcup_{i=1}^{n} C_i \mid \bigcup_{i=1}^{n} S_i \mid \underbrace{\sigma_0 \dots \sigma_{k-1}\sigma'_k\{z \mapsto y\}\sigma''_k \dots \sigma_n}_{\sigma'} \rangle &
\end{tikzcd}$$

where $P_i = \{x_1 : s_1 \triangleq t_1, \dots, x_n : s_n \triangleq t_n\} / \bigcup_{i=1}^{n} \{x_i : s_i \triangleq t_i\}$. Finally, notice that $x\sigma_k = \sigma_{k'}\{z \mapsto y\}\sigma_{k''}$. Then, $x\sigma' = x\sigma \equiv f(u_1, \dots, u_n)$. Therefore, the result holds.

3. If $v_1 \neq v_2$ and $\mathtt{root}(v_1) = \mathtt{root}(v_2)$: This case does not need be analysed because there is no clash of variables with these conditions. In fact, (Dec) will apply in both $\{z_j : v_1 \triangleq v_2\}$ and $\{z_k : v_1 \triangleq v_2\}$ in each respective reduction chain. We can suppose without loss of generality that $\mathtt{root}(v_1) = \mathtt{root}(v_1) = f : n \in \Sigma$, then each application of decompose will create constraints with index variables $y_1, \dots, y_n$ constraints comparing the arguments of v_1 and v_2 and add a substitution $\{z_l \mapsto f(\overline{y_m})\}$. Then, v_1 and v_2 had disappears of $x\sigma$ without produces a clash of variables. The same happens if $v_1 = v_2 \neq y$.

□

Lemma A.8. Let $\mathcal{A}_\emptyset\langle s,t \rangle$ be an AUP. If there exists a derivation of the form

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \Longrightarrow^{*}_{\mathtt{AUnif}_\emptyset} \langle \emptyset \mid S \mid \sigma \rangle$$

with $S = \{y_1 : u_1 \triangleq v_1; y_2 : u_2 \triangleq v_2, \dots, y_n : u_n \triangleq v_n\}$, then the pair of substitution (θ_1, θ_2) given for$\theta_1 = \{y_1 \mapsto u_1, \dots, y_n \mapsto u_n\}$ and $\theta_2 = \{y_1 \mapsto v_1, \dots, y_n \mapsto v_n\}$ is such that

$$x\sigma\theta_1 = s \text{ and } x\sigma\theta_2 = t.$$

Proof. The proof proceeds by analyses of the conflict pairs of s and t.

Case 1: s and t do not have any conflict pair:

This case was analysed in Chapter 2, Lemma 2.8.

Case 2: s and t have conflict positions.

Suppose that $p_1,\dots,p_n$ are all the conflict positions of s and t, with respective conflict pair of subterms $(u_1,v_1),\dots,(u_n,v_n)$. Given two different conflict positions p_l and p_k, by Definition 2.4, $p.l$ is such that for every p' such that $p' < p_l$ or $p' < p_k$, it holds $\mathtt{root}(s|_{p'}) = \mathtt{root}(t|_{p'})$. On other hand, by Definition 2.4, $\mathtt{root}(s|_{p_l}) \neq \mathtt{root}(t|_{p_l})$ and $\mathtt{root}(s|_{p_k}) \neq \mathtt{root}(t|_{p_k})$. Then, p_j and p_k are parallel positions.

Then, by $p||q$ and Remark 2.4 it follows that exist a derivation such that every constraint $\{y_i : u_i \triangleq v_i\}$, with $i = 1,\dots,n$, appears in the set of unsolved constraint, i.e, there exists a derivation as is show bellow:

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P \cup \{y_1 : u_1 \triangleq v_1,\dots,y_n : u_n \triangleq v_n\} \mid \emptyset \mid \sigma'\rangle \qquad (1)$$

There are to cases:

1. If $(u_l,v_l) \neq (u_k,v_k)$ for every $l \neq k$, i.e, for each conflict position its respective conflict pair of subterms is unique. Then each constraint $\{y_i : u_i \triangleq v_i\}$ represents a different anti-unification problem $\mathcal{A}\langle u_i,v_i\rangle$. Applying (Sol) in each of these constraints in (3), we obtain the derivation

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_\emptyset} \langle P \mid \{y_1 : u_1 \triangleq v_1,\dots,y_n : u_n \triangleq v_n\} \mid \sigma'\rangle$$

   Lemma 2.4 yields that any new constraint will be add in this set of solved constraints, because it already have every constraints representing a conflict pair of s and t. By the confluence property of the rules of $\mathtt{AUnif}_\emptyset$ proved in Theorem 2.2, it follows that the final configuration $\langle \emptyset \mid S \mid \sigma\rangle$ of its derivation is such that $S = \{y_1 : u_1 \triangleq v_1,\dots,y_n : u_n \triangleq v_n\}$. Therefore,

$$\theta_1 = \{y_1 \mapsto u_1,\dots,y_n \mapsto u_n\} \text{ and } \theta_2 = \{y_1 \mapsto v_1,\dots,y_n \mapsto v_n\}.$$

   Theorem 2.3 implies that $x\sigma \in \mathtt{lgg}(s,t)$. It means that $x\sigma$ express more as possible both term structures of s and t. Formally, if a position $q \in \mathtt{pos}(s) \cap \mathtt{pos}(t)$ is such that $s|_q = t|_q$, then $x\sigma|_p = s|_q = t|_q$.

In the other hand for every position p such that the terms structures of s and t starts to diverges, this position generates a conflict pair of subterms of s and t, which is composed of subterms of s and t which roots symbols are different of each other. Then, $x\sigma|_p$ must be a variable. By Lemma 2.5 it follows that for each conflict position p_i, with $i = 1, \ldots, n$,

$$x\sigma|_{p_i}\{y_i \mapsto u_i\} = y_i\{y_i \mapsto u_i\} = u_i = s|_{p_i},$$

$$x\sigma|_{p_i}\{y_i \mapsto v_i\} = y_i\{y_i \mapsto v_i\} = v_i = t|_{p_i}.$$

It follows that $x\sigma\theta_1 = s$ and $x\sigma\theta_2 = t$.

2. If $(u_l, v_l) \neq (u_k, v_k)$ for some different $l, k \in \{1, \ldots, n\}$, i.e, there are two different conflict positions that generates the same conflict pair of subterms. In this case, $\{y_l : u_l \triangleq v_l\}$ represents the same anti-unification problem as the constraint $\{y_k : u_k \triangleq v_k\}$. To solve this constraint, we apply a sequence of (Sol) in (3) and after it we apply a (Rec) rule, as follows

$$\begin{aligned}\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\overset{*}{\Longrightarrow}_{\texttt{AUnif}_\emptyset} \langle P \cup \{y_1 : u_1 \triangleq v_1, \ldots, y_n : u_n \triangleq v_n\} \mid \emptyset \mid \sigma'\rangle \\ &\overset{*}{\Longrightarrow}_{(Sol)} \langle P \cup \{y_k : u_k \triangleq v_k\} \mid S' \cup \{y_k : v_l \triangleq y_l\} \mid \sigma'\rangle \\ &\overset{*}{\Longrightarrow}_{(Rec)} \langle P \mid \{y_k : u_k \triangleq v_k\} \mid S' \cup \{y_k : v_l \triangleq y_l\} \mid \sigma'\{y_k \mapsto y_l\}\rangle\end{aligned}$$

where $S' = (\bigcup_{i=1}^{n} \{y_i : u_i \triangleq v_i\}) / \{y_l : u_l \triangleq v_l, y_k : u_k \triangleq v_k\}$.

Hence, by Lemma 2.4, (Sol) will not being applied in any constraint obtained from P. Theorem 2.3 implies that the final configuration $\langle\emptyset \mid S \mid \sigma\rangle$ is such that $S = S' \cup \{y_l : u_l \triangleq v_l\}$. It follows that

$$\theta_1 = \{y_1 \mapsto u_1, \ldots, y_{k-1} \mapsto u_{k-1}, y_{k+1} \mapsto u_{k+1}, \ldots, y_n \mapsto u_n\}$$

$$\theta_2 = \{y_1 \mapsto v_1, \ldots, y_{k-1} \mapsto v_{k-1}, y_{k+1} \mapsto v_{k+1}, \ldots, y_n \mapsto v_n\}.$$

where $y_k \notin \texttt{dom}(\theta_1) \cup \texttt{dom}(\theta_2)$.

Lemma 2.1 hence each constraint created from P have index variable different of $y_1, \ldots, y_n$. Then, the substitution σ is such that $x\sigma|_{p_i} = x\sigma'\{y_k \mapsto y_l\}|_{p_i}$, for every $i = 1, \ldots, n$. Therefore, Lemma 2.5 implies that

$$x\sigma|_{p_j}\{y_j \mapsto u_j\} = y_j\{y_j \mapsto u_j\} = u_j = s_j,$$

$$x\sigma|_{p_j}\{y_j \mapsto v_j\} = y_j\{y_j \mapsto v_j\} = v_j = t_j,$$

for every $j = 1, \ldots, k-1, k+1, \ldots, n$. Meanwhile, for p_k it follows that

$$x\sigma|_{p_k}\{y_l \mapsto u_l\} = y_l\{y_l \mapsto u_l\} = u_l = u_k = s|_{p.k},$$

$$x\sigma|_{p_k}\{y_l \mapsto v_l\} = y_l\{y_l \mapsto v_l\} = v_l = v_k = t|_{p.k}.$$

By Theorem 2.3 it follows that $x\sigma$ already have more structure in common with s and t as possible. And the definition os conflit pair hence that for each conflict position its the respective conflict pair of subterms is composed for the diverging subterms of s and t. Then, $x\sigma\theta_1 = s$ and $x\sigma\theta_2 = t$.

$\square$

Appendix B

Appendix of Chapter 3

B.1 Proofs of Chapter 3

Lemma B.1 (Uniqueness of generalization variables cf. Lemma 12 in [1])**.** Let $\mathcal{A}_C\langle s,t\rangle$ be an AUP$_C$. If $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$ then y does not appear in any constraint in P or S.

Proof. The proof is by induction on the length n of the derivation

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle.$$

Base Case: If $n = 0$ then $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle = \langle P \mid S \mid \sigma\rangle$, i.e, $P = \emptyset$, $S = \emptyset$ and $\sigma = id$. Therefore, x does not appear in any other constraint of $P \cup S = \emptyset$.

Inductive step: If $n > 0$ then we split the derivation in

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P' \mid S' \mid \sigma'\rangle \Longrightarrow_{(R)} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle. \quad \text{(B.1)}$$

The proof follows by analyses of each inference rule (R) of Figure 3.1 applied in the n-th of the derivation. For rules (Dec), (Sol) and (Rec) the proof it was already done in Lemma 2.1. Therefore only *(C-Dec)* case remains to be verified.

Case *(R)* = *(C-Dec)*:
Then $P' = P_1 \cup \{y : g(s_1,s_2) \triangleq g(t_1,t_2)\}$, and the reduction is of the form

$$\begin{aligned}&\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P_1 \cup \{y : g(s_1,s_2) \triangleq g(t_1,t_2)\} \mid S' \mid \sigma'\rangle \\ &\qquad\Longrightarrow_{\textit{(C-Dec)}} \langle P_1 \cup \left\{\begin{matrix}\{y_1 : s_1 \triangleq t_1, y_2 : s_2 \triangleq t_2\} \\ \{y_1 : s_1 \triangleq t_2, y_2 : s_2 \triangleq t_1\}\end{matrix}\right\} \mid S \mid \sigma'\{y \mapsto g(y_1,y_2)\}\rangle\end{aligned}$$

Then, there are two possibilities for $\langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$, i.e,

$$\langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle = \langle P_1 \cup \{y_1 : s_1 \triangleq t_1, y_2 : s_2 \triangleq t_2\} \mid S \mid \sigma\rangle, \text{ or}$$
$$\langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle = \langle P_1 \cup \{y_1 : s_1 \triangleq t_2, y_2 : s_2 \triangleq t_1\} \mid S \mid \sigma\rangle$$

In any case $S = S'$ and $\sigma = \sigma'\{y \mapsto g(y_1, y_2)\}$.

1. $\langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle = \langle P_1 \cup \{y_1 : s_1 \triangleq t_1, y_2 : s_2 \triangleq t_2\} \mid S \mid \sigma\rangle$ .There are to cases to be considered depending on how constraint of P have y as index variable.
   (a) If $y = y_1$ or y_2. Then $\{y : u \triangleq v\}$ was created by application of *(C-Dec)* in the n-th step of the reduction. By definition of *(C-Dec)* rule it follows that y_1, y_2 are fresh variables. Hence, y_1 and y_2 do not occur $P_1 \cup S$.
   (b) If $y \neq y_1, y_2$. Then $\{y : u \triangleq v\} \in P_1$ and the result follows by induction hypothesis.
2. $\langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle = \langle P_1 \cup \{y_1 : s_1 \triangleq t_2, y_2 : s_2 \triangleq t_1\} \mid S \mid \sigma\rangle$. The argument since the variables still the same in any part of the configuration.

$\square$

Lemma B.2 (Range of substitutions cf. Lemma 13 in [1])**.** Let $\mathcal{A}_C\langle s, t\rangle$ be an AUP$_C$. If

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P \mid S \mid \sigma\rangle,$$

then $\mathtt{Index}(S \cup P) \subseteq \mathtt{vran}(\sigma) \cup \{x\}$, and $\mathtt{vran}(\sigma) \subseteq \mathcal{V}(x\sigma)$.

Proof. By induction over length n of the reduction $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P \mid S \mid \sigma\rangle$.

Base Case: If $n = 0$ then $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle = \langle P \mid S \mid \sigma\rangle$, then $P = \{x : s \triangleq t\}$, $S = \emptyset$ and $\sigma = id$. Hence, the definition of index variables (c.f. Definition 2.2) implies that $\mathtt{Index}(P) = \{x\}$, and the definition of range of variables (c.f. Definition 1.7) implies that and $\mathtt{vran}(\sigma) = \emptyset$. Then, it is easy to see that $\mathtt{Index}(P) \subseteq \mathtt{vran}(\sigma) \cup \{x\}$. Furthermore, by Definition 1.1 it follows that $\mathcal{V}(x\sigma) = \{x\}$, then $\mathtt{vran}(\sigma) \subseteq \mathcal{V}(x\sigma)$.

Inductive Step: If $n > 0$, the reduction unfolds as

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P' \mid S' \mid \sigma'\rangle \Longrightarrow_{(R)} \langle P \mid S \mid \sigma\rangle.$$

The proof follows by analyses of each inference rule (R) of Figure 3.1, where the cases which (R) is (Dec), (Sol) and (Rec) was already done during the proof of Lemma 2.2. Therefore remains only *(C-Dec)* case.

Case *(R) = (C-Dec)*:
Then, the reduction is of the form:

$$\begin{aligned}\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\overset{n-1}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P_1 \cup \{y : g(s_1,s_2) \triangleq g(t_1,t_2)\} \mid S' \mid \sigma'\rangle \\ &\Longrightarrow_{\textit{(C-Dec)}} \langle P_1 \cup \left\{\begin{matrix}\{y_1 : s_1 \triangleq t_1, y_2 : s_2 \triangleq t_2\} \\ \{y_1 : s_1 \triangleq t_2, y_2 : s_2 \triangleq t_1\}\end{matrix}\right\} \mid S \mid \sigma'\{y \mapsto g(y_1,y_2)\}\rangle\end{aligned}$$

With $P' = P_1 \cup \{y : g(s_1,s_2) \triangleq g(t_1,t_2)\}$. Notice that are two possibilities for P, depending on how *(C-Dec)* was applied in the n-th step, i.e,

$$P = P_1 \cup \{y_1 : s_1 \triangleq t_1, y_2 : s_2 \triangleq t_2\} \text{ or } P = P_1 \cup \{y_1 : s_1 \triangleq t_2, y_2 : s_2 \triangleq t_1\}$$

In any case $S = S'$ and $\sigma = \sigma'\{y \mapsto g(y_1,y_2)\}$.
By induction hypothesis, it follows that $\texttt{Index}(P' \cup S') \subseteq \texttt{vran}(\sigma') \cup \{x\}$ and $\texttt{vran}(\sigma') \subseteq \mathcal{V}(x\sigma)$.

1. $P = P_1 \cup \{y_1 : s_1 \triangleq t_1, y_2 : s_2 \triangleq t_2\}$.
   By Definition 2.2 it follows that

   $$\texttt{Index}(P \cup S) \subseteq \texttt{Index}(P_1 \cup S) \cup \{y_1,y_2\} \cup \{x\}$$

   Since $\sigma = \sigma'\{y \mapsto g(y_1,y_2)\}$, by Definition 1.7 it follows that $\texttt{vran}(\sigma') \subseteq \texttt{vran}(\sigma) \cup \{y\}$ and also that $\{y_1,y_2\} \subseteq \texttt{vran}(x\sigma)$. On the other hand, the induction hypothesis implies that $\texttt{Index}(P_1 \cup S) \subseteq \texttt{vran}(\sigma) \cup \{x\}$. Then, holds that $\texttt{Index}(P \cup S) \subseteq \texttt{vran}(\sigma') \cup \{x\} \cup \{y\}$. However, by Lemma 3.1, y does not appear in any constraint of $P_1 \cup S$. Hence, $\texttt{Index}(P \cup S) \subseteq \texttt{vran}(\sigma) \cup \{x\}$.
   By Definition 1.7 it follows that $\texttt{vran}(\sigma) \cup \{y\} = \texttt{vran}(\sigma') \cup \{y_1,y_2\}$. Hence, the induction hypothesis implies that $\texttt{vran}(\sigma) \cup \{y\} \subseteq \mathcal{V}(x\sigma') \cup \{y_1,y_2\}$. However, $\mathcal{V}(x\sigma) = (\mathcal{V}(x\sigma') \cup \{y_1,y_2\})/y$ because $\sigma = \sigma'\{y \mapsto g(y_1,y_2)\}$. Therefore, $\texttt{vran}(\sigma) \cup \mathcal{V}(x\sigma)$.
2. $P = P_1 \cup \{y_1 : s_1 \triangleq t_2, y_2 : s_2 \triangleq t_1\}$.
   The argument is the same since $\texttt{Index}(P \cup S), \texttt{vran}(\sigma)$ and $\mathcal{V}(x\sigma)$ did not changed.

□

Lemma B.3 (Lemma 14 in [1])**.** Let $\mathcal{A}_C\langle s,t\rangle$ be an AUP$_C$. There is a sequence of the form $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_C} \langle\{y : u \triangleq v\} \cup P \mid S \mid \sigma\rangle$ if, and only if, (u,v) is a commutative pair of subterms of s and t.

Proof.

($\Longrightarrow$) Suppose there exist a sequence $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$. We want to show that (u, v) is a commutative pair of subterms.

The proof is by induction in the length n of the reduction

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle.$$

Base Case: If $n = 0$ then $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle = \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$, then $S = \emptyset$, $\sigma = id$, $s = u$ and $t = v$ and (s, t) is trivially a commutative pair of subterms of s and t.

Induction step: If $n > 0$ the derivation unfolds as

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P' \mid S' \mid \sigma'\rangle \Longrightarrow_{(R)} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$$

The proof proceeds by analysing of each reduction rule (R) of $\texttt{AUnif}_C$ using in the last step of the derivation. The cases were (R) is (Dec), (Sol) or (Rec) was already verified in the proof of Lemma 2.3. Then, only the case where (R) is the *(C-Dec)* remains to be checked.

Case *(R) = (C-Dec)*:
Then the derivation is of the form

$$\begin{aligned}\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\overset{n-1}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P_1 \cup \{y : g(u_1, u_2) \triangleq g(v_1, v_2)\} \mid S' \mid \sigma'\rangle \\ &\Longrightarrow_{\textit{(C-Dec)}} \langle P_1 \cup \left\{\begin{array}{l}\{y_1 : u_1 \triangleq v_1, y_2 : u_2 \triangleq v_2\} \\ \{y_1 : u_1 \triangleq v_2, y_2 : u_2 \triangleq v_1\}\end{array}\right\} \mid S \mid \sigma'\{y \mapsto g(y_1, y_2)\}\rangle\end{aligned}$$

where $P' = P_1 \cup \{y : g(u_1, u_2) \triangleq g(v_1, v_2)\}$. Notice that are two possibilities of $\langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$, that are

$$\langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle = \langle P_1 \cup \{y_1 : u_1 \triangleq v_1, y_2 : u_2 \triangleq v_2\} \mid S \mid \sigma\rangle,$$
$$\langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle = \langle P_1 \cup \{y_1 : u_1 \triangleq v_2, y_2 : u_2 \triangleq v_1\} \mid S \mid \sigma\rangle.$$

In any case $S = S'$ and $\sigma = \sigma'\{y \mapsto g(y_1, y_2)\}$, thus they differs only in the set of unsolved constraints $P \cup \{y : u \triangleq v\}$.

1. If $P \cup \{y : u \triangleq v\} = P_1 \cup \{y_1 : u_1 \triangleq v_1, y_2 : u_2 \triangleq v_2\}$.
   (a) If $y \neq y_1, y_2$, then $\{y : u \triangleq v\} \in P_1$ and the result follows by induction hypothesis.

(b) If $y = y_1$ then $\{y : u \triangleq v\} = \{y_1 : u_1 \triangleq v_1\}$. Therefore, it is necessary to show that (u_1, v_1) is a commutative pair of subterms.
The induction hypothesis applies in $\{y : g(u_1,u_2) \triangleq g(v_1,v_2)\} \in P'$, which implies that $(g(u_1,u_2), g(v_1,v_2))$ is a commutative pair of subterms of s and t. Therefore, by Definition 3.1 it follows that exist positions $q \in \mathtt{pos}(s)$ and $q' \in \mathtt{pos}(t)$ such that $s|_{p'} = g(u_1,u_2)$ and $t|_{q'} = g(v_1,v_2)$. As taking $p = q.1$ and $p' = q'.1$ it follows that

$$s|_p = s|_{q.1} = g(u_1,u_2)|_1 = u_1 \text{ and } t|_{p'} = t|_{q'.1} = g(v_1,v_2)|_1 = v_1.$$

Claim 1: $\mathtt{depth}(p) = \mathtt{root}(p')$.
Since the induction hypothesis implies that $\mathtt{depth}(q) = \mathtt{depth}(q')$, it follows that

$$\mathtt{depth}(p) = 1 + \mathtt{depth}(q) \overset{(I.H.)}{=} 1 + \mathtt{depth}(q') = \mathtt{depth}(p').$$

Then, Claim 1 is true.
Claim 2: For each $0 \leq i < \mathtt{depth}(p)$, $\mathtt{root}(s|_{(p)^i}) = \mathtt{root}(t|_{(p')^i})$.
Since $(s|_q, t|_{q'}) = (g(u_1,u_2), g(v_1,v_2))$ is a commutative pair of subterms of s and t, the induction hypothesis implies that for each $0 \leq i < \mathtt{depth}(q)$, $\mathtt{root}(s|_{(q)^i}) = \mathtt{root}(t|_{(q')^i})$.
The Lemma 1.2 yields that for each $0 \leq i < \mathtt{depth}(q)$ it holds that $(q)^i = (p)^i$. Hence, it follows that

$$\mathtt{root}(s|_{(p)^i}) = \mathtt{root}(s|_{(q)^i}), \text{ and } \mathtt{root}(t|_{(p')^i}) = \mathtt{root}(t|_{(q')^i}).$$

Therefore, for each $0 \leq i < \mathtt{depth}(q)$ it holds that

$$\mathtt{root}(s|_{(p)^i}) = \mathtt{root}(s|_{(q)^i}) = \mathtt{root}(t|_{(q')^i}) = \mathtt{root}(t|_{(p')^i}).$$

Then, it remains shows that **Claim 2** holds for for $i = \mathtt{depth}(q)$ since $\mathtt{depth}(p) = \mathtt{depth}(q) + 1$.

- If $\mathtt{depth}(q) = 0$.
  Then, $s = g(u_1,u_2)$ and $t = g(v_1,v_2)$. Therefore,

  $$\mathtt{root}(s|_{(p)^0}) = g = \mathtt{root}(t|_{(p')^0})$$

  follows by definition of prefix of a position.

- If $\mathtt{depth}(q) > 0$.
  It follows by Lemma 1.1 that $(q)^i = q$ and $(q')^i = q'$. Hence, Lemma 1.2 implies that

$$\begin{aligned}\mathtt{root}(s|_{(p)^i}) &= \mathtt{root}(s|_{(q)^i})\\ &= \mathtt{root}(g(u_1,u_2))\\ &= \mathtt{root}(g(v_1,v_2))\\ &= \mathtt{root}(t|_{(q')^i}) = \mathtt{root}(t|_{(p')^i}).\end{aligned}$$

Thus, Claim 2 is true.

Claim 3: For each $0 < i \leq \mathtt{depth}(p)$,

i) if $\mathtt{root}(s|_{(p)^{i-1}}) = f \in \Sigma_\emptyset$, then $(p)_i = (p')_i$, or

ii) if $\mathtt{root}(s|_{(p)^{i-1}}) = g \in \Sigma_C$, then $(p)_i = ((p')_i \bmod 2) + 1$.

- If $\mathtt{depth}(q) = 0$.
  Then $s = g(u_1,u_2)$, $t = (v_1,v_2)$ and $\mathtt{depth}(p) = 1$. Then, by definition of prefix of a position, $\mathtt{root}(s|_{(p)^{1-1}}) = \mathtt{root}((p)^0) = g$ and by definition of index of a position $(p)_1 = 1 = (p')_1$
- If $\mathtt{depth}(q) > 0$.
  Then, the Lemma 1.3 yields that for each $1 < i \leq \mathtt{depth}(q)$ holds that $(p)^i = (p')^1$ and $(p')^i = (q')^i$. Therefore, by Lemma 1.2 and induction hypothesis it follows that

  i) if $\mathtt{root}(s|_{(p)^{i-1}}) = \mathtt{root}(s|_{(q)^{i-1}}) = f \in \Sigma_\emptyset$, then

$$(p)_i = (q)_i \overset{(I.H)}{=} (q')_i = (p')_i,$$

  ii) if $\mathtt{root}(s|_{(p)^{i-1}}) = \mathtt{root}(s|_{(q)^{i-1}}) = g \in \Sigma_C$,then

$$(p)^i = (q)^i \overset{(I.H.)}{=} ((q')_i \bmod 2) + 1 = ((p')_i \bmod 2) + 1.$$

  By the previous case, for $i = \mathtt{depth}(p)$,

$$\mathtt{root}(s|_{(p)^{i-1}}) = \mathtt{root}(s|_{(q)^{i-1}}) = \mathtt{root}(s|_q) = \mathtt{root}(g(u_1,u_2)) = g \in \Sigma_C.$$

  The Lemma 1.1 also implies that

$$(p)_i = (q.1)_i = 1 = (q'.1)_i = (p'.1)_i.$$

In conclusion, **Claim 1**, **Claim 2** and **Claim 3** implies that (u_1, v_1) is a commutative pair of subterms of s and t.

(c) If $y = y_2$. Then, $\{y : u \triangleq v\} = \{y_2 : u_2 \triangleq v_2\}$. Therefore, it is necessary shows that (u_1, v_2) is a commutative pair of subterms of s and t. The argument is the same just taking $p = q.2$ and $p' = q'.2$.

2. If $P \cup \{y : u \triangleq v\} = P_1 \cup \{y_1 : u_1 \triangleq v_2, y_2 : u_2 \triangleq v_1\}$.

(a) If $y \neq y_1, y_2$ then $\{y : u \triangleq v\} \in P_1$ and the result follows by induction hypothesis.

(b) If $y = y_1$, then $\{y : u \triangleq v\} = \{y_1 : u_1 \triangleq v_2\}$. Therefore, it is necessary to show that (u_1, v_2) is a commutative pair of subterm.
By the previous case, there are positions $q \in \texttt{pos}(s)$ and $q' \in \texttt{pos}(t)$ such that $q' = g(u_1, u_2)$ and $t|_{q'} = g(v_1, v_2)$. As taking $p = q.1$ and $p' = q.2$. The analyses of the root of the subterms of prefix positions p of s and p' of t is the same as did in the **Claim 2** of the previous case because the prefix of a position analyses the terms trees of s and t in positions above p and p'. And, the analyses of index variables did in **Claim 3** is the same as the previous case to $i = 0$ until $i = \texttt{depth}(q)$, for $i = \texttt{depth}(p)$ it follows, by Lemma 1.1 and Lemma 1.3, that

$$\begin{aligned}(p)_i = (q.1)_i = 1 \\ &= (2 \texttt{ mod } 2) + 1 \\ &= ((q'.1)_i \texttt{ mod } 2) + 1 \\ &= ((p')_i \texttt{ mod } 2) + 1\end{aligned}$$

Therefore p and p' still satisfies the conditions of Definition 3.1 because $\texttt{root}(s|_{(p)_{i-1}}) = g \in \Sigma_C$.

(c) If $y = y_2$ then $\{y : u \triangleq v\} = \{y_2 : u_2 \triangleq v_1\}$. Then it is necessary to show that (u_2, v_1) is a commutative pair of subterms. To do it, the argument is analogous as the did above, just taking $p = q.2$ and $p' = q'.1$.

($\Longleftarrow$) Suppose that (u, v) is an commutative pair of subterms of s and t. We want to show that exists a derivation such that

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma \rangle$$

The definition of commutative pair of subterms given in Definition 3.1 says that exist positions $p \in \texttt{pos}(s)$ and $p' \in \texttt{pos}(t)$ such that $s|_p = u$, $t_{p'} = v$ and $\texttt{depth}(p) = \texttt{depth}(p')$. Then proof follows by induction over the depth of p.

Base Case: If $\texttt{depth}(p) = 0$, then $p = \varepsilon$ and since $\texttt{depth}(p) = \texttt{depth}(p')$ it follows that $p' = \varepsilon$ too. Therefore, $s = s|_\varepsilon = u$ and $t = t|_\varepsilon = v$. Hence the result follows for $P = S = \emptyset$ and $\sigma = id$, i.e, $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle = \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$.

Inductive Step: Suppose that $\texttt{depth}(p) = n+1$ and $(s|_p, t_{p'}) = (u, v)$ is a commutative pair of subterms.

Notice that exists positions $q \in \texttt{pos}(s)$ and $q' \in \texttt{pos}(t)$ such that $p = q.i$, $p' = q'.j$, for $i, j \in \mathbb{N}$, and $\texttt{depth}(q) = \texttt{depth}(q') = n$. Since (u, v) is a commutative pair of subterms, the Definition 3.1 yields that $(s|_q, t|_{q'})$ it is also a commutative pair of subterms. Then, the induction hypothesis implies that exists a derivation of the form

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P_1 \cup \{z : s|_q \triangleq t|_{q'}\} \mid S \mid \sigma'\rangle.$$

We will show that exists a derivation of the form

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle.$$

Since (u, v) is a commutative pair of subterms it follows by definition that

$$\texttt{root}(s|_{(p)^{n+1}}) = \texttt{root}(s|_q) = \texttt{root}(t|_{q'}) = \texttt{root}(t|_{(p')^{n+1}}).$$

then $\texttt{root}(s|_q) = \texttt{root}(t|_{q'})$. There are two cases to analysing depending on if $\texttt{root}(s|_q) = \texttt{root}(t|_{q'})$ is an free function symbol $f \in \Sigma_\emptyset$ or an commutative function symbol $g \in \Sigma_C$.

1. If $\texttt{root}(s|_q) = \texttt{root}(t|_{q'}) = f \in \Sigma$.

   Then, $s|_q = f(\overline{u_n})$ and $t|_{q'} = f(\overline{v_n})$, where $u = s|_{p.i}$ is the i-th argument of $f(\overline{u_n})$ and $v = t|_{q'.j}$ is the j-th argument of $f(\overline{v_n})$, i.e, $u = u_j$ and $v = v_j$.

   Hence, the induction hypothesis implies that

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P_1 \cup \{z : f(\overline{u_m}) \triangleq f(\overline{v_m})\} \mid S \mid \sigma'\rangle \qquad (1)$$

By Definition 3.1 it follows that the index of p and p' must be equals at depth n, in other words, $(p')_n = (p')_n$. On the other hand, by Lemma 1.1 it follows that $(p)_n = (q.i)_n = i$ and $(p')_n = (q'.j)_n = j$. Then, $i = j$ and $(u,v) = (u_i,v_i)$.
As applying (Dec) in (1) we obtain the following derivation

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P \cup \{y : u_i \triangleq v_i\} \mid S \mid \sigma\rangle,$$

where $P = P_1 \cup (\bigcup_{k=1}^{n} y_k : u_k \triangleq v_k)/\{y_i : u_i \triangleq v_i\}$, $\sigma = \sigma'\{y \mapsto f(\overline{y_m})\}$ and $y = y_i$.

2. If $\mathtt{root}(s|_q) = \mathtt{root}(t|_{q'}) = g \in \Sigma_C$.
Then $s|_q = g(u_1,u_2)$ and $t|_q = g(v_1,v_2)$. Where $u = u_i$, with $i = 1,2$ and $v = v_j$ with $j = 1,2$. By the induction hypothesis it follows that exists a derivation such that

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P_1 \cup \{y : g(u_1,u_2) \triangleq g(v_1,v_2)\} \mid S \mid \sigma'\rangle$$

there are two cases to analysing.

a. If $i = (p)_n = (p')_n = j$, it follows that $u = u_i$ and $v = u_i$, with $i = 1,2$, in other worlds: $(u,v) = (u_1,v_1)$ or $(u,v) = (u_2,v_2)$. In both cases, applying *(C-Dec)* in the least configuration of the derivation we obtain that

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_y} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle,$$

where $P = P_1 \cup \{y_j : u_j \triangleq v_j\}$, $\sigma = \sigma'\{y \mapsto g(y_1,y_2)\}$ and $y = y_i$.

b. If $i = (p)_n = ((p')_n \ \mathtt{mod}\ 2) + 1 = (j \ \mathtt{mod}\ 2) + 1$, it follows that $u = u_i$ and $v = v_{(i \ \mathtt{mod}\ 2)+1}$, in other worlds: $(u,v) = (u_1,v_2)$ or $(u,v) = (u_2,v_1)$. In both cases, applying *(C-Dec)* in the last configuration we obtain that

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$$

where $P = P_1 \cup \{y_j : u_j \triangleq v_i\}$, $\sigma = \sigma'\{y \mapsto g(y_1,y_2)\}$ and $y = y_i$.

□

Lemma B.4 (Lemma 15 in [1])**.** Let $\mathcal{A}_C\langle s,t\rangle$ be an AUP$_C$. There is a derivation of the form $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P \mid \{u \triangleq v\} \cup S \mid \sigma\rangle$ if, and only if, (u,v) is a commutative conflict pair of s and t.

Proof.

($\Longrightarrow$) Suppose that $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P \mid \{y : u \triangleq v\} \cup S \mid \sigma \rangle$ then the Solve rule was applied in some step in the constraint $\{y : u \triangleq v\}$. Then, it is possible unfolds the derivation as:

$$\begin{aligned}\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle &\overset{*}{\Longrightarrow}_{\texttt{AUnif}_C} \langle \{y : u \triangleq v\} \cup P' \mid S' \mid \sigma' \rangle \\ &\Longrightarrow_{(Sol)} \langle P' \mid \{y : u \triangleq v\} \cup S' \mid \sigma' \rangle \\ &\overset{*}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P \mid \{y : u \triangleq v\} \cup S \mid \sigma \rangle\end{aligned}$$

Thus, by Lemma 3.2, (u, v) is an commutative pair of subterms of s and t by Definition 3.1. Besides, since Solve rule was applied in $\{y : u \triangleq v\}$ it follows that $\texttt{root}(u) \neq \texttt{root}(v)$, therefore (u, v) is one commutative conflict pair of subterms of s and t by Definition 3.2.

($\Longleftarrow$) If (u, v) is one commutative conflict pair of s and t then by definition (u, v) is also one commutative pair of subterms of s and t, therefore Lemma 3.2 hence there exists a derivation of the form $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \Longrightarrow \langle \{y : u \triangleq v\} \cup P \mid S \mid \sigma \rangle$. There are two cases, depending on the constraints of S.

1. If there is no constraint $\{z : u \triangleq s\} \in S$, then the result follows by application of (Sol).
2. If there exists a constraint $\{z : u \triangleq v\} \in S$, then (Sol) was applied in a previous step of the derivation. Then the result follows immediately.

$\square$

Lemma B.5. Let $\mathcal{A}_C\langle s, t \rangle$ be an AUP$_C$. If exists a derivation of the form

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_C} \langle \{y : u \triangleq v\} \cup P \mid S \mid \sigma \rangle$$

then there exist positions $p \in \texttt{pos}(s), p' \in \texttt{pos}(t)$ such that

$$(x\sigma)|_p = y, \quad s|_p = (x\sigma)|_p\{y \mapsto u\} = u, \quad t|_{p'} = (x\sigma)|_p\{y \mapsto v\} = v,$$

Proof. By induction on the length of n.

Base Case: if $n = 0$ then $\langle P \mid S \mid \sigma \rangle = \langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle$. Then, taking $p = p' = \varepsilon$, it follows that $s|_\varepsilon = (xid)\{x \mapsto s\} = s, t|_\varepsilon = (xid)\{x \mapsto t\} = t$ and $(xid)|_\varepsilon = x$.

Inductive Step: Suppose that $n > 0$ and consider the reduction

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \xRightarrow{n}_{\texttt{AUnif}_C} \langle P' \mid S' \mid \sigma'\rangle \Longrightarrow_{(R)} \langle \{y : u \triangleq v\} \cup P \mid S \mid \sigma\rangle.$$

We want to show that exist positions $p \in \texttt{pos}(s)$, $p' \in \texttt{pos}(t)$ such that $(x\sigma)|_p = y$, $s|_p = (x\sigma)|_p\{y \mapsto u\} = u$ and $t|_{p'} = (x\sigma)|_p\{y \mapsto v\} = v$.

The proof proceeds by analysing of each inference rule of Figure 3.1. The cases for (Dec), (Sol) and (Rec) are similar to the proof of Lemma 2.5, therefore to complete the proof only remains the case for *(C-Dec)*.

Case *(R)* = *(C-Dec)* :

Then, the reduction is of the form

$$\begin{tikzcd}
 & \langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \arrow[d, Rightarrow, "*", "\texttt{AUnif}_C"'] & \\
 & \langle P_1 \cup \{z : g(u_1,u_2) \triangleq g(v_1,v_2)\} \mid S \mid \sigma'\rangle \arrow[dl, Rightarrow, "(C\text{-}Dec)"'] \arrow[dr, Rightarrow, "(C\text{-}Dec)"] & \\
\langle P_1 \cup \{z_1 : s_1 \triangleq t_1, z_2 : s_2 \triangleq t_2\} \mid S \mid \sigma\rangle & & \langle P_1 \cup \{z_1 : s_1 \triangleq t_2, z_2 : s_2 \triangleq t_1\} \mid S \mid \sigma\rangle,
\end{tikzcd}$$

where, $P' = P_1 \cup \{z : g(u_1,u_2) \triangleq g(v_1,v_2)\}$, $S = S'$ and $\sigma = \sigma'\{z \mapsto g(z_1,z_2)\}$. There are two cases for P, depending on how *(C-Dec)* was applied.

- **Case 1:** If $P = P_1 \cup \{z_1 : u_1 \triangleq v_1, z_2 : u_2 \triangleq v_2\}$.
  There are two cases to consider:
  1. If $y \neq z_1, z_2$ then y, then $z : u \triangleq v \in P'$ and the result follows by induction hypothesis.
  2. If $y = z_i$ for some $i = 1,2$, them $u = u_i$ and $v = v_i$, i.e, u is the i-th argument of $g(u_1,u_2)$ and v is the i-th argument of $g(v_1,v_2)$. Notice that the induction hypothesis holds for $\{z : g(u_1,u_2) \triangleq g(v_1,v_2)\}$ and it follows that exist positions $q \in \texttt{pos}(s)$, $q' \in \texttt{pos}(t)$ such that $(x\sigma') = z$, $s|_q = (x\sigma)|_q\{z \mapsto g(u_1,u_2)\}$ and $t|_{q'} = (x\sigma)|_q\{z \mapsto g(v_1,v_2)\} = g(v_1,v_2)$. As take $p = q.i$ and $p' = q'.1$ this prove follows similarly as the proof of Lemma2.5.
- **Case 2:** If $P = P_1 \cup \{z_1 : u_1 \triangleq v_2, z_2 : u_1 \triangleq v_1\}$.
  There are two cases to consider:
  1. If $y \neq z_1, z_2$ then y, then $z : u \triangleq v \in P'$ and the result follows by induction hypothesis.

2. If $y = z_i$ for some $i = 1,2$, them $u = u_i$ and $v = v_{(i+1)\mathtt{mod2}}$, i.e, $(u,v) = (u_1,u_2)$ or $(u,v) = (u_2,v_1)$. In both cases, by the induction hypothesis it follows that exist positions $q \in \mathtt{pos}(s)$, $q' \in \mathtt{pos}(t)$ such that $(x\sigma') = z$, $s|_q = (x\sigma)|_q\{z \mapsto g(u_1,u_2)\}$ and $t|_{q'} = (x\sigma)|_q\{z \mapsto g(v_1,v_2)\} = g(v_1,v_2)$ and the proof is similar to the proof of Lemma 2.5, just
   (a) taking $p = q.1$ and $p' = p.2$ if $(u,v) = (u_1,v_2)$, or
   (b) taking $p = q.2$ and $p' = q'.2$ if $(u,v) = (u_2,v_1)$.

□

Lemma B.6 (Lemma 16 in [1])**.** Given terms $s,t \in T(\mathcal{X},\Sigma_{\emptyset\cup C})$, if $u \in \mathtt{gen}_C(s,t)$ then there is a derivation $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P \mid S \mid \sigma\rangle$ such that $u \equiv_C x\sigma$.

Proof. By induction on the structure of u.

Case 1: $u = x$**.**

Then the initial configuration $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ outputs $x\,id = x$ trivially.

Case 2: $u = f(u_1,\dots,u_n)$**.**

The proof is similar to the proof of Lemma 2.7.

Case 3: $u = g(u_1,u_2)$**.**

There is a pair of substitutions $\overline{\theta} = (\theta_1,\theta_2)$ such that $u\overline{\theta} \equiv_C (s,t)$, that is s and t are of the form $s = g(s_1,s_2)$ and $t = g(t_1,t_2)$. Therefore, rule *(C-Dec)* applies in the initial configuration, as follows:

$$\begin{tikzcd}
& \langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \arrow[dl, Rightarrow, "\textit{(C-Dec)}"'] \arrow[dr, Rightarrow, "\textit{(C-Dec)}"] & \\
\langle\{x_1 : s_1 \triangleq t_1, x_2 : s_2 \triangleq t_2\} \mid \emptyset \mid \underbrace{\{x \mapsto g(x_1,x_2)\}}_{\sigma_0}\rangle & & \langle\{y_1 : s_1 \triangleq t_2, y_2 : s_2 \triangleq t_1\} \mid \emptyset \mid \underbrace{\{x \mapsto g(y_1,y_2)\}}_{\sigma_0'}\rangle
\end{tikzcd}$$

depending on the instances of the subterms u_1,u_2 of u, there are four cases:

1. $u_1 \in \mathtt{gen}_C(s_1,t_1)$ and $u_2 \in \mathtt{gen}_C(s_2,t_2)$.

   Then induction hypothesis in hence there are reductions

$$\langle\{x_1 : s_1 \triangleq t_1\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{(R_1)} \langle C_1 \mid S_1 \mid \sigma_1\rangle,$$
$$\langle\{x_1 : s_1 \triangleq t_2\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{(R_2)} \langle C_2 \mid S_2 \mid \sigma_2\rangle$$

where $(R_1), (R_2)$ represents derivations of $\mathtt{AUnif}_C$ simplification rules.

The deal is combine this derivations like we did in the prove of Lemma 2.7. This combination is analogous as the combination did in the prove of Lemma 2.7. Then, we obtain

$$\langle \left\{ \begin{matrix} x_1 : s_1 \triangleq t_1 \\ x_2 : s_2 \triangleq t_2 \end{matrix} \right\} \mid \emptyset \mid \sigma_0 \rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P' \mid S' \mid \sigma \rangle$$

where σ is such that $x\sigma =_C u \in \mathtt{gen}_C(s,t)$.

2. $u_1 \in \mathtt{gen}_C(s_1,t_2)$ and $u_2 \in \mathtt{gen}_C(s_2,t_1)$.

   This combination is similar to what was done in the previous case, obtaining

$$\langle \left\{ \begin{matrix} y_1 : s_1 \triangleq t_2 \\ y_2 : s_2 \triangleq t_1 \end{matrix} \right\} \mid \emptyset \mid \sigma_0 \rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_C} \langle P' \mid S' \mid \sigma \rangle$$

3. $u_1 \in \mathtt{gen}_C(s_2,t_2)$ and $u_2 \in \mathtt{gen}_C(s_2,t_1)$.

   This generalizer is equivalent modulo C as the generalizer of the (Case 1), then the result follows.

4. $u_1 \in \mathtt{gen}_C(s_2,t_1)$ and $u_2 \in \mathtt{gen}_C(s_1,t_2)$:

   This generalizer is equivalent modulo C as the generalizer of the **Case 2**, then the result follows.

□

Appendix C

Appedinx of Chapter 4

C.1 Proofs of Section 4.3

Lemma C.1 (Uniqueness of generalization variables cf. Lemma 17 in [1].)**.** Let $\mathcal{A}_A\langle s,t\rangle$ be an AUP$_A$. If $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle$ then y does not appear in any constraint in P or S.

Proof. The proof is by induction on the length n of the derivation

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle.$$

Induction Base: If $n = 0$ then $\langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle = \langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$, i.e, $P = \emptyset$, $S = \emptyset$ and $\sigma = id$. Therefore, x does not appear in any constraint of $P \cup S = \emptyset$.

Inductive Step: If $n > 0$ let's split the derivation in

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P' \mid S' \mid \sigma'\rangle \Longrightarrow_{(R)} \langle P \mid S \mid \sigma\rangle$$

The proof follows by analyses of each inference rule (R) of Figure 4.1 applies in the n-th of the derivation. For rules (Dec), (Sol) and (Rec) the proof was already done in Lemma 2.1. Therefore, only *(A-Left)* and *(A-Right)* cases remains.

Case *(R)=(A-Left)*:

Then the n-step of the reduction is of the form

$$\langle P_1 \cup \{y : h(\overline{u_n}) \triangleq h(\overline{v_m})\} \mid S \mid \sigma' \rangle$$
$$\Big\Downarrow_{((A\text{-}Left))}$$
$$\langle P_1 \cup \begin{Bmatrix} y_1 : h(u_1,\dots,u_k) \triangleq u_1, \\ y_2 : h(u_{k+1},\dots,u_n) \triangleq h(v_2,\dots,v_m) \end{Bmatrix} \mid S \mid \sigma \rangle$$

There are two cases to analysing

1. If $y \neq y_1, y_2$ then $\{y : u \triangleq v\} \in P_1$ and the result follows by induction hypotheses,
2. If $y = y_1$ or $y = y_2$. The result follows by the freshness of y_1 and y_2.

Case *(R) = (A-Right)*:

The argument is analogue as the previous case.

□

Lemma C.2 (Range of substitutions c.f. in Lemma 18 of [1]). Let $\mathcal{A}_A\langle s,t\rangle$ be an AUP$_A$. If

$$\{x : s \triangleq t \mid \emptyset \mid id\} \Longrightarrow_{\mathtt{AUnif}_A} \langle P \mid S \mid \sigma \rangle$$

then

$$\mathtt{Index}(P \cup S) \subseteq \mathtt{vran}(\sigma) \cup \{x\} \text{ and } \mathtt{vran}(\sigma) \subseteq \mathcal{V}(x\sigma).$$

Proof. By induction on the length $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P \mid S \mid \sigma \rangle$.

Induction base: If $n = 0$ then $\langle P \mid S \mid \sigma\rangle = \langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle$, thus $P = \{x : s \triangleq t\}$, $S = \emptyset$ and $\sigma = id$. Therefore, the result follows by Definition 1.1, 1.7 and 2.2, i.e, $\mathtt{vran}(\sigma) = \emptyset \subseteq \{x\} = \mathcal{V}(x\sigma)$ and $\mathtt{Index}(P \cup S) = x \subseteq \{x\}$

Induction step: If $n > 0$, the reduction unfolds as

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{n-1}{\Longrightarrow}_{\mathtt{AUnif}_A} \langle P' \mid S' \mid \sigma' \rangle \Longrightarrow_{(R)} \langle P \mid S \mid \sigma \rangle$$

The proof follows by analyses of each inference rule (R) of Figure 4.1, where the cases where (R) is (Dec), (Sol) or (Rec) were already verified in the proof of Lemma 2.2. Thus, remains analyses the cases which (R) is *(A-Left)* or *(A-Left)*.

Case *(R) = (A-Left)*:

Thus, the n-th of the reduction is of the form

$$\begin{array}{c}\langle P_1 \cup \{y : h(\overline{u_n}) \triangleq h(\overline{v_m})\} \mid S \mid \sigma' \rangle \\ \Big\Downarrow \scriptstyle{((A\text{-}Left))} \\ \langle P_1 \cup \left\{ \begin{array}{l} y_1 : h(u_1, \ldots, u_k) \triangleq u_1, \\ y_2 : h(u_{k+1}, \ldots, u_n) \triangleq h(v_2, \ldots, v_m) \end{array} \right\} \mid S \mid \sigma \rangle \end{array}$$

With $P' = P_1 \cup \{y : h(\overline{u_n}) \triangleq h(\overline{v_m})\}$. Notice that are $n-1$ possibilities for P, depending on how *(A-Left)* was applied, i.e, one possibility for each $1 \leq k \leq n-1$,

$$P = P_1 \cup \{y_1 : h(u_1, \ldots, u_k) \triangleq u_1, y_2 : h(u_{k+1}, \ldots, u_n) \triangleq h(v_2, \ldots, v_m)\}.$$

In any case $S = S'$ and $\sigma = \sigma'\{y \mapsto h(y_1, y_2)\}$.

The definition of index variables hence that

$$\texttt{Index}(P \cup S) \subseteq \texttt{Index}(P_1 \cup S) \cup \{y_1, y_2\} \cup \{x\}.$$

Since $\sigma = \sigma'\{y \mapsto h(y_1, y_2)\}$ it follows that $\texttt{vran}(\sigma) = (\texttt{vran}(\sigma') \cup \{y_1, y_2\}) / \{y\}$. On the other hand, the variable y does not appear in any constraint of $S \cup P$. Hence, the result follows by induction hypotheses, i.e. $\texttt{Index}(P \cup S) \subseteq \texttt{vran}(\sigma)$.

Besides, $\sigma = \sigma'\{y \mapsto h(y_1, y_2)\}$ implies that $\mathcal{V}(x\sigma) = (\mathcal{V}(x\sigma') \cup \{y_1, y_2\} / \{y\}$. Then, by induction hypotheses it follows that $\texttt{vran}(\sigma) \subseteq \mathcal{V}(x\sigma)$.

Case *(R) = (A-Right)*: The proof is analogous as the previous case.

□

Lemma C.3 (Lemma 20 in [1]). Given flattened terms s, t of terms in $T(\mathcal{X}, \Sigma_{\emptyset \cup A})$ then there exists a sequence $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P \mid \{y : u \triangleq v\} \cup S \mid \sigma \rangle$ if, and only if (u, v) is an associative conflict pair of s and t.

Proof.

($\Longrightarrow$) Suppose that is a sequence $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_C} \langle P \mid \{y : u \triangleq v\} \cup S \mid \sigma \rangle$, we want to show that (u, v) is a conflict pair of subterms of s and t.

Since $\{y : u \triangleq v\}$ in the set of solved constraints, which implies that (Sol) rule was applied at some step in the constraint $\{y : u \triangleq v\}$. Then, the reduction can be unfolded as

$$\begin{aligned}\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle\{y : u \triangleq v\} \cup P' \mid S' \mid \sigma'\rangle \\ &\Longrightarrow_{(Sol)} \langle P' \mid \{y : u \triangleq v\} \cup S'\} \mid \sigma'\rangle \\ &\overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P \mid \{y : u \triangleq v\} \cup S\} \mid \sigma\rangle\end{aligned}$$

Then, Lemma 4.4 in $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle\{y : u \triangleq v\} \cup P' \mid S' \mid \sigma'\rangle$ hence (u,v) is an associative pair of subterms. And, by definition of (Sol) rule, $\texttt{root}(u) \neq \texttt{root}(v)$. Therefore the result follows by definition of associative conflict pair of subterms.

$(\Longleftarrow)$ Suppose that (u,v) is an associative conflict pair of s and t. We want to show that there exists a derivation such that $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P \mid \{y : u \triangleq v\} \cup S \mid \sigma\rangle$.

By definition, (u,v) is an associative pair of s and t such that $\texttt{root}(u) \neq \texttt{root}(v)$. Since the definition of associative pair have two cases, this proof is divided accordingly.

Case 1: $(u,v) = (s|_p, t_{p'})$ and (p,p') is an associative pair of positions of s and t. Then, Lemma 4.2 implies that there exists a sequence

$$\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P \cup \{y : u \triangleq v\} \mid S \mid \sigma\rangle = C.$$

The results follows by an application of (Sol) in C.

Case 2: $u = h(\overline{u_n})$ and $v = h(\overline{v_m})$ there exists an associative pair of positions (p',p) of s and t, such that

$$s|_p = h(s_1,\dots,s_k,u_1,\dots,u_n,s_{k+1}\dots,s_q),$$

$$t|_{p'} = h(t_1,\dots,t_{k'},v_1,\dots,v_m,t_{k'+1},\dots,t_{q'})$$

There are tree cases to be analysed, depending on the arguments of $s|_q$ that appears before and after $u_1,\dots,u_n$.

1. If both $\{s_1,\dots,s_k\} = \emptyset$ and $\{s_{k+1},\dots,s_q\} = \emptyset$.
   Then, by Definition 4.3 it follows that $\{t_1,\dots,t_{k'}\} = \emptyset$ and $\{t_{k'+1},\dots,t_{q'}\} = \emptyset$. Thus, $s|_{p'} = h(\overline{u_n}) = u$ and $t|_{p'} = h(\overline{v_m}) = v$ and the result follows by the **Case 1.**
2. If $\{s_1,\dots,s_k\} = \emptyset$ and $\{s_{k+1},\dots,s_q\} \neq \emptyset$.
   Notice that if both $n,m > 1$, then $u = h(u_1,\dots,u_n)$, $v = h(v_1,\dots,v_m)$ and $\texttt{root}(u) = \texttt{root}(v)$, which yields that (u,v) is not an associative conflict pair of s and t and there is nothing to show. Therefore, we will only analysing the cases where $(u,v) = (u_1, h(\overline{v_m}))$ and $(u,v) = (h(\overline{u_n}), v_1)$.

(a) $(u,v) = (u_1, h(\overline{v_m}))$.
Then, by Definition 4.3 it follows that $\{t_1,\ldots,t_{k'}\} = \emptyset$ and $\{t_{k'+1},\ldots,t_{q'}\} \neq \emptyset$. Therefore, $s|_p = h(u_1, s_{k_1},\ldots,s_q)$ and $t|_{p'} = (v_1,\ldots,v_m,t_{k'+1}\ldots,t_{q'})$. By Lemma 4.2 it follows that exists a reduction of the form

$$\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P' \cup \{z : (s|_p) \triangleq (t|_{p'})\} \mid S \mid \sigma'\rangle. \quad (1)$$

Hence,

$$\begin{array}{c}
\langle P' \cup \{z : (s|_p) \triangleq (t|_{p'})\} \mid S \mid \sigma'\rangle \\
\Big\Downarrow \textit{(A-Right)} \\
\langle P' \cup \left\{ \begin{array}{l} y_1 : u_1 \triangleq h(v_1,\ldots,v_m), \\ y_2 : h(u_2,\ldots,s_q) \triangleq h(v_{k'+1},\ldots,v_{q'}) \end{array} \right\} \mid S \mid \underbrace{\sigma'\{z \mapsto h(y_1,y_2)\}}_{\sigma}\rangle \\
\Big\Downarrow \textit{(Sol)} \\
\langle P \mid \{y_1 : u_1 \triangleq h(v_1,\ldots,v_m)\} \cup S \mid \sigma\rangle
\end{array}$$

Where $P = P' \cup \{y_2 : h(u_2,\ldots,s_q) \triangleq h(v_{k'+1},\ldots,v_{q'})\}$. Thus, the result follows as taking $y = y_1$ and by the transitivity of $\overset{*}{\Longrightarrow}_{\texttt{AUnif}_A}$.

(b) $(u,v) = (h(\overline{u_n}), v_1)$.
Thus, $s|_p = h(u_1,\ldots,u_n,s_{k_1}\ldots,s_q)$ and $t|_{p'} = h(v_1,t_{k'+1},\ldots,t_{q'})$.
The result follows by analogous argument, just applying *(A-Left)* in (1) instead of *(A-Right)*.

3. If $\{s_1,\ldots,s_k\} \neq \emptyset$ and $\{s_{k+1},\ldots,s_q\} = \emptyset$.
The argument is analogous as the previous case.

□

Lemma C.4 (Lemma 21 in [1]). Let $\mathcal{A}_A\langle s,t\rangle$ be an AUP$_A$. If $u \in \text{gen}_A(s,t)$ then there exist a sequence $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P \mid S \mid \sigma\rangle$ such that $u \equiv_A x\sigma$.

Proof. The proof is by structural induction on term $x\sigma$

Case 1: $u = x$.

Then the initial configuration $\langle \{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ outputs $x\,id = x$ trivially.

Case 2: $u = f(u_1,\ldots,u_n)$, with $f \in \Sigma_\emptyset$.

The proof is similar to the proof of Lemma 2.7.

Case 3: $u = h(u_1,\dots,u_k)$, with $h \in \Sigma_A$.

Then there is a pair of substitutions $\overline{\theta} = (\theta_1,\theta_2)$ such that $\theta_1 = s$ and $\theta_2 = t$. Therefore, $\texttt{root}(s) = \texttt{root}(t) = h \in \Sigma_A$ which implies that s and t are of the form $s = h(s_1,\dots,s_n)$ and $t = h(t_1,\dots,t_m)$.

Notice that $k \leq \texttt{min}(n,m)$ and that each u_i generalizers an associative pair of subterms (s_i',t_i') of s and t. Further, by induction hypotheses it follows that exist derivations

$$\langle\{x_i : s_i' \triangleq t_i'\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{(R_i)} \langle C_i \mid S_i \mid \sigma_i\rangle$$

such that $u_i \equiv_A (x\sigma_i)$ for each $i \leq k$. Where (R_i), for $i \leq k$, represent derivations of $\texttt{AUnif}_A$ simplification rules.

The deal is to combine the derivations (R_i) like we did in the prove of Lemma 2.7. However, to it is necessary first to find a derivation $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \overset{*}{\Longrightarrow}_{\texttt{AUnif}_A} \langle P' \mid S' \mid \sigma'\rangle$ such that each constraint $\{x_i : s_i' \triangleq t_i'\}$ occurs on P' and σ' is a unflattened form of $h(x_1,\dots,x_n)$.

1. $k = m = n$.

   Then $u_i \in \texttt{gen}_A(s_i,t_i)$, for each $1 \leq i \leq n$, and by Lemma 4.2 it follows that exist a derivation of the form:

$$\begin{aligned}
\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle &\Longrightarrow_{\textit{(A-Left)}} \langle\{x_1 : s_1 \triangleq t_1, y_1 : h(s_2,\dots,s_n) \triangleq h(t_2,\dots,t_n)\} \mid \sigma'\rangle \\
&\Longrightarrow_{\textit{(A-Left)}} \langle P_1 \cup \begin{Bmatrix} x_1 : s_1 \triangleq t_1, \\ x_2 : s_2 \triangleq t_2, \\ y_2 : h(s_3,\dots,s_n) \triangleq h(t_3,\dots,t_n) \end{Bmatrix} \mid \emptyset \mid \sigma''\rangle \\
&\overset{*}{\Longrightarrow}_{\textit{(A-Left)}} \langle\{x_1 : s_1 \triangleq t_1,\dots,x_n : s_n \triangleq t_n\} \mid \emptyset \mid \sigma'''\rangle
\end{aligned}$$

   with $\sigma' = \{x \mapsto h(x_1,y_1)\}$, $\sigma'' = \{x \mapsto h(x_1,h(x_2,\dots h(x_n,y_2)))\}$ and

$$x\sigma''' = h(x_1,h(x_2,h(\dots,h(x_{n-1},x_n)))),$$

   which implies that $x\sigma'''$ is a unflattened form of $h(x_1,\dots,x_n)$.

   Therefore, the result follows as combine the derivations (R_i) with $i \leq 1$ like we did in Lemma 2.7.

2. $n \neq m$.

   Then in $u = h(u_1,\dots,u_n)$ there exist u_j such that $u_j \in \texttt{gen}_A(s_l,h(t_{l'},\dots,t_{r'}))$ or $u_j \in \texttt{gen}_A(h(s_l,\dots,s_r),t_l)$.

(a) if $u_j \in \texttt{gen}_A(s_l, h(t|_{l'}), \dots, t|_{r'})$ then by Lemma 4.2 and Lemma 4.3 it follows that exist a reduction

$$\begin{array}{c} \langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle \\ \texttt{AUnif}_A \Big\Downarrow * \\ \langle \{x_n : s_1 \triangleq t_1, \dots, x_l : s_l \triangleq h(t_{l'}, \dots, t_{r'}), \dots, x_k : s_k \triangleq t_k\} \mid \emptyset \mid \sigma' \rangle. \end{array}$$

Where, by Lemma 4.1, it follows that $x\sigma'$ is a unflattened version of $h(x_1, \dots, x_n)$. Thus, the result follows as combine the derivations (R_i) with $i \le k$ like we did in Lemma 2.7.

(b) If $u_j \in \texttt{gen}_A(h(s_l, \dots, s_r), t_l)$, the proof is analogous as the case above.

$\square$

C.2 Remarks of Chapter 4

C.2.1 Counter Example For Lemma 19 in [1].

The Lemma 19 in [1] (see Figure C.1) says that for every associative pair of subterms (u, v), exist a constraint representing the AUP$_A$ for u and v, $\{y : u \triangleq v\}$, in some derivation of $\texttt{AUnif}_A$.

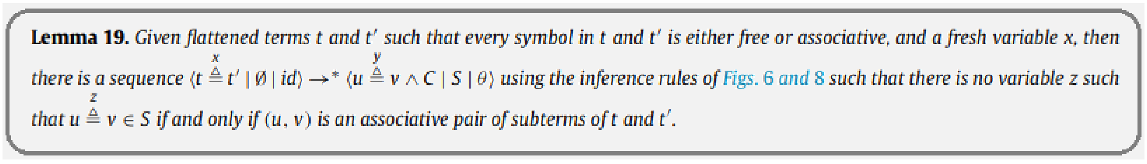

Lemma 19. *Given flattened terms t and t' such that every symbol in t and t' is either free or associative, and a fresh variable x, then there is a sequence $\langle t \overset{x}{\triangleq} t' \mid \emptyset \mid id \rangle \longrightarrow^* \langle u \overset{y}{\triangleq} v \wedge C \mid S \mid \theta \rangle$ using the inference rules of Figs. 6 and 8 such that there is no variable z such that $u \overset{z}{\triangleq} v \in S$ if and only if (u, v) is an associative pair of subterms of t and t'.*

Fig. C.1 Lemma 19 in [1].

However, the rules of $\texttt{AUnif}_A$ presented in Figure 4.1 applied in $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle$ does not produce constraints that represents all associative pairs of subterms during its simplification tree. As instance, for flattened terms $s = h(a, b, c, d)$ and $t = h(a', b', c', d')$, with $h \in \Sigma_A$. The Definition 4.3 yields that the following pairs of subterms are associative ones:

$$(h(a,b), h(a',b')),\ (h(a,b,c), h(a',b')),\ (h(a,b), h(a',b',c')) \text{ and } (h(b,c), h(b',c')) \qquad (\star)$$

And, applying the rules of Figure 4.1 in $\langle \{x : s \triangleq t\} \mid \emptyset \mid id \rangle$, we obtain that the derivation tree of $\texttt{AUnif}_A(s, t)$ is of the form illustrated bellow. In this figure we use $\overset{*}{\Longrightarrow}$ to denote $\overset{*}{\Longrightarrow}_{\texttt{AUnif}_A}$ for short.

$$\begin{tikzcd}
 & & & \langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle \arrow[dlll, Rightarrow, "\star"'] \arrow[dl, Rightarrow, "\star"'] \arrow[dr, Rightarrow, "\star"] \arrow[drr, Rightarrow, "\star"] \arrow[drrrr, Rightarrow, "\star"] & & & & \\
C_1 \arrow[d, Rightarrow, "\star"'] \arrow[dr, Rightarrow, "\star"] & & C_2 \arrow[d, Rightarrow, "\star"'] \arrow[dr, Rightarrow, "\star"] & & C_3 \arrow[ddd, Rightarrow, "\star"'] & C_4 \arrow[d, Rightarrow, "\star"'] \arrow[dr, Rightarrow, "\star"] & & C_5 \arrow[ddd, Rightarrow, "\star"'] \\
C_{1.1} \arrow[d, Rightarrow, "\star"'] & C_{1.2} \arrow[dd, Rightarrow, "\star"'] & C_{2.1} \arrow[dd, Rightarrow, "\star"'] & C_{2.2} \arrow[dd, Rightarrow, "\star"'] & & C_{4.1} \arrow[dd, Rightarrow, "\star"'] & C_{4.2} \arrow[dd, Rightarrow, "\star"'] & \\
C_{1.1.1} \arrow[d, Rightarrow, "\star"'] & & & & & & & \\
N_1 & N_2 & N_3 & N_4 & N_5 & N_6 & N_7 & N_8
\end{tikzcd}$$

with

$$\begin{aligned}
C_1 &= \langle\{x_2 : h(b,c,d) \triangleq h(b',c',d')\} \mid \{x_1 : a \triangleq a'\} \mid \{x \mapsto h(x_1,x_2)\}\rangle,\\
C_2 &= \langle\{x_2 : h(c,d) \triangleq h(b',c',d')\} \mid \{x_1 : h(a,b) \triangleq a'\} \mid \{x \mapsto h(x_1,x_2)\}\rangle,\\
C_3 &= \langle\{x_1 : h(a,b,c) \triangleq a', x_2 : d \triangleq h(b',c',d')\} \mid \emptyset \mid \{x \mapsto h(x_1,x_2)\}\rangle,\\
C_4 &= \langle\{x_2 : h(b,c,d) \triangleq h(c',d')\} \mid \{x_1 : a \triangleq h(a',b')\} \mid \{x \mapsto h(x_1,x_2)\}\rangle,\\
C_5 &= \langle\{x_1 : a \triangleq h(a',b',c'), x_2 : h(b,c,d) \triangleq d'\} \mid \emptyset \mid \{x \mapsto h(x_1,x_2)\}\rangle,\\
C_{1.1} &= \langle\{x_3 : b \triangleq b', x_4 : h(c,d) \triangleq h(c',d')\} \mid \{x_1 : a \triangleq a'\}\{x \mapsto h(x_1,h(x_3,x_4))\}\rangle,\\
C_{1.2} &= \langle\{x_3 : h(b,c) \triangleq b', x_4 : d \triangleq h(c',d')\} \mid \{x_1 : a \triangleq a'\} \mid \{x \mapsto h(x_1,h(x_3,x_4))\rangle,\\
C_{2.1} &= \langle\{x_3 : c \triangleq b', x_4 : d \triangleq h(c,d')\} \mid \{x_1 : h(a,b) \triangleq a'\} \mid \{x \mapsto h(x_1,h(x_3,x_4))\rangle,\\
C_{2.2} &= \langle\{x_3 : c \triangleq h(b',c'), x_4 : d \triangleq d' \mid \{x_1 : h(a,b) \triangleq a'\} \mid \{x \mapsto h(x_1,h(x_3,x_4))\rangle,\\
C_{4.1} &= \langle\{x_3 : b \triangleq c, x_4 : h(c,d) \triangleq d'\} \mid \{x_1 : a \triangleq h(a,b')\} \mid \{x \mapsto h(x_1,h(x_3,x_4))\rangle,\\
C_{4.2} &= \langle\{x_3 : h(b,c) \triangleq c', x_4 : d \triangleq d'\} \mid \{x_1 : a \triangleq h(a,b')\} \mid \{x \mapsto h(x_1,h(x_3,x_4))\rangle,\\
C_{1.1.1} &= \langle\{x_5 : c \triangleq c', x_6 : d \triangleq d'\} \mid \{x_1 : a \triangleq a', x_3 : b \triangleq b'\} \mid \{x \mapsto h(x_1,h(x_3,h(x_5,x_6)))\}\rangle,
\end{aligned}$$

where $N_1,\ldots,N_8$ are the normal forms of $\langle\{x : s \triangleq t\} \mid \emptyset \mid id\rangle$ w.r.t. $\Longrightarrow_{\texttt{AUnif}_A}$. Notice that the application of the rules of Figure 4.1 will generates only repeat constraints, i.e, constraints that had previously appeared in the derivation. Thus, we have described all the constraints that appears in the simplification tree of this problem. Therefore, in this derivation there is no constraint representing any of the associative pairs of subterms listed in ($\star$) and it is a contradiction of Lemma 19 in [1].